

Extending the multi-criteria decision making method DEX with numeric attributes, value distributions and relational models

Nejc Trdin^{1,2} · Marko Bohanec^{1,2}

Published online: 21 January 2017
© Springer-Verlag Berlin Heidelberg 2017

Abstract DEX is a qualitative multi-criteria decision analysis method. The method supports decision makers in making complex decisions based on multiple, possibly conflicting, attributes. The attributes in DEX have qualitative value scales and are structured hierarchically. The hierarchical topology allows for decomposition of the decision problem into simpler sub-problems. In DEX, alternatives are described with qualitative values, taken from the scales of corresponding input attributes in the hierarchy. The evaluation of alternatives is performed in a bottom-up way, utilizing aggregation functions, which are defined for every aggregated attribute in the form of decision rules. DEX has been used in numerous practical applications—from everyday decision problems to solving decision problems in the financial and ecological domains. Based on experience, we identified the need for three major methodological extensions to DEX: introducing numeric attributes, the probabilistic and fuzzy aggregation of values and relational models. These extensions were proposed by users of the existing method and by the new demands of complex decision problems, which require advanced decision making approaches. In this paper, we introduce these three extensions by describing the extensions formally, justifying their contributions to the decision making process and illustrating them on a didactic example, which is followed throughout the paper.

Keywords Multiple criteria analysis · Qualitative decision models · DEX method · Relational models · Probability and fuzzy distributions

✉ Nejc Trdin
nejc.trdin@ijs.si

¹ “Jožef Stefan” Institute, Ljubljana, Slovenia

² Jožef Stefan International Postgraduate School, Ljubljana, Slovenia

1 Introduction

Decision making is a process in which a decision maker (DM) selects an alternative from among several possible alternatives which best satisfies his/her goals (Bouyssou et al. 2006; Figueira et al. 2005; French 1986). Decision analysis (Clemen and Reilly 2001; Nagel 1993; Skinner 2009) is a discipline that provides a framework for analysing decision problems, which typically involves the development of a model for the evaluation and analysis of alternatives. A common class of models employs methods of multi-criteria decision analysis (MCDA), where alternatives are evaluated using multiple, possibly conflicting, criteria (Bouyssou et al. 2006; Figueira et al. 2005; Ishizaka and Nemery 2013).

A common group of MCDA methods use numerical variables for values and preferences; typical representatives are the Analytic Hierarchy Process (AHP) (Saaty 2008; Saaty and Vargas 2012), French abbreviation for Elimination and Choice Expressing Reality (ELECTRE) (Roy 1991), Preference Ranking Organization METHOD for Enrichment of Evaluations (PROMETHEE) (Brans and Vincke 1985) and French abbreviation for Additive Utilities (UTA) (Jacquet-Lagrèze and Siskos 1982). In contrast, there is a class of qualitative multi-criteria methods which are characterized by using qualitative variables whose value scales contain a finite predefined set of qualitative (or symbolic, “verbal”) values. There are two groups of qualitative MCDA methods which differ in the way knowledge is acquired from the DM while building a decision model (Boose et al. 1993): (1) methods based on an interactive questioning procedure for obtaining the DM’s preference, and (2) methods that acquire the DM’s preferences directly.

Representative methods of the interactive questioning procedure are MACBETH, ZAPROS and ORCLASS. Measuring Attractiveness by Categorical Based Evaluation Technique (MACBETH) (Bana e Costa and Vansnick 1999) uses attractiveness and differential judgments between attributes in order to build preferential relations between alternatives. ZAPROS and ORCLASS are methods belonging to Verbal Decision Analysis (VDA) (Larichev and Moshkovich 1994, 1997; Moshkovich and Mechtov 2013). Russian abbreviation for Closed Procedures near Reference Situations (ZAPROS) (Larichev 2001; Larichev and Moshkovich 1995) provides outranking relationships among alternatives through a verbal decision making approach. ORdinal CLASSification (ORCLASS) (Gomes et al. 2010) assigns alternatives to predefined, ordered classification categories, where alternatives are placed into classes using a set of criteria.

Typical representatives of the second group are the methods DRSA, Doctus and DEX. Dominance-based Rough Set Approach (DRSA) (Greco et al. 2001, 2002) uses rough sets theory with the goal of solving alternative classification and sorting problems represented by decision tables, using the principle of dominance. DRSA has a strong mathematical foundation (Greco et al. 2001) and has evolved in many directions—for example, considering imprecise evaluations and assignments (Dembczyński et al. 2009) and dealing with decisions under uncertainty and time preference (Greco et al. 2010). Doctus (Baracskaï and Dörfler 2003) is a Knowledge-Based Expert System Shell used for evaluation of alternatives and supporting three types of alternative evaluation: Rule-Based Reasoning, Case-Based Reasoning and Case-Based Rule

Reasoning. The third method from this group is DEX, which is addressed in this work.

Decision EXpert (DEX) is a qualitative multi-attribute modelling method which integrates multi-criteria decision modelling with rule-based expert systems (Bohanec et al. 2013). A DEX model consists of hierarchically structured qualitative attributes whose values are words rather than numbers. The aggregation is defined by decision rules. DEX has been widely used in practice to support complex decision processes in health threats and crisis management (Žnidaršič et al. 2009), the use of genetically modified crops (Bohanec et al. 2009; Bohanec 2008; Bohanec and Žnidaršič 2008; Žnidaršič et al. 2008), the evaluation of data mining work flows (Žnidaršič et al. 2012), the evaluation of public administration portals (Leben et al. 2006), the assessment of bank reputational risk (Bohanec et al. 2014), environmental decision making (Kuzmanovski et al. 2015; Žnidaršič et al. 2006b) and many others (Bohanec et al. 2013). DEX is described in Sect. 2.

The many uses of DEX have indicated a great practical value of the method but have also revealed the need to extend it in several directions. Three substantial extensions (Trdin and Bohanec 2012, 2014a) were identified by practical needs, especially with the aim to provide additional features for DMs. The extensions were proposed by the developers, users and other contributors to the method, who are facing new and more demanding decision problems.

The main purpose of this paper is to propose and formally develop the following three extensions to DEX:

1. The inclusion of numeric attributes which allow for combined qualitative and quantitative modelling and eliminate the current need to transform numeric attribute values to qualitative.
2. Support for probabilistic or fuzzy value distributions, allowing one to cope with uncertainty in models and data.
3. Support for relational models and alternatives, allowing for the opportunity to handle relational connections between alternatives and their parts, when alternatives are composed of multiple similar sub-parts.

Each extension is developed formally and illustrated using a didactic example.

The structure of this paper is as follows. Section 2 describes the DEX method and introduces a didactic example, which is used throughout the paper. Sections 3, 4 and 5 present the proposed extensions of DEX: using numerical attributes, probabilistic and fuzzy distributions and relational models, respectively. The contributions and consequences are discussed in Sect. 6. Section 7 concludes the paper.

2 DEX method

DEX (Bohanec 2015a; Bohanec and Rajkovič 1990, 1999; Bohanec et al. 2013; Bohanec and Trdin 2014; Trdin and Bohanec 2012, 2014a) is a qualitative multi-criteria decision modelling method. DEX uses qualitative scales which consist of a finite set of symbolic attribute values, such as “bad”, “medium” and “good”, rather than numeric values. These values are usually, but not necessarily, preferentially ordered.

Consequently, a DEX model generally consists of *attributes*, some of which are preferentially ordered and can be thus referred to as *criteria*.

A DEX model has a form of a *hierarchy*, which represents the decomposition of the decision problem and relations between attributes: higher-level attributes depend on lower-level ones. The terminal nodes of the hierarchy are called *input* or *basic* attributes, whereas all other attributes are called *aggregated* attributes. Additionally, attributes without any parents are called *roots*. A typical model has only one root, which represents the primary outcome of the evaluation of alternatives. There are no conceptual limitations, however, to having multiple roots—for example, to represent evaluations from different viewpoints. In MCDA, hierarchies are commonly used in methods such as AHP (Saaty 2008) and Multiple Criteria Hierarchy Process (MCHP) (Corrente et al. 2012).

The aggregation of values in the model is facilitated by *decision rules*. To compute an aggregated attribute's value from the values of its children, each aggregated attribute has an associated total *aggregation function*. The function is defined by a *decision table*—interpretable as a set of decision rules. The total aggregation function needs to specify an output value for each combination of children's values. Such tables are typically prepared by the DM according to his/her preferences. Another method in MCDA that relies on decision tables is DRSA (Greco et al. 2001). There, a decision table is understood mainly as a collection of decision alternatives rather than a constructed specification of an aggregation function. In contrast to DEX, decision tables in DRSA may contain both qualitative and numeric criteria.

The *evaluation of alternatives* is done in a *bottom-up* way. The input attribute values for some alternative are acquired directly from the alternative. The evaluation is carried out progressively from the lowest attributes in the hierarchy to its roots. Each aggregate attribute's value is computed using the corresponding aggregation function.

Currently, DEX method is implemented in the software called DEXi (Bohanec 2014, 2015b). DEXi supports an interactive construction of the decision model and alternatives. The software aids in defining decision rules and checking their completeness and consistency and provides a number of analytical decision tools. The user can make an in-depth analysis of evaluations and decision alternatives. For example, each computed value has an associated decision rule, which was used to obtain that value; the rule explains *how* the evaluation was obtained. Moreover, the decision rule was applied on the basis of values computed by previous evaluations, which can also be drilled down even further, explaining the sources for such evaluation and thus providing evidence of *why* the evaluation was such.

DEXi software includes four different analysis procedures for the evaluated alternatives (Bohanec 2014): (1) “Plus-minus-1 analysis” checks to which extents alternative evaluations are affected by small changes to the input attribute values; (2) “Selective explanation” informs the user about the strong and weak components of each alternative; (3) “Compare” compares the pre-selected alternatives attribute-wise, and (4) “Charts” are able to plot k sided utility diagrams based on the selected alternatives and k selected attributes.

2.1 DEX model

Formally, a DEX model M is a four-tuple $M = (X, D, S, F)$, where X is the set of attributes, S is the descendant function that determines the hierarchical structure of M , D is the set of value scales (domains) of attributes in X and F is the set of aggregation functions.

The set X consists of n attributes:

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

In practice, attributes are usually given a name, which uniquely identifies the attribute—for instance “price”, “quality”, “location”, etc. In the didactic example, introduced later in Sect. 2.2, we will often denote an attribute by its name (e.g. *location*) and use a named subscript to denote related components (e.g. $D_{location}$).

Each attribute $x_i \in X$ has a corresponding value scale $D_i \in D$, which is an ordered set of symbolic (qualitative) values:

$$D_i = \{w_{i_1}, w_{i_2}, \dots, w_{i_{m_i}}\}, l \neq k \Leftrightarrow w_{i_l} \neq w_{i_k}. \tag{2}$$

Here, m_i is the number of values in the scale of attribute x_i . In practice, attribute values are also represented by words, such as “low”, “good”, “acceptable” and “yes”.

With respect to the DM’s preferences, scales can be either ordered or unordered. If the scale is totally ordered, then the DM defines the preferential operator \succsim . For each $j \leq k, j, k \in [1, m_i]$, we write $w_{i_j} \succsim w_{i_k}$ and interpret that w_{i_k} is preferentially at least as good as w_{i_j} . For ordered scales, we use the notation:

$$D_i = (w_{i_1}, w_{i_2}, \dots, w_{i_{m_i}}). \tag{3}$$

According to the usual convention (Greco et al. 2001), attributes with preferentially ordered scales are called criteria.

Attributes are structured hierarchically: each attribute x_i may have some descendants (children) and/or predecessors (parents) in the model. This relationship is described by the function $S : X \rightarrow 2^X$, which maps each attribute $x_i \in X$ to a set of its descendants $S(x_i)$:

$$S(x_i) = \{x_{i_1}, x_{i_2}, \dots, x_{i_k}\}, l \neq m \Leftrightarrow x_{i_l} \neq x_{i_m}, \\ x_{i_j} \neq x_i, j = 1, \dots, k, \\ x_{i_1}, x_{i_2}, \dots, x_{i_k} \in X. \tag{4}$$

The relations induced by S must represent a hierarchy—that is, a connected and directed acyclic graph. Such a graph contains finitely many edges and vertices, where each edge is directed from one vertex to another. Further, starting in arbitrary vertex x and traversing the graph through all possible paths, one will never reach vertex x again. In most practical cases the graph induced by S represents a tree—all attributes

except one root attribute, have exactly one parent. In general, however, a hierarchy may have several roots and may contain attributes that influence more than one parent attribute. If S does not form a hierarchy, then the directed graph contains one or more cycles, and the evaluation of alternatives is generally not possible—for that reason we restrict S to acyclic graphs.

