Complex Decision Rules in for jRule Algorithm and Performance Analysis

Adem Kikaj
Jozef Stefan International Postgraduate School
Jozef Stefan Institute, Department of Knowledge Technologies
Jamova 39, 1000 Ljubljana, Slovenia
adem.kikaj@ijs.si

Marko Bohanec
Jozef Stefan Institute, Department of Knowledge Technologies
Jamova 39, 1000 Ljubljana, Slovenia
marko.bohanec@ijs.si

ABSTRACT: *DEX (Decision EXpert) is a qualitative multi-criteria decisionmodeling methodology. DEX models are used to evaluate and analyze decision alternatives. An essential component of DEX models are decision rules, represented in terms of decision tables. Decision tables may contain many elementary decision rules and may be difficult to be understood by the decision maker. A more compact and comprehensible representation is obtained by converting elementary decision rules to complex rules. The DEXRule algorithm, which is currently implemented in software DEXi, has been found inefficient with large decision tables. This research is aimed at improving the efficiency of the DEX-Rule algorithm. We propose a novel algorithm, called jRule, which generates complex rules by specialization. According to performance analysis, jRule is indeed more efficient than DEXRule. The compactness of complex rules produced by both algorithms varies and there is no clear winner.*

Keywords: DEX Methodology, Decision Rules, Complex Decision Rules, Algorithm Analysis

Received: 24 September 2018, Revised 23 November 2018, Accepted 30 November 2018

DOI: 10.6025/jnt/2019/10/1/9-17

1. Introduction

Decision-making is a difficult and complex process. During this process, a decision maker (DM) faces several decision alternatives. To choose a particular alternative from the set of possible alternatives, a decision-analysis approach [3, 4] can help to satisfy the aims or goals of a decision maker. Decision analysis [3, 4] 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 develop a decision model, which can help decision makers to evaluate alternatives and to choose the best action.

The decision maker in a decision problem have to deal with multiple and possibly conflicting criteria. Multiple Criteria Decision Analysis (MCDA) or Multiple Criteria Decision Making (MCDM) [3] provides methods for structuring, planning and solving such decision problems. DEX methodology is one of the MCDM methods. DEX is a qualitative multi-criteria decision making methodology [1, 2, 5] aimed at the assessment and analysis of decision alternatives. DEX is supported by software DEXi (http://kt.ijs.si/MarkoBohanec/dexi.html).

DEX models have a hierarchical structure, which represents a decomposition of some decision problem into smaller, less complex sub-problems. DEX models are developed by defining (i) attributes, (ii) scales, (iii) hierarchically structured attributes (the tree of attributes), and (iv) decision rules. In DEX models, attributes are variables that represent properties of decision alternatives. Attributes can be either basic or aggregated. Aggregated attributes have subordinate attributes, while basic attributes do not. Basic attributes represent inputs and aggregate attributes represent outputs (results). A scale represents a set of values that can be assigned to an attribute. Scales are qualitative and can take discrete values like 'excellent', 'acceptable', 'inappropriate', etc. Decision rules represent the mapping of subordinate attributes to an aggregated attribute (see section 2 on more details about decision rules in DEX).

In a DEX model, an aggregated attribute may involve many subordinate attributes (e.g., more than five) in which case the decision table will contain many elementary decision rules and may be difficult to be understood. In order to obtain a more comprehensible representation, the DEXi software implements DEX-Rule, an algorithm that converts elementary decision rules to more compact complex rules. DEX-Rule has been found inefficient in decision tables with many subordinate attributes and many elementary decision rules that map to a single decision value.

This research is aimed at improving the efficiency of the DEXRule algorithm. We propose a novel algorithm, called jRule, which finds complex rules by specialization, i.e., by narrowing down too general rules that are constructed initially. The jRule algorithm performed better regarding the running time. The results generated by both algorithms are guaranteed to cover the whole decision table.