The model input attributes $modellInputs_M$ constitute a set of all attributes that do not have any children in the hierarchy. Additionally, the model input space $inputSpace_M$ is defined as the Cartesian product of attributes' scales in $modellInputs_M$:

$$\begin{aligned} modellInputs_M &= \{x_i \in X \mid S(x_i) = \emptyset\}, \\ inputSpace_M &= D_j \times D_l \times \dots \times D_k, x_j, x_l, \dots, x_k \in modellInputs_M. \end{aligned} \quad (5)$$

Model output attributes and model output space are defined similarly— $modelOutputs_M$ as a set of all attributes in model M that are not children of any other attribute; $outputSpace_M$ as the Cartesian product of these attributes' scales:

$$\begin{aligned} modelOutputs_M &= \left\{ x_i \in X \mid x_i \notin \bigcup_{x_j \in X} S(x_j) \right\}, \\ outputSpace_M &= D_j \times D_k \dots \times D_l, x_j, x_l, \dots, x_k \in modelOutputs_M. \end{aligned} \quad (6)$$

Another useful set is the set of all aggregated attributes, $aggAttributes_M$. These attributes are the ones whose values have to be computed when evaluating alternatives. Structurally, they appear as internal nodes in the hierarchy:

$$aggAttributes_M = \{x_i \in X \mid x_i \notin modellInputs_M\}. \quad (7)$$

Each aggregated attribute $x_i \in aggAttributes_M$ needs an aggregation function, so that the decision alternatives can be evaluated. We denote $f_i \in F$ as the aggregation function corresponding to attribute x_i . Each aggregation function f_i is a total function that maps all scale value combinations of x_i 's inputs to an interval value in scale D_i :

$$f_i : D_{i_1} \times D_{i_2} \times \dots \times D_{i_k} \rightarrow I(D_i), \{x_{i_1}, x_{i_2}, \dots, x_{i_k}\} = S(x_i). \quad (8)$$

Here, $I(D_i)$ represents the space of all possible intervals $[v_l, v_h] = \{v_j \mid v_l \lesssim v_j \lesssim v_h \wedge v_l, v_j, v_h \in D_i\}$.

Such aggregation functions are usually represented as tables which assign an output value to each possible input value combination. Such tables can also be interpreted as if-then rules. Examples of such tables are given in Sect. 2.2.

For the purpose of using functions f_i in the evaluation of alternatives, especially in the extended DEX method described later in this paper, we also define functions F_i , which are generalizations of f_i and are defined on sets of values rather than on crisp values. The F_i function definition corresponding to the associated function f_i is:

$$F_i : 2^{D_{i_1}} \times 2^{D_{i_2}} \times \dots \times 2^{D_{i_k}} \rightarrow 2^{D_i}. \quad (9)$$

Functions F_i are expected to produce the same results on the same singleton input values as their f_i counterparts:

$$\begin{aligned} \forall v_{i_1} \in D_{i_1}, v_{i_2} \in D_{i_2}, \dots, v_{i_k} \in D_{i_k} : W_{i_1} = \{v_{i_1}\}, \\ W_{i_2} = \{v_{i_2}\}, \dots, W_{i_k} = \{v_{i_k}\} \Rightarrow \\ F_i(W_{i_1}, W_{i_2}, \dots, W_{i_k}) = \{f_i(v_{i_1}, v_{i_2}, \dots, v_{i_k})\}. \end{aligned} \tag{10}$$

Given f_i , the value for function F_i is computed as:

$$F_i(W_{i_1}, W_{i_2}, \dots, W_{i_k}) = \bigcup_{(v_{i_1}, v_{i_2}, \dots, v_{i_k}) \in W_{i_1} \times W_{i_2} \times \dots \times W_{i_k}} \{f_i(v_{i_1}, v_{i_2}, \dots, v_{i_k})\}. \tag{11}$$

With A_M , we denote the set of all decision alternatives for some model M :

$$A_M = \{a_1, a_2, \dots, a_m\}. \tag{12}$$

Here, each alternative is defined in the Cartesian space $inputSpace_M$, so that $a_i \in inputSpace_M$.

Alternatives in A_M are evaluated with model M with function $evaluation_M$:

$$evaluation_M : inputSpace_M \rightarrow outputSpace_M. \tag{13}$$

The evaluation of alternative a_i on model M is done by computing the function $evaluation_M(a_i)$ and assigning output values to all attributes from $outputSpace_M$. The evaluation is done as a bottom-up aggregation of model inputs toward its outputs according to the hierarchical structure of the model. Algorithmically, all aggregated attributes in model M are first topologically sorted with respect to S . The sorting determines the order of aggregation function evaluations and ensures that all inputs to the current aggregation function are readily available. The values of aggregated attributes are computed using the corresponding functions F_i in the given order. This produces the output evaluations in $modelOutputs_M$.

2.2 Didactic example

We will follow the same didactic example through the whole paper: choosing an apartment to buy. At the first stage, the model is developed using the DEX method without any extensions. Later, we will gradually develop the example using the DEX extensions proposed in this paper.

The model has one root attribute, *apartment*, and three children: *price*, *location* and *layout*. These three attributes are split into finer detail, which can be seen in Fig. 1. For example, *price* depends on the *buying price* and the price of *utilities*.

All attributes have qualitative values assigned to their value scales, which are also shown in Fig. 1 beneath the attributes' names. Notably, the model has a simple tree structure, in which none of the attributes influences more than one parent attribute. In total, there are 18 attributes: 11 input and 7 aggregated, including one root attribute.

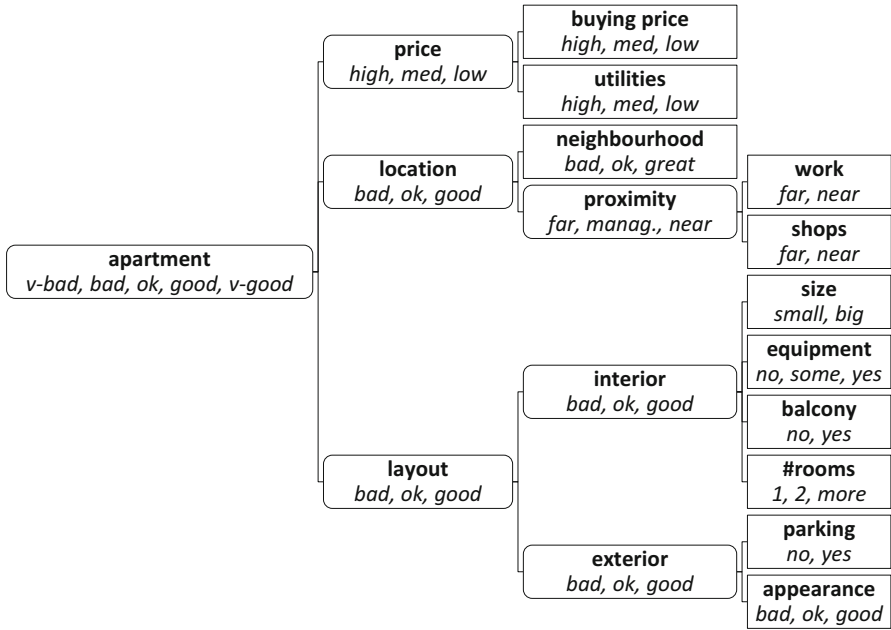


Fig. 1 Model for evaluating an apartment. The tree structure of attributes is presented. A name and value scale is shown for each attribute. The type of the attribute is indicated by the box shape: input attributes (rectangular) and aggregate attributes (round corners)

Formally, the model has attributes:

$$X = \{apartment, price, location, layout, buying price, utilities, neighbourhood, proximity, interior, exterior, work, shops, size, equipment, balcony, \#rooms, parking, appearance\}. \tag{14}$$

The scales of the attributes are defined in Fig. 1. All the scales are ordered preferentially, for instance:

$$D_{apartment} = (v\text{-bad}, bad, ok, good, v\text{-good}). \tag{15}$$

The function S can be read directly from the model structure presented in Fig. 1:

$$\begin{aligned} S(apartment) &= \{price, location, layout\}, \\ &\dots \\ S(exterior) &= \{parking, appearance\}, \\ S(buying price) &= \dots = S(appearance) = \emptyset. \end{aligned} \tag{16}$$

With all of these in place, we can now define $modelInputs_M$, $inputSpace_M$, $modelOutputs_M$, $outputSpace_M$, and $aggAttributes_M$, where M is the Apartment model. These can be directly constructed from Fig. 1:

Table 1 Aggregation function for final *apartment* assessment, based on its *price*, *location* and *layout*

<i>price</i>	<i>location</i>	<i>layout</i>	<i>apartment</i>
<i>high</i>	<i>bad</i>	<i>bad</i>	<i>v-bad</i>
<i>high</i>	<i>bad</i>	<i>ok</i>	<i>v-bad</i>
<i>high</i>	<i>bad</i>	<i>good</i>	<i>v-bad</i>
<i>high</i>	<i>ok</i>	<i>bad</i>	<i>v-bad</i>
<i>high</i>	<i>ok</i>	<i>ok</i>	<i>v-bad</i>
<i>high</i>	<i>ok</i>	<i>good</i>	<i>v-bad</i>
<i>high</i>	<i>good</i>	<i>bad</i>	<i>v-bad</i>
<i>high</i>	<i>good</i>	<i>ok</i>	<i>v-bad</i>
<i>high</i>	<i>good</i>	<i>good</i>	<i>bad</i>
<i>med</i>	<i>bad</i>	<i>bad</i>	<i>v-bad</i>
<i>med</i>	<i>bad</i>	<i>ok</i>	<i>v-bad</i>
<i>med</i>	<i>bad</i>	<i>good</i>	<i>v-bad</i>
<i>med</i>	<i>ok</i>	<i>bad</i>	<i>bad</i>
<i>med</i>	<i>ok</i>	<i>ok</i>	<i>bad</i>
<i>med</i>	<i>ok</i>	<i>good</i>	<i>bad</i>
<i>med</i>	<i>good</i>	<i>bad</i>	<i>bad</i>
<i>med</i>	<i>good</i>	<i>ok</i>	<i>ok</i>
<i>med</i>	<i>good</i>	<i>good</i>	<i>ok</i>
<i>low</i>	<i>bad</i>	<i>bad</i>	<i>bad</i>
<i>low</i>	<i>bad</i>	<i>ok</i>	<i>bad</i>
<i>low</i>	<i>bad</i>	<i>good</i>	<i>ok</i>
<i>low</i>	<i>ok</i>	<i>bad</i>	<i>good</i>
<i>low</i>	<i>ok</i>	<i>ok</i>	<i>good</i>
<i>low</i>	<i>ok</i>	<i>good</i>	<i>good</i>
<i>low</i>	<i>good</i>	<i>bad</i>	<i>good</i>
<i>low</i>	<i>good</i>	<i>ok</i>	<i>v-good</i>
<i>low</i>	<i>good</i>	<i>good</i>	<i>v-good</i>

$$\begin{aligned}
 \text{modelInputs} &= \{\text{buyingprice, utilities, neighbourhood, work,} \\
 &\quad \text{shops, size, equipment, balcony, \#rooms, parking, appearance}\}, \\
 \text{inputSpace} &= (\text{high, med, low}) \times (\text{high, med, low}) \times \dots \times (\text{bad, ok, good}), \\
 \text{modelOutputs} &= \{\text{apartment}\}, \\
 \text{outputSpace} &= (\text{v-bad, bad, ok, good, v-good}), \\
 \text{aggAttributes} &= \{\text{apartment, price, location, layout, proximity, interior, exterior}\}. \quad (17)
 \end{aligned}$$

Finally, to complete the model, we need formal definitions of aggregation functions for the aggregated attributes. The aggregation functions for attributes in *aggAttributes* are described by the sets of decision rules in Tables 1, 2, 3, 4, 5, 6 and 7. Even though generally DEX aggregation functions map their arguments to interval values over

Table 2 Aggregation function for *price*, dependent *buying price* and monthly *utilities*

<i>buying price</i>	<i>utilities</i>	<i>price</i>
<i>high</i>	<i>high</i>	<i>high</i>
<i>high</i>	<i>med</i>	<i>high</i>
<i>high</i>	<i>low</i>	<i>med</i>
<i>med</i>	<i>high</i>	<i>high</i>
<i>med</i>	<i>med</i>	<i>med</i>
<i>med</i>	<i>low</i>	<i>med</i>
<i>low</i>	<i>high</i>	<i>med</i>
<i>low</i>	<i>med</i>	<i>low</i>
<i>low</i>	<i>low</i>	<i>low</i>

Table 3 Aggregation function for *location*, dependent on *neighbourhood* and *proximity*

<i>neighbourhood</i>	<i>proximity</i>	<i>location</i>
<i>bad</i>	<i>far</i>	<i>bad</i>
<i>bad</i>	<i>manag.</i>	<i>bad</i>
<i>bad</i>	<i>near</i>	<i>ok</i>
<i>ok</i>	<i>far</i>	<i>bad</i>
<i>ok</i>	<i>manag.</i>	<i>ok</i>
<i>ok</i>	<i>near</i>	<i>good</i>
<i>great</i>	<i>far</i>	<i>ok</i>
<i>great</i>	<i>manag.</i>	<i>good</i>
<i>great</i>	<i>near</i>	<i>good</i>