This paper is structured as follows: Section 2 formulates the Decision Rules in DEX, Section 3 presents the DEX-Rule algorithm, Section 4 presents the jRule algorithm, Section 5 presents the comparison of the two algorithms regarding the algorithm complexity and the number and form of complex decision rules that they generate. Section 6 summarizes and concludes the paper.

2. Decision Rules in DEX

In DEX models, attributes can be either basic or aggregated. Aggregated attributes are attributes which depend on their descendants, known as subordinate attributes. Decision rules in DEX define the bottom-up mapping of the scale values of subordinate attributes to the values of the aggregated attribute. An example of such mapping, represented in terms of a decision table, is shown in Table 1. The example is taken from a wellknown model for evaluating cars based on attributes such as buying price, maintaining price, safety, and comfort [1]. The example occurs at the top level (root) of the model and maps the subordinate attributes *PRICE* and *TECH.CHAR* (technical characteristics) to the overall evaluation of a *CAR*. The value scale of the involved attributes are ordered values as follows:

- *PRICE = {high, medium, low}*,
- *TECH.CHAR = {bad, acc, good, exc}*, and
- *CAR = {unacc, acc, good, exc}*.

Each row in Table 1 defines the value of the aggregated attribute *CAR* for each combination of subordinate attributes' values. Therefore, the decision table maps all the combination of *PRICE* and *TECH.CHAR* scale values into the value of *CAR*.

A decision rule consists of the condition and decision part:

if $subAttr1 = value_1$

and $subAttr2 = value_2$

...

and *subAttrn* = $value_n$

then *aggAttr* = *value* (or interval of values)

	PRICE	*TECH.CHAR*	*CAR*
1	high	bad	**unacc**
2	high	acc	**unacc**
3	high	good	**unacc**
4	high	exc	**unacc**
5	medium	bad	**unacc**
6	medium	acc	**acc**
7	medium	good	**good**
8	medium	exc	**exc**
9	low	bad	**unacc**
10	low	acc	**good**
11	low	good	**exc**
12	low	exc	**exc**

Table 1. Decision table with elementary decision rules of DEX model known as CAR Evaluation Model [1]

The condition part is the Cartesian product of the scale values of the subordinate attributes of an aggregated attribute (*subAttr*1, *subAttr*2, …, *subAttrn*). The decision-maker defines the *value* of each decision rule, which might be a single value or an interval of values of the aggregated attribute. Such decision rules are also called *elementary decision rules*, since each rule defines the value for exactly one combination of subordinate attributes' values.

In this way, the first row in Table 1 represents the following elementary rule:

if *PRICE* = *high* **and** *TECH.CHAR* = *bad* **then** *CAR* = *unacc*

An alternative representation of the decision rules can be by an $\underline{n}$ dimensional matrix, depending on the number of subordinate attributes. Figure 1 shows such a representation of Table 1. Here, each cell of the matrix represents one elementary decision rule from the decision table.

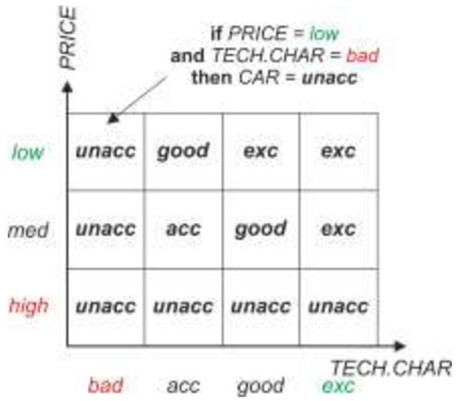

Figure 1. Elementary decision rules represented in a matrix

In order to represent the decision table in a more compact and possibly comprehensible way, DEX uses *complex decision rules*. A complex decision rule consists of the condition and decision value part. In contrast with elementary rules, each clause in the condition part can represent an interval. The decision value is always a single value. Thus, a complex rule generally takes the form:

$$\textbf{if } subAttr_1 \in [low\_value_1,\ high\_value_1]$$
$$\textbf{and } subAttr_2 \in [low\_value_2,\ high\_value_2]$$
$$\ldots$$
$$\textbf{and } subAttr_n \in [low\_valuen\ ,\ high\_valuen]$$
$$\textbf{then } aggAttr = value$$

For comprehensibility, DEXi software traditionally represents intervals as follows:

- '*': the asterisk include all possible scale values of a specific subordinate attribute;
- '>=*w*': stands for *better than or equal to value*;
- '<=*w*': stands *for worse than or equal to value*;
- 'w_1:w_2': interval between value w_1 and value w_2, including the two values.