Table 4 Aggregation function for *layout*, based on *interior* and *exterior*

<i>interior</i>	<i>exterior</i>	<i>layout</i>
<i>bad</i>	<i>bad</i>	<i>bad</i>
<i>bad</i>	<i>ok</i>	<i>bad</i>
<i>bad</i>	<i>good</i>	<i>ok</i>
<i>ok</i>	<i>bad</i>	<i>bad</i>
<i>ok</i>	<i>ok</i>	<i>ok</i>
<i>ok</i>	<i>good</i>	<i>ok</i>
<i>good</i>	<i>bad</i>	<i>ok</i>
<i>good</i>	<i>ok</i>	<i>good</i>
<i>good</i>	<i>good</i>	<i>good</i>

Table 5 Aggregation function for *proximity*, dependent on proximity to *work* and *shops*

<i>work</i>	<i>shops</i>	<i>proximity</i>
<i>far</i>	<i>far</i>	<i>far</i>
<i>far</i>	<i>near</i>	<i>manag.</i>
<i>near</i>	<i>far</i>	<i>manag.</i>
<i>near</i>	<i>near</i>	<i>near</i>

Table 6 Aggregation function for *interior*, based on *size*, *equipment*, *balcony* presence and number of *rooms*

<i>size</i>	<i>equipment</i>	<i>balcony</i>	<i>#rooms</i>	<i>interior</i>
<i>small</i>	<i>no</i>	<i>no</i>	1	<i>bad</i>
<i>small</i>	<i>no</i>	<i>no</i>	2	<i>bad</i>
<i>small</i>	<i>no</i>	<i>no</i>	<i>more</i>	<i>bad</i>
<i>small</i>	<i>no</i>	<i>yes</i>	1	<i>bad</i>
<i>small</i>	<i>no</i>	<i>yes</i>	2	<i>bad</i>
<i>small</i>	<i>no</i>	<i>yes</i>	<i>more</i>	<i>bad</i>
<i>small</i>	<i>some</i>	<i>no</i>	1	<i>bad</i>
<i>small</i>	<i>some</i>	<i>no</i>	2	<i>bad</i>
<i>small</i>	<i>some</i>	<i>no</i>	<i>more</i>	<i>bad</i>
<i>small</i>	<i>some</i>	<i>yes</i>	1	<i>bad</i>
<i>small</i>	<i>some</i>	<i>yes</i>	2	<i>ok</i>
<i>small</i>	<i>some</i>	<i>yes</i>	<i>more</i>	<i>ok</i>
<i>small</i>	<i>yes</i>	<i>no</i>	1	<i>bad</i>
<i>small</i>	<i>yes</i>	<i>no</i>	2	<i>ok</i>
<i>small</i>	<i>yes</i>	<i>no</i>	<i>more</i>	<i>ok</i>
<i>small</i>	<i>yes</i>	<i>yes</i>	1	<i>ok</i>
<i>small</i>	<i>yes</i>	<i>yes</i>	2	<i>ok</i>
<i>small</i>	<i>yes</i>	<i>yes</i>	<i>more</i>	<i>ok</i>
<i>big</i>	<i>no</i>	<i>no</i>	1	<i>bad</i>
<i>big</i>	<i>no</i>	<i>no</i>	2	<i>bad</i>
<i>big</i>	<i>no</i>	<i>no</i>	<i>more</i>	<i>bad</i>
<i>big</i>	<i>no</i>	<i>yes</i>	1	<i>bad</i>
<i>big</i>	<i>no</i>	<i>yes</i>	2	<i>bad</i>
<i>big</i>	<i>no</i>	<i>yes</i>	<i>more</i>	<i>ok</i>
<i>big</i>	<i>some</i>	<i>no</i>	1	<i>bad</i>
<i>big</i>	<i>some</i>	<i>no</i>	2	<i>bad</i>
<i>big</i>	<i>some</i>	<i>no</i>	<i>more</i>	<i>ok</i>
<i>big</i>	<i>some</i>	<i>yes</i>	1	<i>ok</i>
<i>big</i>	<i>some</i>	<i>yes</i>	2	<i>good</i>
<i>big</i>	<i>some</i>	<i>yes</i>	<i>more</i>	<i>good</i>
<i>big</i>	<i>yes</i>	<i>no</i>	1	<i>ok</i>
<i>big</i>	<i>yes</i>	<i>no</i>	2	<i>ok</i>
<i>big</i>	<i>yes</i>	<i>no</i>	<i>more</i>	<i>ok</i>
<i>big</i>	<i>yes</i>	<i>yes</i>	1	<i>ok</i>
<i>big</i>	<i>yes</i>	<i>yes</i>	2	<i>good</i>
<i>big</i>	<i>yes</i>	<i>yes</i>	<i>more</i>	<i>good</i>

Table 7 Aggregation function for *exterior*, dependent on *parking* space availability and *appearance*

<i>parking</i>	<i>appearance</i>	<i>exterior</i>
<i>no</i>	<i>bad</i>	<i>bad</i>
<i>no</i>	<i>ok</i>	<i>bad</i>
<i>no</i>	<i>good</i>	<i>bad</i>
<i>yes</i>	<i>bad</i>	<i>ok</i>
<i>yes</i>	<i>ok</i>	<i>good</i>
<i>yes</i>	<i>good</i>	<i>good</i>

$x_i \in X$ (Eq. (8)), only single values are used in this case. This is done by convention if the lower and upper bounds of an interval are equal to some word w_m ; then, instead of writing $[w_m, w_m]$, we write just w_m . For example, a decision rule from Table 6 is:

$$f_{apartment}(\text{high,ok,ok}) = \text{v-bad}. \quad (18)$$

It can be interpreted as: “if *price* is *high* and *location* is *ok* and *layout* is *ok*, then *apartment* is *v-bad*”.

Now that the model is complete, alternatives can be defined and evaluated. Let us consider three different apartments, each described by a n -tuple of values corresponding *modelInputs_M*. The three apartments are shown in Table 8. Notice that all input values are formulated in terms of qualitative values. In reality, we determine these values from real values of alternatives. For example, we may assess the buying price of 50.000 € as *med*. We assume that this process has already been carried out by the DM, as shown in Table 8.

To illustrate the evaluation procedure, let us consider the alternative “Big”. A possible topological sorting of the aggregated attributes is:

$$(\text{price, proximity, location, interior, exterior, layout, apartment}). \quad (19)$$

Consequently, the attributes should be evaluated according to this order—that is, from *price* to *apartment*. The value of *price* is computed by the aggregation function f_{price} , given the values for child attributes *buying price* and *utilities*. The functional value for $f_{price}(\text{med,med})$ is *med*, which is given in Table 2 in the fifth row. Similarly, the value for *proximity* is computed by computing function $f_{proximity}$ with values (*far, far*)—produced value is *far*. The values for the remaining aggregated attributes are produced in the same way, as shown in Table 9. The apartment “Big” is assessed as *very bad*. The other two apartments, “Equipped” and “Nice”, are assessed as *bad* and *good*, respectively.

3 Numeric attributes

Most MCDM methods are *quantitative*—they involve numeric attributes and real-valued utility functions. DEX is a *qualitative* method and currently uses only qualitative attributes. This approach is not always appropriate because it requires

Table 8 The table describes three different available apartments for evaluation

Alternative	buying price	utilities	neighbourhood	work shops	size	equipment	balcony	#rooms	parking	appearance	
Big	med	med	bad	far	far	big	no	no	more	yes	bad
Equipped	med	low	ok	near	far	small	yes	yes	2	yes	ok
Nice	low	med	ok	near	near	small	some	no	1	no	ok

Alternative names, input attributes and corresponding values are given

Table 9 Evaluations of three alternatives according to the developed model

Alternative	price	proximity	location	interior	exterior	layout	apartment
Big	med	far	bad	bad	ok	bad	v-bad
Equipped	med	manag.	ok	ok	good	ok	bad
Nice	low	near	good	bad	bad	bad	good

Values of aggregated attributes and the final evaluation are presented

that quantitative values are discretized before use, even for quantities that are naturally measured and expressed with numbers. Furthermore, in many decision problems there is a need to support both qualitative and quantitative attributes within the same model (Bohanec et al. 2014; Kontić et al. 2014). Even though a few quantitative features have already been considered within DEX (Mileva-Boshkoska and Bohanec 2012; Trdin and Bohanec 2012, 2014a, b; Žnidaršič et al. 2003), there is a strong need to systematically introduce numerical attributes into the DEX method.

Introducing numeric attributes in DEX is thus aimed at extending the expressive power of the method, so that both discrete and numeric attributes can be used and combined in a single model. We wish to do this in a general and flexible way, making sure that numerical attributes integrate well into the existing framework. The goal is not to introduce any specific quantitative MCDA method in DEX, but to provide a flexible schema which allows using different quantitative methods for eliciting numeric attributes, their weights and utility functions.

Adding numerical attributes requires a number of representational and algorithmic extensions, such as adding numerical aggregation functions and handling transformations between qualitative and numeric values. Attribute value scales have to be extended to include *real* numbers, *integer* numbers, *bounded intervals* over real numbers and *bounded integer intervals*, as expressed later in Eq. (20).

Supporting numeric attributes does not affect the principle of structuring attributes in a hierarchy and assigning aggregation functions to aggregated attributes. However, there is an essential difference in the aggregation functions themselves because, in general, they have to cope with various combinations of qualitative and quantitative attributes, both at function inputs (arguments) and function outcomes. For this reason, we consider six different function types which differ by value type. There are three combinations of basic types the function can receive: (1) all qualitative values (denoted *QQ*), (2) all quantitative (numeric) values (*NN*) or (3) a mix of quantitative

and qualitative values (NQ). Regarding the output, the function can yield only one type of output—either qualitative (Q) or quantitative (N). Note that the basic DEX method covers only one of these cases: functions that map from qualitative arguments to qualitative values (denoted $QQ \rightarrow Q$).

The introduction of quantitative attribute values increases the expressiveness of attributes' value scales and the aggregation of such quantitative values. The extension naturally incorporates numeric values and operations. Consequently, it provides additional flexibility to the DM and decision analyst, and allows addressing a wider class of decision problems. Inevitably, the drawback of the extension is an increased complexity due to new types of aggregation functions.

3.1 Extension formalization

First, we need to extend the definition of the value scale D_i for some x_i given in Eqs. (2, 3) to include numeric quantities:

$$D_i = \begin{cases} (w_{i_1}, w_{i_2}, \dots, w_{i_{m_i}}), & \text{or} \\ [a..b], a, b \in \mathbb{Z} \cup \{-\infty, \infty\}, & \text{or} \\ [c, d], c, d \in \mathbb{R} \cup \{-\infty, \infty\}. & \end{cases} \quad (20)$$

This equation represents three possibilities for defining the value scale D_i :

1. The first option remains the same as before: a finite list of *words*. The list may be preferentially ordered.
2. The second option defines an *integer interval* by specifying two integer values $a, b, a \leq b$. The scale of the particular attribute is then composed of all integers between a and b , inclusively. Note that we can also specify an unbounded interval using negative or positive infinities.
3. The third option defines a *real interval* by choosing two real values $c, d, c \leq d$. The possible values of such attribute are all real values between c and d , inclusively. Both infinities are also allowed.

These three cases introduce five types of *primitive aggregation functions* (Fig. 2). Here, “primitive” means that the function can be defined in “one step”, using either a single formula or a single decision table. In contrast, a “non-primitive” or “compound” aggregation is defined by a hierarchical composition of primitive functions.

Function type 1 ($QQ \rightarrow Q$): These functions have only qualitative inputs and a qualitative output. They are already covered by the current DEX method and modelled by decision rules according to Eq. (8).

Function type 2 ($QQ \rightarrow N$): These functions have only qualitative inputs and a numeric output. They are modelled similarly to Function type 1 by decision rules; the only difference is that they output a single number or numeric interval instead of a single word or interval of words. The output number is restricted to the scale of the aggregated attribute. Function type 2 is already formally covered by Eq. (8), taking the extended definition of D_i in Eq. (20). Note that this function type includes the

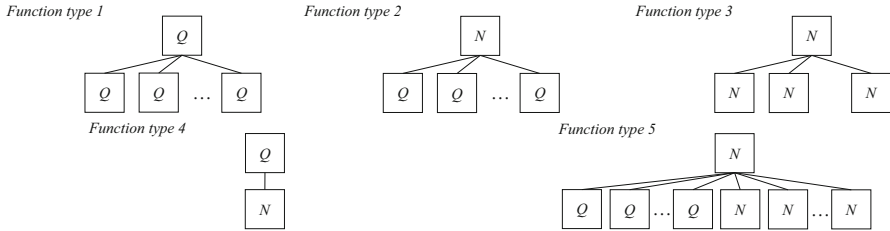


Fig. 2 The five possible cases for configurations of qualitative and numeric primitive aggregation functions. The function type is dependent upon its input types and its output type. The letters denote qualitative (\$Q\$) attribute type or numeric (\$N\$) attribute type

special case \$Q \to N\$, where only one qualitative attribute is converted to a numerical attribute.