Figure 2 shows several complex decision rules on the matrix from Figure 1. It is important to notice that each complex decision rule covers an area that corresponds to one or more elementary decision rules. In this way, the number of complex rules that completely cover the matrix is generally lower than the number of elementary rules, and the resulting representation is more compact.

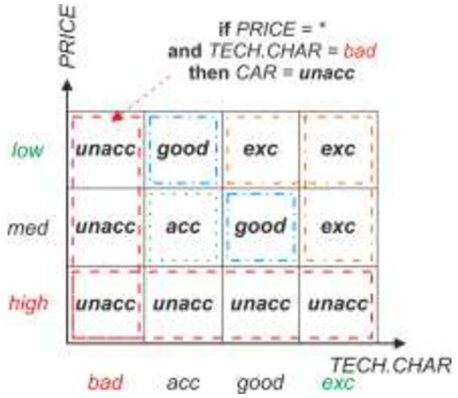

Figure 2. Complex decision rules represented in a matrix through different dotted rectangles for each decision value

3. DEX-Rule Algorithm

DEX-Rule is an algorithm currently implemented in DEXi [1] that converts elementary decision rules into more compact complex decision rules. The DEX-Rule generates complex decision rules by finding areas limited by bounds, which may cover more than one elementary decision rule. An area is represented by two bounds: a low and a high bound. Both are vectors of scale values of the subordinate attributes.

The input to the DEX-Rule algorithm is a decision table, represented in a form of a decision matrix, such as in Figure 1. All the rules are marked as uncovered. The low and high bound (l and h) are vectors (coordinates) that define an area of decision rules with the target value t. Initially, $l = h$, which means that they define a single elementary decision rule. Later, with recursive invocation of the algorithm, these boundaries are gradually extended to cover larger areas with the target value t. On the output, DEX-Rule generates a set of decision rules, such as in the example shown in Table 4. DEX-Rule proceeds by considering all

target decision values, *t*, in succession. For each *t*, DEX-Rule proceeds by generalization, as shown in Algorithm 1.

Algorithm 1. Pseudo-code of the DEX-Rule Algorithm.

Inputs:

l := low bound.

h := high bound.

t := target decision value.

m := last elementary decision rule from decision table (representing the highest current bound).

Outputs:

p := complex decision rules

begin

cover := ValidateBounds(*l*, *h, t*)

if *cover* **then**

for i = 0 **to** |*h*| **do**

if *h*[*i*] < *m*[*i*] **then**

DEXRule(*l*, Increase(*h*), *t*, *m*)

end if

end for

for *i* = 0 **to** |*l*| **do**

if *l*[*i*] > 0 **then**

DEXRule(Decrease(*l*), *h*, *t*, *m*)

end if

end for

p.add(*l* + *h*)

end if

end

For each decision value *t* and each elementary decision rule that has not been covered so far (represented by *l* and *h*, *l* = *h*), DEXRule tries to extend the boundaries *l* and *h* in different directions. When the area cannot be extended any more, a complex decision rule is created. More precisely, a complex decision rule is generated in two cases:

a) Lowest highest scale value of PRICE subordinate attribute b) Finding of increasing decreasing the bounds

Figure 3. Two cases of generating complex decision rules with DEX-Rule algorithm

• When the algorithm reaches the highest or lowest scale value for the specific subordinate attribute, see Figure 3.a, or

• When an extension would cover an elementary decision rules with a different target value, see Figure 3.b.