Function type 3 (\$NN \to N\$): These functions have only numeric inputs and a numeric output. They are modelled in a purely mathematical way using mathematical operators on numeric functions' arguments, such as \$+\$, \$-\$, \$\times\$, \$/\$, *power*, *square*, *root*, etc., and numeric constants. A common representative of this function type is the weighted average—for example, \$\sum_{i=1}^n w_i v_i\$—where \$w_i\$ is the \$i\$-th attribute's weight and \$v_i\$ is its value. Similar to Function type 2, this function type is already formally covered by Eq. (8). The difference is that all \$D_{i_1}, D_{i_2}, \dots, D_{i_k}, D_i\$ are numeric scales, and function \$f_i\$ is a mathematical expression. Functions of this type are commonly used in quantitative MCDA methods and can thus be easily adopted in DEX, together with already developed support for their acquisition and use in practice. For instance, Function type \$NN \to N\$, can use the weighted sum and employ AHP's (Saaty 2008) pairwise comparison to elicit attribute weights.

Function type 4 (\$N \to Q\$): These functions discretize a single numeric input attribute to an output qualitative attribute. Generally, a discretization function of the \$i\$-th attribute (\$NQ_i\$) maps real numbers to qualitative value scales:

$$NQ_i : A \subseteq \mathbb{R} \to D_i, \tag{21}$$

so that:

$$NQ_i (y) = \begin{cases} w_{i_p} | y \in [a, b] \\ \dots \\ w_{i_r} | y \in [c, d] \end{cases}, w_{i_p}, \dots, w_{i_r} \in D_i. \tag{22}$$

Function type 5 (\$NQ \to N\$): These functions have qualitative and numeric inputs and a numeric output. The primitive specification of such functions is based on decision tables, similar to function types 1 and 2. First, we define a decision table using all qualitative function arguments. Second, in each row of the table a numerical function of the remaining numeric attributes is defined. Formally, suppose that \$q_1, q_2, \dots, q_k\$ are indices of all qualitative inputs to \$x_i\$ and \$n_1, n_2, \dots, n_m\$ all numeric ones. Let \$\Phi(x_i)\$ denote the set of functions which receive all \$x_i\$'s numeric inputs and output a numeric value in \$D_i\$.

$$\Phi(x_i) = \{f | f : D_{i_{n_1}} \times D_{i_{n_2}} \times \dots \times D_{i_{n_m}} \to D_i\} \tag{23}$$

Let h_i be a function which maps from all combinations of qualitative inputs to space $\Phi(x_i)$:

$$h_i : D_{i_{q_1}} \times D_{i_{q_2}} \times \dots \times D_{i_{q_k}} \rightarrow \Phi(x_i). \quad (24)$$

The overall aggregation function f_i is then evaluated in two steps: (1) the value of $h_i(v_{q_1}, v_{q_2}, \dots, v_{q_k})$ is used as a table lookup index to find the corresponding function $g \in \Phi(x_i)$, and (2) evaluate $g(v_{n_1}, v_{n_2}, \dots, v_{n_m})$. Formally:

$$f_i(v_{q_1}, v_{q_2}, \dots, v_{q_k}, v_{n_1}, v_{n_2}, \dots, v_{n_m}) = h_i(v_{q_1}, v_{q_2}, \dots, v_{q_k})(v_{n_1}, v_{n_2}, \dots, v_{n_m}). \quad (25)$$

Using these five primitive function types, we can express other aggregations of numeric and qualitative values. Among these, two are of special interest: $NN \rightarrow Q$ and $NQ \rightarrow Q$.

Compound function type $NN \rightarrow Q$: These compound functions have only numeric inputs and a qualitative output. They can be modelled in two ways:

1. Each numeric attribute can be first discretized and then aggregated by *Function type 1*: $(N \rightarrow Q)(N \rightarrow Q) \rightarrow Q$;
2. A mathematical expression can be constructed by *Function type 3*, and then a single discretization can be applied on the output: $NN \rightarrow N \rightarrow Q$.

Compound function type $NQ \rightarrow Q$: This function type has qualitative and numeric inputs and a numeric output. It can be modelled with primitive function types in three ways:

1. Discretizing the result retrieved from a *Function type 5* function: $(NQ \rightarrow N) \rightarrow Q$;
2. Discretizing numeric inputs and creating a qualitative function on all qualitative attributes: $(N \rightarrow Q)Q \rightarrow Q$;
3. Using *Function type 3* on numeric attributes, discretizing its output and creating a qualitative function on the result and qualitative attributes: $((NN \rightarrow N) \rightarrow Q)Q \rightarrow Q$.

The two compound function types are presented graphically according to their descriptions in Figs. 3 and 4.

The compound function types $NN \rightarrow Q$ and $NQ \rightarrow Q$ can thus be modelled in a number of different ways. On one hand, this is convenient, as the functions can be flexibly adapted to the characteristics of the problem. On the other hand, this puts additional burden on DM, as he/she has to decide which option to use in a given situation. This decision is important, as different compound functions generally lead to different results. The proposed approach fulfils the requirement for flexibility, but does not answer the question on how to decide which option should be used. This is left for further research and practice. At the time being, we assume that the choice of compound functions is entirely at the DM's responsibility.

3.2 Didactic example

We shall introduce numerical attributes in the didactic example while keeping its hierarchical structure intact. First, we replace the attributes *price*, *buying price*, *utilities*,

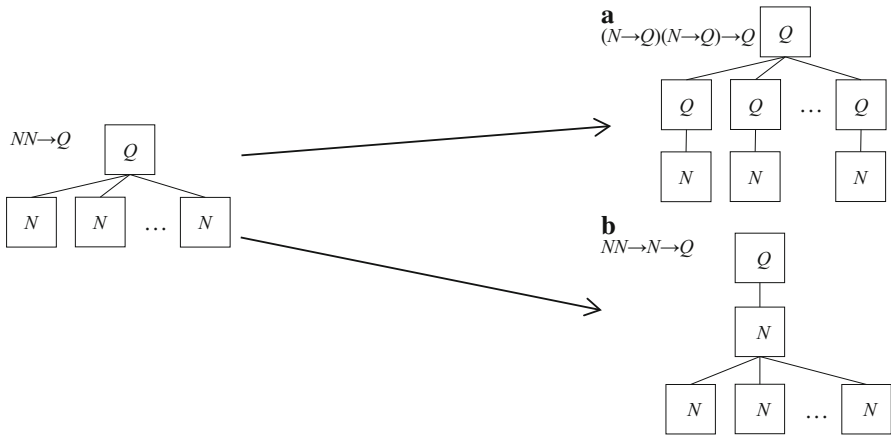


Fig. 3 Two possibilities for modelling Compound function type $NN \rightarrow Q$. Option *a* suggests that numeric attributes are first discretized to qualitative and then the function is modelled as a rule based qualitative function. Option *b* indicates that numeric attributes are aggregated using Function type 3, and then discretizing the produced result

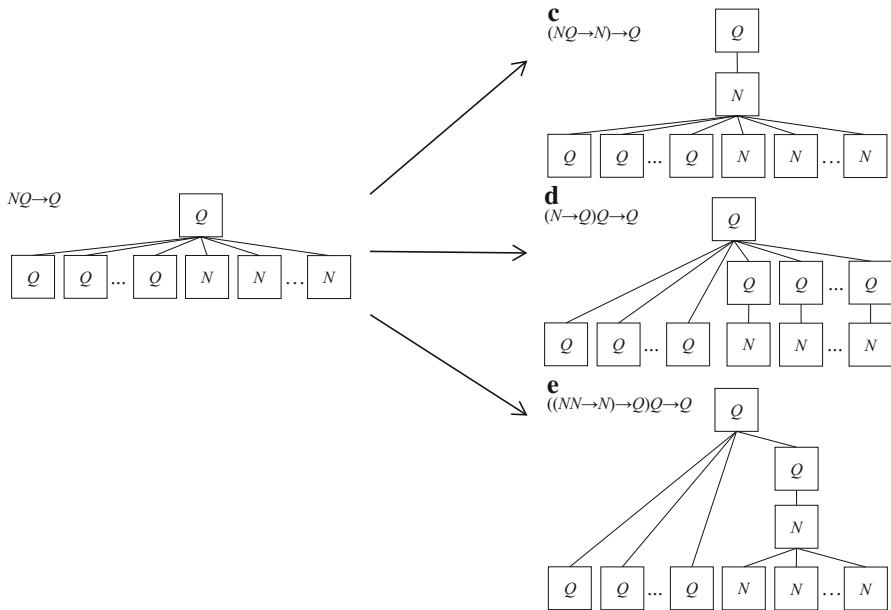


Fig. 4 Three possibilities for modelling Compound function type $NQ \rightarrow Q$. Option *c* gives the configuration to use Function type 5 on the inputs and then discretize the computed output. Option *d* gives the possibility to discretize the numeric inputs and creates a qualitative function on outputs and remaining qualitative inputs. Option *e* presents the possibility to aggregate numeric attributes to single numeric value, discretize the value and then construct a qualitative function based on this and on qualitative inputs

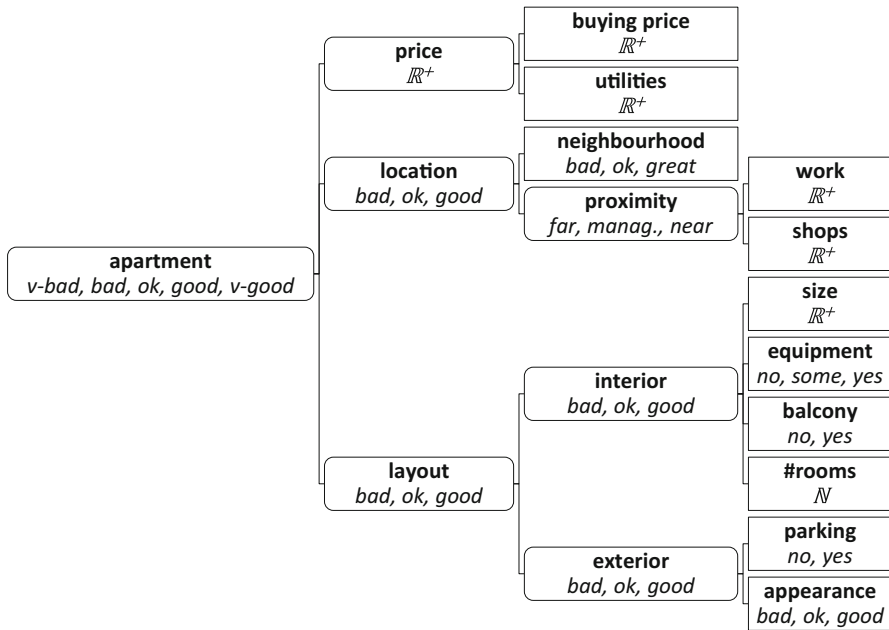


Fig. 5 Model for evaluating an apartment with numeric attributes included. Attributes with value scales \mathbb{R}^+ and \mathbb{N} are numeric; others are qualitative

work, *shops*, *size* and *#rooms* with their numeric counterparts by changing attributes' scales (Fig. 5). Due to these conversions, the aggregation functions numeric attributes' parents (i.e., *price*, *apartment*, *proximity* and *interior*) need to be replaced by functions of different types. All other functions remain the same.

Given the meaning of the attributes described above, we can safely assume that their numeric scales are all positive real numbers. An exception is the scale of *#rooms*, which is in the form of an integer:

$$D_{price} = D_{buyingprice} = D_{utilities} = D_{work} = D_{shops} = D_{size} = \mathbb{R}^+, \quad D_{\#rooms} = \mathbb{N}. \tag{26}$$

The new function for *price* is of Function type 3 ($NN \rightarrow N$). Suppose that we pay off the apartment in 20 years, so our combined price will represent the *buying price* with 20 years of monthly *utilities* costs added:

$$f_{price} = buyingprice + 12 \cdot 20 \cdot utilities. \tag{27}$$

Next, the aggregation function for *proximity* is defined based on distance to *work* and *shops* and has a qualitative output of *proximity* and is hence of Compound type $NN \rightarrow Q$. We compose this function using the schema $NN \rightarrow N \rightarrow Q$. The rationale for the following function definition is that we will be going to *work* five times a week, whereas will be going to the *shops* only twice a week. Here $g_{proximity}$ is the intermediate Function of type 3:

Table 10 The first part of the aggregation function for *interior*. The function arguments define a new function, which depends on the two numeric inputs (*size* and *#rooms*)

<i>equipment</i>	<i>balcony</i>	<i>interior</i>
<i>no</i>	<i>no</i>	0
<i>no</i>	<i>yes</i>	$\frac{size}{\#rooms}$
<i>some</i>	<i>no</i>	0
<i>some</i>	<i>yes</i>	$\frac{size}{\#rooms} + 10$
<i>yes</i>	<i>no</i>	$\frac{size}{\#rooms} + 15$
<i>yes</i>	<i>yes</i>	$\frac{size}{\#rooms} + 20$

$$g_{proximity} = 5 \cdot work + 2 \cdot shops,$$

$$NQ_{proximity}(y) = \begin{cases} \text{near} | y \in (0, 15] \\ \text{manag.} | y \in (15, 25] \\ \text{far} | y \in (25, \infty) \end{cases} \quad (28)$$

The aggregation function for *interior* is of the Compound type $NQ \rightarrow Q$. We shall compose it according to the schema $(NQ \rightarrow N) \rightarrow Q$ —that is: (1) first using the primitive Function type 5 and (2) discretizing the result. The output of (1) is an estimate of each room size (see Table 10). To include some preferential information in the table, constant values were added to favourable qualitative combinations (e.g. +20 for the combination (*yes, yes*) in Table 10). For the unfavourable combinations, we disregarded the numeric inputs and assigned the value 0. In step (2), we use the following discretization:

$$NQ_{interior}(y) = \begin{cases} \text{bad} | y \in [0, 25] \\ \text{ok} | y \in (25, 35] \\ \text{good} | y \in (35, \infty) \end{cases} \quad (29)$$

Finally, we need to define the aggregation function for *apartment*, which is also of the Compound type $NQ \rightarrow Q$. Similarly as before, we define Table 11 for step (1) and the following discretization for step (2):

$$NQ_{apartment}(y) = \begin{cases} \text{v-good} | y \in [-10,000, 50,000] \\ \text{good} | y \in (50,000, 90,000] \\ \text{ok} | y \in (90,000, 110,000] \\ \text{bad} | y \in (110,000, 140,000] \\ \text{v-bad} | y \in (140,000, \infty) \end{cases} \quad (30)$$

With this, the model is completely defined and ready for the evaluation of alternatives. The new values for the alternatives are presented in Table 12. All evaluation results are given in Table 13.

Let us illustrate the evaluation of the alternative “Big”. The other two alternatives follow the same procedure.

At first, the *price* of alternative “Big” is obtained by computing $f_{price}(85,000, 100)$, which is 109,000. Next, the value of $f_{proximity}(6, 5)$, is a two-stage function: at first,

Table 11 First part of the aggregation function for apartment

<i>location</i>	<i>layout</i>	<i>apartment</i>
<i>bad</i>	<i>bad</i>	$price + 20,000$
<i>bad</i>	<i>ok</i>	$price + 15,000$
<i>bad</i>	<i>good</i>	$price + 5000$
<i>ok</i>	<i>bad</i>	$\frac{price}{1.1} + 7500$
<i>ok</i>	<i>ok</i>	$\frac{price}{1.1}$
<i>ok</i>	<i>good</i>	$\frac{price}{1.1} - 5000$
<i>good</i>	<i>bad</i>	$\frac{price}{1.1}$
<i>good</i>	<i>ok</i>	$\frac{price}{1.1}$
<i>good</i>	<i>good</i>	$\frac{price}{1.2} - 10,000$

The function produces a new function dependent on the numeric input (*price*)

Table 12 The table gives new values for the input attributes for the three apartments: “Big”, “Equipped” and “Nice”

Alternative	buying price	utilities	neighbourhood	work	shops	size	equipment	balcony	#rooms	parking	appearance
Big	85.000	100	<i>bad</i>	6	5	80	<i>no</i>	<i>no</i>	3	<i>yes</i>	<i>bad</i>
Equipped	80.000	60	<i>ok</i>	2	4	40	<i>yes</i>	<i>yes</i>	2	<i>yes</i>	<i>ok</i>
Nice	50.000	90	<i>ok</i>	1	1.5	30	<i>some</i>	<i>no</i>	1	<i>no</i>	<i>ok</i>

Table 13 Evaluations of three alternatives according to the newly developed model

Name	price	proximity	location	interior	exterior	layout	apartment
Big	109.000	<i>far</i>	<i>bad</i>	<i>bad</i>	<i>ok</i>	<i>bad</i>	<i>bad</i>
Equipped	94.400	<i>manag.</i>	<i>ok</i>	<i>good</i>	<i>good</i>	<i>good</i>	<i>good</i>
Nice	71.600	<i>near</i>	<i>good</i>	<i>bad</i>	<i>bad</i>	<i>bad</i>	<i>good</i>

Values of aggregated attributes and final evaluation are presented

$g_{proximity}(6, 5)$ gives 40, which is discretized by $NQ_{proximity}$ to *far*. Function $f_{location}$ is a $QQ \rightarrow Q$ mapping, which is *bad* on the inputs *bad* and *far*. For *interior*, the value of the intermediate function is needed, which simply becomes 0 based on the worst possible combination of values of *equipment* and *balcony*. Then, $NQ_{interior}$ maps 0 to the value *bad*. The value for *exterior* is computed in the same way as in Sect. 2.2, and the resulting value is *ok*. The value of *layout* with $f_{layout}(bad,ok)$, which produces *bad*. Finally, the value *apartment*, which is of Compound type $NQ \rightarrow Q$, is computed in two steps. The first step selects the function $price + 20,000$, where $price = 109,000$, so the resulting value is 129,000. Using $NQ_{apartment}$, this value is discretized to the final evaluation—*bad*.

Let us compare these results with the evaluations acquired in Sect. 2.2. Because of the addition of numeric attributes, some values have changed, but most of them stayed the same. The most notable difference is the change of the final evaluation of alternative “Equipped”, which increased from *bad* to *good*. Due to added numerical attributes

and changed aggregation functions, the attributes *interior* and *layout* increased by one value for this alternative. Consequently, the apartments “Equipped” and “Nice” are both evaluated as *good*. Now, “Equipped” differs from “Nice” in terms of a better *interior*, *exterior* and *layout*. On the other hand, “Nice” is better than “Equipped” in terms of *price*, *proximity* and the *location* of the apartment.

We see that even though the developed model is more complex, the introduction of numeric attributes brought additional modelling possibilities to the example. The inclusion of numeric values allowed for a more natural interpretation of some concepts in comparison to describing them with qualitative values. For example, the price of an apartment is naturally expressible with monetary values. The extension further provides the ability to explicitly consider numeric concepts—the average room size is accurately measured with the quantity *size/#rooms*, whereas the same measure cannot be accurately obtained given qualitative values for *size* and *#rooms*.

4 Probabilistic and fuzzy distributions

In its current form, DEX evaluates alternatives with crisp values. The only exception is in cases when a decision table yields an interval of values; in this case, the resulting evaluation is generally a set of values. However, alternatives and the DM’s preferences are often *imprecise* or *uncertain*. For instance, alternatives’ values may be difficult to measure or assess precisely because they may depend on factors that cannot be controlled by the DM. Similarly, sometimes it is difficult for the DM to define a decision rule that would produce a single crisp value because the outcome is *distributed* between several values. Therefore, we propose an extension to DEX that explicitly models these types of uncertainty by introducing *probabilistic* or *fuzzy* distributions of values.

The extension of probabilistic distributions incorporates ideas from probabilistic inference methods (Durbach and Stewart 2012; Shachter and Peot 1992; Yang et al. 2006). These ideas have already been addressed in DEX to some extent. An early version of the method, called DECMAK, used probabilistic and fuzzy distributions for the representation of alternatives’ values (Rajkovič et al. 1987). Later, probabilistic distributions were introduced to aggregation functions, to facilitate the revision of the models (Žnidaršič et al. 2006a), which was implemented in the system called proDEX (Žnidaršič et al. 2006b). Some steps towards a decision making method with probabilistic extension were presented in Trdin and Bohanec (2012, 2013, 2014a) and applied in real-life decision making scenarios in Bohanec et al. (2016), Kontić et al. (2014). Similar approaches were explored by other authors (Bergez 2013; Holt et al. 2013; Omero et al. 2005).

Here, we propose a systematic extension to DEX which introduces probabilistic and fuzzy value distributions to both alternative values and aggregation functions, taking into account the formal model described in Sect. 2.1 and the other extensions proposed in this paper. The distributions are introduced at two points:

1. Values describing decision alternatives are extended to probabilistic or fuzzy distributions of values over corresponding attribute scales, and

2. outputs of aggregation functions are extended from crisp values or intervals to distributions of values.

That is, wherever we could have previously used a crisp value or an interval of values in a model, we may now use a distribution of values. This requires three changes to the formal model: (1) attribute scales have to be extended to cope with value distributions, (2) aggregation functions should in general map to distributions rather than intervals and (3) the evaluation procedure should propagate distributed values.

Value distributions are general in the sense that they are capable of representing all previous value types: single crisp values, value intervals and value sets. Required probabilistic and fuzzy distributions of values are added, though. Previously defined aggregation functions, which did not include distributions, can thus be used without change in the new setting.

4.1 Prerequisites

Given an attribute x_i and its scale D_i , we define a *value distribution* V as a tuple (B, r) , where $B \subseteq D_i$ and $r : B \rightarrow [0, 1]$. For finite B , we use the notation $V = \{v_{i_1} \setminus p_{i_1}, v_{i_2} \setminus p_{i_2}, \dots, v_{i_m} \setminus p_{i_m}\}$, where $p_{i_j} = r(v_{i_j})$ for each $j = 1, 2, \dots, m$. Generally, we do not impose any specific interpretation on p_{i_j} ; however, we are interested in two specific cases:

1. V is a *discrete probability distribution*: here, p_{i_j} s represent *probabilities* of corresponding elements v_{i_j} , and are normalized so that $\sum_{j=1}^m p_{i_j} = 1$.
2. V is a *fuzzy set* (Zadeh 1965): p_{i_j} s represent *fuzzy grades of membership* of corresponding elements v_{i_j} . Moreover, a fuzzy set is said to be *normalized* when $\max_{j=1}^m p_{i_j} = 1$.

In this way, we can represent all needed value types using both representations:

- A *crisp value* v can be represented as a value distribution $\{v \setminus 1\}$.
- A qualitative *interval* of values $[v_{i_j}, v_{i_k}]$, $j \leq k$, which includes all values $v_{i_j}, v_{i_{j+1}}, \dots, v_{i_k}$, is represented as a discrete probability distribution $\left\{v_{i_j} \setminus \frac{1}{k-j+1}, v_{i_{j+1}} \setminus \frac{1}{k-j+1}, \dots, v_{i_k} \setminus \frac{1}{k-j+1}\right\}$ or, alternatively, as a fuzzy set $\{v_{i_j} \setminus 1, v_{i_{j+1}} \setminus 1, \dots, v_{i_k} \setminus 1\}$.
- The *set* of values $V = \{v_{i_j}, v_{i_k}, \dots, v_{i_l}\}$, $|V| = m$ is represented as a discrete probability distribution $\{v \setminus \frac{1}{m} \mid v \in V\}$ or as a fuzzy set $\{v \setminus 1 \mid v \in V\}$.

For infinite S , we treat an attribute x_i as a random variable X_i and consider the *probability distributions* of its values. When X_i is a continuous random variable, we assume the existence of *probability density function* (Feller 1968) $f_{X_i} : D_i \rightarrow [0, 1]$, with $a, b \in D_i$, so that

$$\Pr[a \leq X_i \leq b] = \int_a^b f_{X_i}(t) dt. \tag{31}$$

When X_i is a discrete random variable, we assume the existence of *probability mass function* (Feller 1968) $f_{X_i} : D_i \rightarrow [0, 1]$, with $t \in D_i$, so that

$$f_{X_i}(t) = \Pr(X_i = t). \tag{32}$$

To formally describe the evaluation procedure using value distributions, we need to transform all value distributions to a common representation. When interpreted as probability distributions, both discrete and continuous variables can be treated in a unified way. In order to represent a discrete random variable with a probability density function, we employ the *Dirac delta function* (Hewitt and Stromberg 1965):

$$\begin{aligned} x \neq 0 &\Rightarrow \delta(x) = 0, \\ \int_{-\infty}^{\infty} \delta(x) dx &= 1. \end{aligned} \tag{33}$$

If a discrete random variable takes values $v_{i_1}, v_{i_2}, \dots, v_{i_k}$ with respective probabilities $p_{i_1}, p_{i_2}, \dots, p_{i_k}$, then its associated probability density function is:

$$f(t) = \sum_{j=1}^k p_{i_j} \delta(t - v_{i_j}). \tag{34}$$

Suppose there are n independent random variables $X_i, i = 1, \dots, n$ which are associated with probability density functions $f_{X_i}(x_i)$. Let $Y = G(X_1, X_2, \dots, X_n)$ be a random variable which is a function of X_i s. Then the density function $f_Y(y)$ is (Feller 1968; Hewitt and Stromberg 1965):

$$\begin{aligned} f_Y(y) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} f_{X_1}(x_1) f_{X_2}(x_2) \dots f_{X_n}(x_n) \\ &\delta(y - G(x_1, x_2, \dots, x_n)) dx_1 dx_2 \dots dx_n. \end{aligned} \tag{35}$$

In general, solving this multiple integral is a difficult problem to tackle analytically, thus we employ statistical sampling with Monte Carlo method (Cafisch 1998) for random variables defined with a probability density function: a random variable X_i is sampled m times, giving a finite discrete probability distribution $V_i = \{v_j \setminus \frac{1}{m} | v_j \in X_i, j = 1, 2, \dots, m\}$.