The process continues until the matrix has been completely covered by complex rules.

4. JRULE Algorithm

The aim of this research was to improve the efficiency of the DEX-Rule algorithm. We propose a novel algorithm, called jRule. While the main idea behind DEX-Rule is to find areas by generalization (extending the area bounds), the main idea of the jRule is to reverse this method and use specialization. jRule proceeds by finding largest areas covering yet uncovered rules for *t* and gradually reducing them.

Algorithm 2. Pseudo-code of the jRule Algorithm.

```
Inputs:
    t := target decision value.
    ger := elementary decision rules for target value t,
           lexicographically sorted by subordinate attribute values.
Outputs:
    p := complex decision rules
begin
    l := lowest subordinate attributes' values from ger
    for i = |ger| to 0
    if !ger[i].isCoveredBy(p) then
        lb = l
        hb = ger[i]
        while !ValidateBounds(lb, hb, t) do
            lb = Increase(lb) // reduce the area by increasing the lb
        end while
        p.add(lb + hb)
    end if
  end for
end
```

The pseudo-code of the jRule algorithm is shown in Algorithm 2. First, the algorithm finds *l*, the lowest bounds for each subordinate attribute of elementary rules for the target value *t*. Then, it locates the last (i.e., highest) currently uncovered elementary rule. This gives the high bound of the area. If the area with bounds *lb* and *hb* is valid, meaning that covers only rules for *t*, a new complex rule is generated. Otherwise, this area is reduced by increasing the low bound *lb*. This process is repeated until all elementary rules for *t* have been covered by complex rules. Notice that, unlike DEX-Rule, areas in jRule are gradually reduced by increasing only the low bound *lb*. Figure 4 illustrates this process for elementary rules shown in Table 1 and the target value *t* = *unacc*. In this case, *ger* is composed of rules 1, 2, 3, 4, 5 and 9 (in this order) from Table 1. The low bound is *lb* = <high, bad>. jRule makes two iterations, first finding the high bound from rule 9 (*hb* = <low, bad>) and then from rule 4 (*hb* = <high, exc>). In both cases, the areas cover *t* and no reduction is necessary.

5. Performance Analysis

The comparison between the DEX-Rule and jRule algorithms is made with respect to (i) time complexity, (ii) the running time and

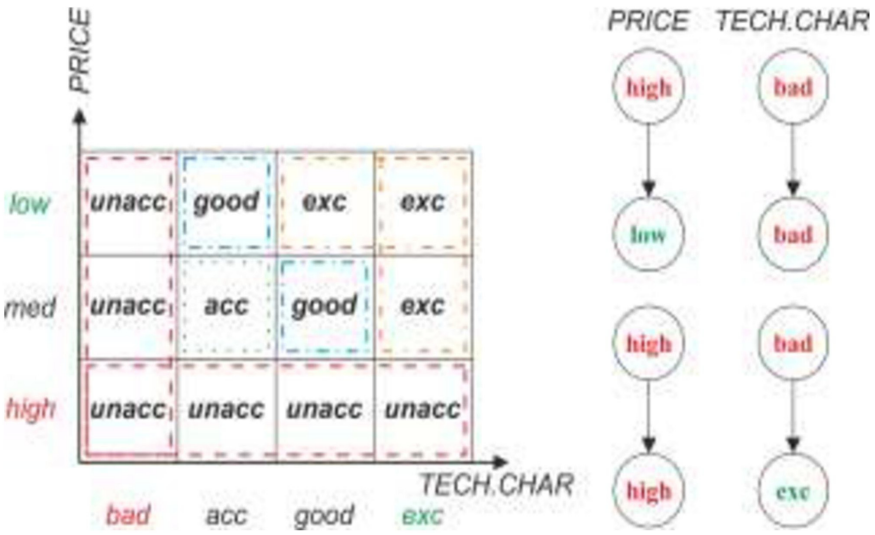

Figure 4. jRule Algorithm identifying the lowest and highest bound for elementary decision rules with decision value t = *unacc*

(iii) the number of complex rules that these two algorithms generate.