Now, we can consider only value distributions with a finite number of elements for probabilistic aggregation. Given some aggregation function g , we need to compute the value distribution $W = g(V_1, V_2, \dots, V_n)$, where each distribution V_i is obtained either by sampling or as a result of computation on lower levels of the attribute hierarchy. Also note that function g is defined only on atomic crisp values; see Eq. (8). In order to obtain $W = \{w_1 \setminus p_1, w_2 \setminus p_2, \dots, w_k \setminus p_k\}$, where $w_i \in D_Y$, we need to compute all $p_i, i = 1, 2, \dots, k$:

$$p_i = \sum_{(v_1 \setminus q_1, v_2 \setminus q_2, \dots, v_n \setminus q_n) \in V_1 \times V_2 \times \dots \times V_n} \prod_{w_i \setminus r_i \in g(v_1, v_2, \dots, v_n)} r_i \prod_{j=1}^n q_j. \tag{36}$$

This expression produces a probability for value w_i in the final value distribution W . The outer sum runs over the Cartesian product of all basic values in input value distributions. Given some value combination v_1, v_2, \dots, v_n , the basic function $g(v_1, v_2, \dots, v_n)$ is computed. The result of computation is generally a value distribution $\{w_1 \setminus r_1, w_2 \setminus r_2, \dots, w_k \setminus r_k\}$. From this distribution, only the probability r_i of w_i is considered. This particular combination adds to the outer sum the probability of the output r_i multiplied by the probability of this value combination ($\prod_{j=1}^n q_j$). This procedure follows the intuitive explanation of handling probabilities. Furthermore, Eq (36) is in line with the constraint imposed in Eq. (10), specifying that the function F_i should for singleton input value distributions produce the same value as the function f_i .

Fuzzy sets are handled in the same way, except that product and sum operators are replaced by the standard fuzzy set operators: *t-norms* (\top) as logical conjunction and the corresponding *t-conorms* (\perp) as logical disjunction (Bede 2012):

$$p_i = \sum_{(v_1 \setminus q_1, v_2 \setminus q_2, \dots, v_n \setminus q_n) \in V_1 \times V_2 \times \dots \times V_n} \perp (0, \top_{w_i \setminus r_i \in g(v_1, v_2, \dots, v_n)} (r_i, \top (q_1, q_2, \dots, q_n))). \tag{37}$$

The most commonly used t-norms and t-conorms in fuzzy sets are minimum and maximum, respectively:

$$\top (a, b) = \min (a, b), \tag{38}$$

$$\perp (a, b) = \max (a, b). \tag{39}$$

By substituting the t-norm and t-conorm, we get an equation for computing p_i s for fuzzy sets:

$$p_i = \sum_{(v_1 \setminus q_1, v_2 \setminus q_2, \dots, v_n \setminus q_n) \in V_1 \times V_2 \times \dots \times V_n} \max (0, \min_{w_i \setminus r_i \in g(v_1, v_2, \dots, v_n)} (r_i, \min (q_1, q_2, \dots, q_n))). \tag{40}$$

This equation can be interpreted similarly to Eq. (36), where product and sum are replaced by min and max, respectively.

4.2 Extension formalization

In order to formally extend DEX to handle value distributions, we first need to extend the scales of attributes. For practical reasons, we wish to keep the definition of value scale D_i of attribute $x_i \in X$ the same as before. However, we extend the range of aggregation functions to include value distributions over D_i and accommodate the propagation of value distributions during evaluation. Thus, we define ED_i as the space of all value distributions over D_i . When D_i is a finite discrete scale $D_i = \{v_{i_1}, \dots, v_{i_{m_i}}\}$, then

$$ED_i = \left\{ \left\{ v_{i_1} \setminus p_{i_1}, \dots, v_{i_{m_i}} \setminus p_{i_{m_i}} \right\} \mid p_{i_j} \in [0, 1], v_{i_j} \in D_i, j = 1, 2, \dots, m_i \right\}. \tag{41}$$

When D_i is infinite, then $ED_i = X_i$ is a random variable with an associated probability density or probability mass function f_{x_i} as defined in Eq. (31, 32).

Then, we need to extend Eq. (8) so that each aggregation function f_i can, in general, map to extended domain ED_i instead of the space of intervals $I(D_i)$:

$$f_i : D_{i_1} \times D_{i_2} \times \dots \times D_{i_k} \rightarrow ED_i, \{x_{i_1}, x_{i_2}, \dots, x_{i_k}\} = S(x_i). \tag{42}$$

This substitution affects the aggregation procedure carried out at each aggregated attribute x_i . When evaluating the associated aggregation function f_i , we should expect value distributions at inputs. Also, the output result will, in general, be a value distribution, too. Therefore, using $\{x_{i_1}, x_{i_2}, \dots, x_{i_k}\} = S(x_i)$ we have to redefine the function F_i , defined in Eq (9) to

$$F_i : ED_{i_1} \times ED_{i_2} \times \dots \times ED_{i_k} \rightarrow ED_i. \tag{43}$$

When encountering a finite value distribution of values from some scale, the evaluation must correctly propagate the corresponding probabilities or membership values to construct the final evaluation. Even though the scales of attributes have been extended, the aggregation functions f_i stay the same—only the evaluation procedure needs to be adapted to handle the new extended scales. The adaptation defines how the aggregation is performed for function F_i , using function f_i . In Sect. 4.1, we suggested handling only finite value distributions since infinite value distributions are sampled.

We suggest using two types of aggregation: probabilistic aggregation or fuzzy aggregation. We leave the choice to the user of the model and assume that the type of aggregation has been defined in advance. We also assume that all value distributions involved in the aggregation are normalized accordingly—that is, $\sum_{i=1}^k p_i = 1$ for probabilistic aggregation and $\max_{i=1}^k p_i = 1$ for fuzzy aggregation with normalization. The computation of F_i is performed according to Eq. (36) for probabilistic aggregation and according to Eq. (40) for fuzzy aggregation. In both cases, the obtained result is a value distribution which can be interpreted as a probability or fuzzy distribution, respectively.

For practical reasons, we introduce the final step that simplifies the obtained value distributions so that they are more readable for the user. The procedure is called *simplify*: at input, it takes the full representation of a value distribution $v = \{v_1 \setminus p_1, v_2 \setminus p_2, \dots, v_n \setminus p_n\}$ and produces an equivalent simplified representation. It detects whether v represents some special distribution, such as a single crisp value, an interval or a set. Formally, $simplify_i(v)$, applied on the value distribution v assigned to attribute x_i , is a recursive procedure defined as follows:

$$simplify_i(v) = \begin{cases} simplify_i(\{v_1, v_2, \dots, v_n\}) | v = \{v_1 \setminus p_1, v_2 \setminus p_2, \dots, v_n \setminus p_n\}, p_1 = p_2 = \dots = p_n \\ simplify_i([v_l, v_h]) | v = \{v_1, v_2, \dots, v_n\}, D_i \cap v = [v_l, v_h] \\ v_l | v = [v_l, v_h], v_l = v_h \\ v | \text{otherwise} \end{cases} \tag{44}$$

Here, the first case f_i detects that all p_i s are equal, and v represents at least a set. The second case determines whether v is an interval, and the third case checks whether an interval has the same bounds and thus represents a single value.

When using value distributions, evaluations cannot always be compared directly. Using stochastic dominance (Hadar and Rusell 1969) on probabilistic value distributions is one possibility. Suppose we are given two value distributions $V_1, V_2 \in ED_i$, where D_i is a preferentially ordered scale. We say that V_1 (stochastically) dominates V_2 if and only if:

$$\begin{aligned} \forall w \in D_i : \Pr[V_1 \geq w] &\geq P[V_2 \geq w], \\ \exists w \in D_i : \Pr[V_1 \geq w] &> P[V_2 \geq w]. \end{aligned} \quad (45)$$

Stochastic dominance can bring some insight into the final evaluations; however, it cannot always generate a total order of alternatives. This means that the best alternative may not exist for the current model and its input values. In such cases, the DM must interpret the results, and chose the best alternative among those that are not dominated.

4.3 Didactic example

We continue the didactic example from Sect. 3.2. Hereafter, we shall include value distributions at alternatives' input values and leave aggregation functions unchanged. Probabilistic value distributions are included in alternatives according to the following rationale. Observations are quantified in Table 14.

- Alternative “Big” does not actually have a fixed utilities bill of 100; rather, the value is uniformly distributed through the year in the amount from 80 to 120. The same applies to alternative “Nice”, but the value is distributed between 60 and 80. However, the utilities costs for apartment “Equipped” are still fixed at 60.
- Recall that we do not know the quality of the neighbourhood of alternative “Equipped”. For this reason we previously decided to assign the middle value. Now we are able to specify that the value is actually undetermined or unknown, assigning an interval of all possible values.
- The *equipment* of apartments “Big” and “Nice” was previously determined by rule of thumb, and we felt that specifying a single value was too limited. Instead we now specify the situation with a distribution.
- Parking for apartment “Equipped” is usually available, but sometimes it is not. On the other hand, parking may sometimes be available for alternative “Nice”. With apartment “Big”, there is a dedicated parking space.
- Before, we were not able to decide about the outer appearance of the apartment “Nice”. Now we know that the appearance is something between *ok* and *good*.

Because the model aggregation functions and scales of attributes did not change from the previous example, we can immediately evaluate the alternatives. As in previous examples, we will show the evaluation of the first alternative “Big” and present the evaluations of the other two alternatives. All alternatives' evaluations are summarized in Table 15.

First, “Big’s” *price* is computed using F_{price} with argument values of 85,000 and $U(80, 120)$. The values are interpreted as value distributions: 85,000 is $\{85,000 \setminus 1\}$ and $U(80, 120)$ is sampled m times (we choose $m = 100$), obtaining the value distribution $\{80.05 \setminus 0.01, 111.96 \setminus 0.01, \dots, 90 \setminus 0.01\}$. For all 100 combinations of values in *buying price* ($\{85.000 \setminus 1\}$) distribution and *utilities*

Table 14 The table gives new values with probabilistic value distributions for the input attributes

<i>Name</i>	<i>buying price</i>	<i>utilities</i>	<i>neighbourhood</i>	<i>work</i>	<i>shops</i>	<i>size</i>	<i>equipment</i>	<i>balcony</i>	<i>#rooms</i>	<i>parking</i>	<i>appearance</i>
Big	85.000	$U(80, 120)$	<i>bad</i>	6	5	80	{ <i>no</i> \0.9, <i>some</i> \0.1}	<i>no</i>	3	<i>yes</i>	<i>bad</i>
Equipped	80.000	60	[<i>bad</i> , <i>great</i>]	2	4	40	<i>yes</i>	<i>yes</i>	2	{ <i>no</i> \0.1, <i>yes</i> \0.9}	<i>ok</i>
Nice	50.000	$U(60, 80)$	<i>ok</i>	1	1.5	30	{ <i>some</i> \0.6, <i>yes</i> \0.4}	<i>no</i>	1	{ <i>no</i> \0.9, <i>yes</i> \0.1}	[<i>ok</i> , <i>good</i>]

Table 15 Evaluations of three alternatives according to the model developed with the non-determined input values of alternatives

<i>Name</i>	<i>price</i>	<i>proximity</i>	<i>location</i>	<i>interior</i>	<i>exterior</i>	<i>layout</i>	<i>apartment</i>
Big	{104.212\0.01, 111.870.4\0.01, ..., 106.600\0.01}	<i>far</i>	<i>bad</i>	<i>bad</i>	<i>ok</i>	<i>bad</i>	<i>bad</i>
Equipped	94.400	<i>manag.</i>	[<i>bad, good</i>]	<i>good</i>	{ <i>bad</i> \0.1, <i>good</i> \0.9}	{ <i>ok</i> \0.1, <i>good</i> \0.9}	{ <i>ok</i> \0.33, <i>good</i> \0.67}
Nice	{64.431.2\0.01, 68.204\0.01, ..., 67.184\0.01}	<i>near</i>	<i>good</i>	{ <i>bad</i> \0.6, <i>good</i> \0.4}	{ <i>bad</i> \0.9, <i>good</i> \0.1}	{ <i>bad</i> \0.54, <i>ok</i> \0.42, <i>good</i> \0.04}	{ <i>good</i> \0.96, <i>v-good</i> \0.04}

distribution $(\{80.05 \setminus 0.01, 111.96 \setminus 0.01, \dots, 90 \setminus 0.01\})$, function f_{price} is evaluated during computation of F_{price} . The final produced evaluation for *price* is $\{104,212 \setminus 0.01, 111,870.4 \setminus 0.01, \dots, 106,600 \setminus 0.01\}$.