Regarding the time complexity, the DEX-Rule algorithm is $O(mn)$ because of its recursive nature, where m is the number of subordinate attributes and n is the number of the elementary decision rules. On the other hand, the time complexity of the jRule algorithm is $O(n^2m)$.

The experimental comparison of the algorithms is based in different DEX models for different aggregated attributes. Both algorithms are implemented in JDEXi (http://kt.ijs.si/MarkoBohanec/jdexi.html) and DEX.NET2 (http://kt.ijs.si/MarkoBohanec/dexinet.html). The algorithms were compiled with the same compiler and run on the same computing environment. Table 4 shows running times of the algorithms on three selected DEX models. Generally, jRule is more efficient, and a major difference occurs with Model 2, which is a large decision table having five subordinate attributes and 1728 elementary decision rules.

#	Running time [s]		# of complex decision rules	
Models	**DEX-Rule**	**jRule**	**DEX-Rule**	**jRule**
1	0.75	0.200	30	18
2	1280.00	0.395	121	64
3	1.94	0.980	11	26

Table 4. Difference between two algorithms based on running time and number of generated complex rules

The two algorithms, in general, produce different complex decision rules. For example, Tables 5 and 6 show the respective complex rules for the *CAR* evaluation model. The rules are very similar, there is only a small difference in rule 7. In some other cases (Table 4), the differences between the algorithms are more pronounced: jRule produces more compact representations for Models 1 and 2, but less compact for Model 3. More research is needed to establish which algorithm is better and under which circumstances.

6. Conclusions

In this work, we proposed a novel algorithm jRule for converting elementary decision rules to complex decision rules in the DEX

#	*PRICE*	*TECH.CHAR*	*CAR*
1	**high**	*	**unacc**
2	*	**bad**	**unacc**
3	medium	acc	acc
4	medium	**good**	good
5	***low***	acc	good
6	>=medium	***exc***	***exc***
7	***low***	>=good	***exc***

Table 5. Complex decision rules generated by DEX-Rule for *CAR* aggregated attribute of CAR Evaluation model

#	*PRICE*	*TECH.CHAR*	*CAR*
1	**high**	*	**unacc**
2	*	**bad**	**unacc**
3	medium	acc	acc
4	medium	**good**	good
5	***low***	acc	good
6	>=medium	***exc***	***exc***
7	***low***	good	***exc***

Table 6. Complex decision rules generated by jRule for *CAR* aggregated attribute of CAR Evaluation model

methodology. In contrast with the current DEX-Rule algorithm, which employs generalization, jRule uses the principle of specialization.

Regarding the time complexity and running time, jRule algorithm perform better than DEX-Rule in all experiments performed for different DEX models. On the other hand, none of the algorithm was clearly better with respect to the number of generated complex decision rules. As part of this work, both algorithms were implemented in two open source libraries, JDEXi V4 and DEXi.NET2.

Acknowledgments

The Public Scholarship, Development, Disability and Maintenance Fund of the Republic of Slovenia (Contract no. 11011-44/2017-14) financially supported this research.

References

[1] Bohanec, M. (2015). DEXi: Program for Multi-Attribute Decision Making User's Manual. *Ljubljana, Slovenia: Institut Jozef Stefan*.

[2] Bohanec, M., Znidarši , M., Rajkovi , V., Bratko, I., Zupan, B. (2013). DEX methodology: three decades of qualitative multi-attribute modeling. *Informatica*, 37 (1).

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

[4] Greco, S., Figueira, J., Ehrgott, M. (2016). Multiple criteria decision analysis. New York: Springer.

[5] Trdin, N., Bohanec, M. (2018). Extending the multicriteria decision making method DEX with numeric attributes, value distributions and relational models. *Central European Journal of Operations Research*, 26 (1) 1-41.