The values of *proximity* and *location* stay the same as before because there were no changes in their sub-trees.

The interior value of apartment “Big” is calculated using function $F_{interior}$. Values supplied as arguments to the function are $(80, no, no, 3)$ with the probability 0.9 and $(80, some, no, 3)$ with the probability 0.1. In both cases, an intermediate value 0 is produced, which creates the value *bad* by function $NQ_{interior}$. Even though apartment “Big” has some equipment, it does not improve the evaluation of interior. The final value distribution $\{bad \setminus 1\}$ is simplified to *bad*.

The value for *exterior* stays the same as in Sect. 3.2. Even though *layout*’s inputs are now value distributions, a crisp value of *bad* is obtained. The apartment may come with some equipment, but the value of *interior* did not change, which in turn means that the value for *layout* will not change, either.

The final evaluation for apartment “Big” is acquired with function $F_{apartment}$. The values supplied to the function are $\{104,212 \setminus 0.01, 111,870.4 \setminus 0.01, \dots, 106,600 \setminus 0.01\}$, *bad* and *bad*. For each combination of values, $f_{apartment}$ is computed. Note, probability for each value combination is 0.01. Because the only possible values for *location* and *layout* are *bad*, function $price + 20,000$ from space $\Phi(x_{apartment})$ is always used. The first value from the distribution gives 124,212, which results in value *bad* by $NQ_{apartment}$. This is done for all values inside the value distribution of *price*. $NQ_{apartment}$ always produces *bad*, which in turn, after simplification, produces the final evaluation of *bad*.

The final evaluations now provide an additional insight regarding the alternatives. Even though the *utilities* costs will be variable for alternative “Big”, and there is some equipment in the apartment, the final evaluation is still *bad*. On the other hand, we can now better compare apartments “Equipped” and “Nice”. With only numeric values, we were not able to distinguish between the apartments in terms of the final evaluation. Now we can see that apartment “Equipped” is mostly *good* (with probability 0.67), but there is the probability of 0.33 that the evaluation is a bit worse, giving the evaluation *ok*. For the apartment “Nice”, the evaluation is strongly *good* (probability 0.96) with a small chance (probability 0.04) that the apartment is even *very good*. Overall, apartment “Nice” is better than “Equipped”.

Furthermore, we can check if apartment “Nice” stochastically dominates apartment “Equipped”. For that, we check for each value $v \in D_{apartment}$, that the probability of the value v or better in the evaluation of “Nice” (V_{Nice}) is greater or equal than the probability of the value v or better in the evaluation of “Equipped” ($V_{Equipped}$):

$$\begin{aligned}
 1 &= \Pr[V_{Nice} \geq v\text{-bad}] \geq P[V_{Equipped} \geq v\text{-bad}] = 1, \\
 1 &= \Pr[V_{Nice} \geq bad] \geq P[V_{Equipped} \geq bad] = 1, \\
 1 &= \Pr[V_{Nice} \geq ok] \geq P[V_{Equipped} \geq ok] = 1, \\
 1 &= \Pr[V_{Nice} \geq good] \geq P[V_{Equipped} \geq good] = 0.67, \\
 0.04 &= \Pr[V_{Nice} \geq v\text{-good}] \geq P[V_{Equipped} \geq v\text{-good}] = 0.
 \end{aligned}
 \tag{46}$$

Additionally, for the apartment “Nice” to stochastically dominate apartment “Equipped”, we need to find a value $v' \in D_{apartment}$, for which the probability of producing value at least v' in V_{Nice} is strictly greater than probability of producing v' in $V_{Equipped}$. There are two such values: *good* and *v-good*:

$$\begin{aligned} 1 &= \Pr[V_{Nice} \geq \textit{good}] > P[V_{Equipped} \geq \textit{good}] = 0.67, \\ 0.04 &= \Pr[V_{Nice} \geq \textit{v-good}] > P[V_{Equipped} \geq \textit{v-good}] = 0. \end{aligned} \quad (47)$$

Therefore, “Nice” stochastically dominates “Equipped”, and can be considered a better choice. In the same way, it can be shown that both “Nice” and “Equipped” dominate “Big”.

5 Relational models

The data encountered in everyday life are frequently of a relational nature in the sense that one entity is composed of several similar *sub-entities*—similar to the extent that they can be evaluated by the same criteria. For example, when evaluating a company, a DM may want to evaluate all departments of that company. Here the company’s departments are the similar sub-entities, and there is a “one-to-many” relationship between the company and departments. All departments can be evaluated in a similar way, by the same model. The problem, however, is that an arbitrary number of sub-evaluations are acquired in this way, which need to be combined in the evaluation of the main alternative (company).

Currently, relational models are not supported in DEX. Adding them would be a substantial improvement, which would facilitate addressing a much larger group of decision problems. Introducing relational models, however, requires extensions that affect the representation of decision alternatives and introduces new components, such as relational attributes and relational aggregation functions. The purpose of this section is to extend the existing DEX method for the support of relational models.

Relational data are frequently modelled in relational databases. Several disciplines of machine learning explicitly consider the development of relational models—for example, inductive logic programming (Lavrač and Džeroski 1994). Relational data are also studied in quantitative multi-criteria decision making methods, but rarely in an explicit way. There, it rarely causes difficulties because it naturally involves common operators based on summation and averaging while at the same time handling numeric values. Such operators are useless in qualitative settings and require special approaches.

We propose to extend DEX to handle situations where one alternative is composed of several sub-alternatives (see examples in (Bohanec et al. 2014; Trdin and Bohanec 2012, 2013, 2014a, b)). For this purpose, we propose to modify the modelling process by developing two models: one (M) for the evaluation of the main alternative and another (RM) for the evaluation of relational sub-alternatives. To evaluate a single main alternative, each sub-alternative is first evaluated by RM . Then, all evaluations are aggregated, providing an input value to M . The evaluations in both M and RM

are carried out in the same way as described previously; the only differences occur at the point where the two models are connected with each other.

Relational models were already employed with DEX in two real-world use cases, in which they proved very useful for defining a decision model and evaluating alternatives on relational data: the reputational risk assessment of banks (Bohanec et al. 2014) and the appraisal of energy production technologies in Slovenia (Bohanec et al. 2016; Kontić et al. 2014).

5.1 Extension formalization

To formalize this extension, we need to introduce three new entities and one type of aggregation function. The three new entities are: (1) relational alternative, (2) relational model and (3) relational aggregated attribute. The new aggregation function is called relational aggregation function and is used only by the relational aggregated attributes.

Given some alternative $a \in A$, the term *relational alternatives* denotes all entities $ra \in RA$ that are in a many-to-one relation with a . For instance, $a \in A$ may be a company and $ra \in RA$ one of its departments, assuming a one-to-many relation $department : A \rightarrow RA$. Since RM , as any other model, contains input attributes, it also holds that

$$ra \in inputSpace_{RM}. \tag{48}$$

The purpose of the *relational model* RM is to evaluate relational alternatives RA . RM is modelled in the same way as any other DEX model; the only difference is in its purpose in the evaluation procedure: it evaluates relational alternatives RA and provides inputs to the main model M , which evaluates alternatives A .

A *relational aggregated attribute* rx is a special type of attribute that provides a connection point between the main model M and the relational model RM . rx 's scale is formalized in the same way as for other attributes. The relational attribute rx is connected to its counterpart output attribute ox from RM ; rx serves as an input to M . In other words, rx is a connecting point between M and RM : it receives output from RM (through attribute ox) and provides input to M .

As there is a one-to-many relation between A and RA , the aggregation at rx should aggregate all output values coming from RM , that correspond to one alternative $(V_1, V_2, \dots, V_n), V_i \in ED_{ox}$, into a single input value $V \in ED_{rx}$ of M . Here, n is the number of relational alternatives $ra \in RA$ evaluated by RM that are part of a single alternative $a \in A$ evaluated by M .

Consequently, the aggregation function f_{rx} is special in the sense that it aggregates an arbitrary number of values coming from ox rather than single values from descendant attributes. Formally:

$$V = f_{rx}(V_1, V_2, \dots, V_n), \tag{49}$$

$$f_{rx} : (ED_{ox})^n \rightarrow ED_{rx}. \tag{50}$$

In general, f_{rx} is any aggregation function defined for an arbitrary number of arguments—for example, *min*, *max*, *sum*, *average*, *mean* or *count*. The value of

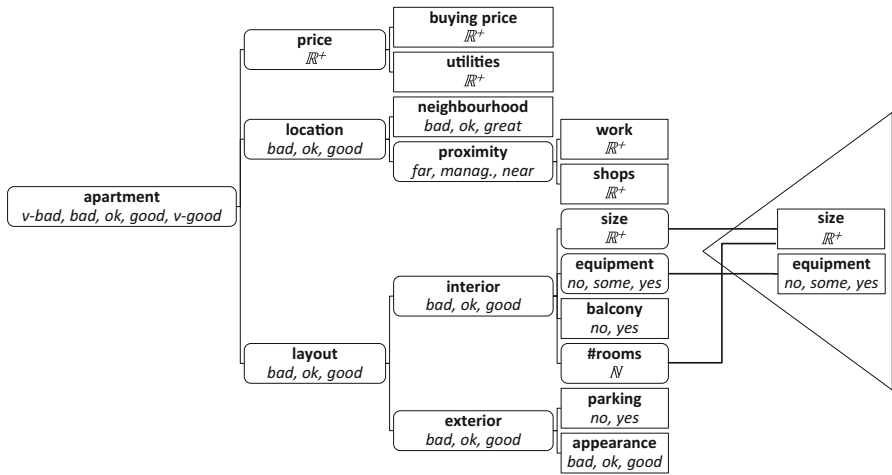


Fig. 6 Model for evaluating an apartment with support for numeric values and relational models. The nodes with round corners are aggregated attributes and nodes with pointed corners are input attributes. The triangle at the far right represents the relational model for evaluation of rooms

the function is computed with the corresponding function F_{rx} as per Eq. (43). The function is constrained to the types of ox 's and rx 's scales. For instance, *sum* cannot be applied to qualitative values.

Given this formalization, an arbitrary number of relational models is supported in some model M . Moreover, an arbitrary number of relational models can be nested inside each other. Therefore, decision problems with nested relational alternatives can be solved as well.

5.2 Didactic example

To show the relational models in practice, we will once again extend the previously developed model, taking into account that each apartment consists of multiple rooms. Each room has some features worth evaluating, such as *size* and *equipment*. Thus, in addition to the main model for apartments, we introduce a relational model for the evaluation of rooms. The relational model contains two attributes: (1) *size*, a numeric attribute giving the absolute size of the room and (2) *equip.*, a qualitative attribute giving the amount and quality of equipment in the room.

Due to the nature of this design, three relational aggregated attributes are introduced—more precisely, exchanged for their current counterparts—in the apartment model: (1) *size*'s value is computed by summing the values of *size* attribute from the room relational model, (2) *equipment* transforms the room *equipment* values with uniform weighting and (3) *#rooms* counts the number of rooms by counting the number of *size*'s values that come from the relational model.

Figure 6 shows the new structure of the model. The apartment model structure is the same as in Fig. 5. However, the relational room model now appears on the right-hand side and is connected to the model through the relational attributes.

To complete the model, we need to define the relational aggregation functions for *size*, *equipment* and *#rooms*. According to the model idea described above, the functions are defined as follows:

$$\begin{aligned}
 f_{size}(v_1, v_2, \dots, v_n) &= \sum_{i=1}^n v_i, \\
 f_{equipment}(v_1, v_2, \dots, v_n) &= \sum_{i=1}^n \frac{1}{n} v_i, \\
 f_{\#rooms}(v_1, v_2, \dots, v_n) &= count(v_1, v_2, \dots, v_n) = n.
 \end{aligned}
 \tag{51}$$

To use the new model, we have to define input values for rooms; see Table 16.

The evaluation procedure is almost the same as before. The values for the *price*, *proximity* and *location* are computed in the same way as before. However, some inputs to the sub-tree *layout* are now obtained from the relational model and may differ from previous calculations. The main difference is in the aggregation that involves functions f_{size} , $f_{equipment}$ and $f_{\#rooms}$. The aggregation function f_{size} first evaluates the size of three rooms of the “Big” apartment, which give values of {40, 25, 15}. Function f_{size} produces their sum—80. Similarly, the value for *equipment* is computed. The evaluation of rooms gives {no, {no\0.9, some\0.1}, no}. The function $f_{equipment}$ weighs each value with 1/3, and the final distribution of {no\0.97, some\0.03} is produced. This distribution cannot be simplified. The computation of *#rooms* follows the same principle as the computation for *size*, except that $f_{\#rooms}$ counts the number rather than the sum of values and gives 3. These values now enter the calculations as ordinary inputs of the apartment model, giving the results as shown in Table 17.

The final evaluation of alternative “Big” did not change due to the relational aggregation functions computing the same values as before. The same holds for the “Nice” apartment. However, the new evaluation of “Equipped” is different. Previously, the value of *equipment* was *yes*. Now, as we explicitly considered two rooms, one of which is small and incompletely equipped, the aggregation of the rooms gives the value distribution {yes\0.95, some\0.05}. Thus, the value *some* is additionally propagated through the aggregation and, even though it has a very small probability of 0.05, it affects the final outcome, which becomes {bad\0.0017, ok\0.33, good\0.665} instead of {ok\0.33, good\0.67}.

6 Discussion

We introduced three extensions to the DEX method: numeric attributes, value distributions, and relational alternatives and models. All extensions were motivated by needs identified when applying DEX in complex real decision modelling tasks. They substantially increase the range of decision problems that can be addressed, but they also require a number of changes and additions to the method. In order to introduce numeric attributes, it was necessary to add a class of numeric value scales, which in turn increased the number of primitive function types from one to five. The introduction of value distributions required a generalization of value scales to extended scales and an

Table 16 The table gives new values for the input attributes with the relational model

<i>Alternative</i>	<i>buying price</i>	<i>utilities</i>	<i>neighbourhood</i>	<i>work</i>	<i>shops</i>	<i>balcony</i>	<i>parking</i>	<i>appearance</i>	<i>room alternatives size</i>	<i>equipment</i>
Big	85.000	$U(80, 120)$	<i>bad</i>	6	5	<i>no</i>	<i>yes</i>	<i>bad</i>	40	<i>no</i>
Equipped	80.000	60	[<i>bad, great</i>]	2	4	<i>yes</i>	{ <i>no</i> \0.1, <i>yes</i> \0.9}	<i>ok</i>	25	{ <i>no</i> \0.9, <i>some</i> \0.1}
Nice	50.000	$U(60, 80)$	<i>ok</i>	1	1.5	<i>no</i>	{ <i>no</i> \0.9, <i>yes</i> \0.1}	[<i>ok, good</i>]	15	{ <i>some</i> \0.1, <i>yes</i> \0.9}
									30	{ <i>some</i> \0.6, <i>yes</i> \0.4}

The right-hand side of the table gives input values for relational alternatives

Table 17 Evaluations of three alternatives according to the model developed with the relational model included

Alternative	price	proximity	location	size	equipment	#rooms	interior	exterior	layout	apartment
Big	{109.881.3\0.01, 104.231.8\0.01, ..., 107.310.1\0.01}	far	bad	80	{no\0.97, some\0.03}	3	bad	ok	bad	bad
Equipped	94.400	manag.	[bad,good]	40	{some\0.05, yes\0.95}	2	{ok\0.05, good\0.95}	{bad\0.1, good\0.9}	{bad\0.005, ok\0.14, good\0.855}	{bad\0.002, ok\0.33, good\0.665}
Nice	{64.521.2\0.01, 69.194\0.01, ..., 65.238.1\0.01}	near	good	30	{some\0.6, yes\0.4}	1	{bad\0.6, good\0.4}	{bad\0.9, good\0.1}	{bad\0.54, ok\0.42, good\0.04}	{good\0.96, v-good\0.04}

The values of the aggregated attributes (including relational attribute values for size, equipment and #rooms) are presented

extension of the evaluation algorithm to use existing aggregation functions on value distributions. The new ability to represent and evaluate relational alternatives required the introduction of relational models, relational attributes and relational aggregation functions. All the extensions are compatible with each other so that they can be used simultaneously.

Considering the previous applications of DEX and the solutions found in some other MCDA methods, the extensions are not entirely new but are for the first time systematically brought together and formalized. Previously, DEX was combined with numeric attributes and relational models in the modelling and assessment of bank reputational risk (Bohanec et al. 2014), qualitative relational models were introduced in the assessment of public administration e-portals (Leben et al. 2006) and qualitative probabilistic distributions were used in ecological domains (Bohanec et al. 2009; Bohanec 2008; Bohanec and Žnidaršič 2008; Žnidaršič et al. 2008). All three extensions were for the first time used together in the evaluation of sustainable electrical energy production in Slovenia (Bohanec et al. 2016; Kontić et al. 2014). These applications were typically formulated and implemented in an ad-hoc manner and tailored to a particular application. However, they clearly indicated the need for extending the method.

The proposed extensions provide new means for representing a DM's knowledge and preferences and facilitate solving a wider variety of decision problems. Specifically, the introduction of numeric attributes allows for a more natural treatment of numeric quantities; instead of being limited only to qualitative attributes, the DM can choose between the qualitative and quantitative representations, as illustrated in Sect. 3.2. Numeric attributes also solve a common problem in DEX: mapping from numeric measurements to qualitative input values. To date, this had to be done implicitly and manually by the DM, whereas the extended method facilitates this mapping with ease. The introduction of numeric attributes opens DEX for the inclusion of other approaches of quantitative MCDA, such as pairwise preference and weights elicitation of AHP (Saaty 2008; Saaty and Vargas 2012) and using marginal utility functions of MAUT (Multi-attribute utility theory) (Wang and Zionts 2008). Another possible approach to build aggregation functions, especially in case of numeric values, is the ordinal regression (Jacquet-Lagrèze and Siskos 1982, 2001; Mihelčič and Bohanec 2016). An approach to multiple criteria sorting based on robust ordinal regression (Kadziński et al. 2014) seems particularly suitable. DEX requires a direct acquisition of utility functions; outranking MCDA methods that acquire DM's preferences through comparison of alternatives, such as ELECTRE (Roy 1991) and PROMETHEE (Brans and Vincke 1985), are less suitable for this purpose.

The second extension, value distributions, introduces capabilities that are well known and proven in contexts such as expert systems, fuzzy control systems and uncertainty and risk analysis. They allow an explicit formulation of "soft" (imprecise, uncertain and even missing) knowledge and data. Specifically, we introduced two types of value distributions: probabilistic, which are suitable for representing uncertainty, and fuzzy to represent vague concepts and values. The former resembles the use of probabilities in MAUT (Wang and Zionts 2008), whereas the latter is in line with the trend of extending MCDA methods with fuzzy sets (Baracskai and Dörfler 2003; Kahraman 2008; Omero et al. 2005).

The third extension addresses decision problems in which alternatives are composed of similar sub-components. In quantitative MCDM, such problems are rarely mentioned because they can be easily handled by common aggregation functions, such as the average. In the qualitative world of DEX, there are no such obvious functions – thus the need for the extension. The key contribution of relational models is allowing the DM to granulate the decision problem to a finer level, explicitly considering parts of the whole.

The extensions increase the complexity of model development. Previously in DEX, there was just one conceptual way to define an attribute or an aggregation function; now there is plenty from which the DM can choose. Even though this increases the flexibility of the modelling, it also requires the mastering of newly available tools. Particularly, the elicitation of attributes and aggregation functions becomes more difficult. In addition to dealing with new types of numeric attributes, the DM should control the interplay between qualitative and numeric attributes, for which there are now five different primitive forms instead of just one (see Sect. 3). Regarding aggregation functions, the complexity is increased by a new class of numerical functions, which have to be formulated by the DM in accordance with his or her preferences. Regarding the evaluation of alternatives, DEX was previously limited to only qualitative values and their sets. The extended DEX additionally produces a wide variety of other value types: integers, real numbers and fuzzy or probabilistic value distributions; this increases the complexity of the evaluation results and requires more efforts for their interpretation. In summary, the increased complexity of the models and computed values may degrade the comprehensibility of the method and results for the DM. Consequently, at this stage the extended DEX seems a more useful tool for a skilled decision analyst than for an ordinary DM.

We are aware that retrieving knowledge from the DM, especially with methods of such complexity, is a difficult problem. In this paper, we were not concerned with the difficulties of knowledge acquisition; our primary goal was to open up the DEX method to make it more suitable for addressing a wide range of real decision problems. We were interested in giving the DM the ability to specify his/her preference as flexibly as possible. We are aware that more research and practice is needed to fully assess the strengths and weaknesses of the extended method and to identify potential difficulties associated with preference elicitation. In further research, we will attempt to find a suitable balance between the flexibility, simplicity and comprehensibility of the method, which may even require narrowing down its broadness.

In practice, a method such as DEX requires the support of computer software for both the development of models and evaluation of alternatives. While the basic DEX method is already implemented in the software DEXi, the extended method has not been fully implemented yet. To date, we have developed a java software library that supports all the proposed extensions, but it cannot be used interactively by the DM. Instead, a model is created and used through program code. The library was used to model water flows in agriculture (Kuzmanovski et al. 2015) and was incorporated into a web service to evaluate user-supplied alternatives through a web browser (Kuzmanovski et al. 2015). In the future, we aim to implement an interactive computer program that will fully support the extended DEX method in a way that is similar to the way the current program DEXi supports DEX.

7 Conclusion

The main contribution of this paper is an extension of the already established qualitative multi-criteria modelling method DEX. Based on identified practical needs in complex decision situations, we proposed three extensions: including numeric attributes, evaluating alternatives with probabilistic and fuzzy value distributions and supporting the evaluation of relational alternatives. In order to support these concepts, a number of formal and algorithmic components were added, specifically: numeric scale types, extended value domains, probabilistic and fuzzy value distributions, numeric aggregation functions, five primitive function types and relational attributes, aggregation functions, models and alternatives. All these concepts greatly increase the flexibility and representational richness of the method, enlarge the class of decision problems that can be addressed and improve the decision process by providing additional methods and tools to the decision maker. On the other hand, the extensions increase the complexity of the method and developed models themselves, which may have negative effects on the efficiency and comprehensibility of the modelling. Due to this complexity, the extended DEX method is currently aimed at skilled decision analysts rather than ordinary decision makers.

Acknowledgements Funding was provided by ARRS, the Slovenian Research Agency, grant number 1000-11-310228.

References

- Bana e Costa C, Vansnick J-C (1999) The MACBETH approach: basic ideas, software, and an application. In: Meskens N, Roubens M (eds) *Advances in decision analysis*, vol 4. Kluwer Academic Publishers, Netherlands, pp 131–157
- Baracskaï Z, Dörfler V (2003) Automated fuzzy-clustering for Doctus expert system. In: *International conference on computational cybernetics*, Siófok, Hungary
- Bede B (2012) *Mathematics of fuzzy sets and fuzzy logic*. In: Kacprzyk J (ed) *Studies in fuzziness and soft computing*, vol 295. Springer, Heidelberg
- Bergez J-E (2013) Using a genetic algorithm to refine worst-best and best-worst options of a DEXi-type model: application to the MASC model of cropping-system sustainability. *Comput Electron Agric* 90:93–98
- Bohanec M (2014) DEXi: program for multi-attribute decision making: user's manual, version 4.01. IJS Report DP-11739. Jožef Stefan Institute, Ljubljana
- Bohanec M (2015a) DEX: an expert system shell for multi-attribute decision making. <http://kt.ijs.si/MarkoBohanec/dex.html>. Accessed 21 May 2015
- Bohanec M (2015b) DEXi: a program for multi-attribute decision making. <http://kt.ijs.si/MarkoBohanec/dexi.html>. Accessed 21 May 2015
- Bohanec M, Aprile G, Constante M, Foti M, Trdin N (2014) A hierarchical multi-attribute model for bank reputational risk assessment. In: Phillips-Wren G, Carlsson S, Respício A (eds) *17th conference for IFIP WG8.3 DSS*, Paris, France. IOS Press, pp 92–103
- Bohanec M et al (2009) The co-extra decision support system: a model-based integration of project results. In: *Co-extra international conference*. France, Paris, pp 63–64
- Bohanec M et al (2008) A qualitative multi-attribute model for economic and ecological assessment of genetically modified crops. *Ecol Model* 215:247–261
- Bohanec M, Rajkovič V (1990) DEX: an expert system shell for decision support. *Sistemica* 1:145–157
- Bohanec M, Rajkovič V (1999) Multi-attribute decision modeling: industrial applications of DEX. *Informatika* 23:487–491

- Bohanec M, Rajkovič V, Bratko I, Zupan B, Žnidaršič M (2013) DEX methodology: three decades of qualitative multi-attribute modeling. *Informatica* 37:49–54
- Bohanec M, Trdin N (2014) Qualitative multi-attribute decision method DEX: theory and practice. In: 20th conference of the international federation of operational research societies, Barcelona Spain. p 239
- Bohanec M, Trdin N, Kontić B (2016) A qualitative multi-criteria modelling approach to the assessment of electric energy production technologies in Slovenia. *Cent Eur J Oper Res*. doi:[10.1007/s10100-016-0457-4](https://doi.org/10.1007/s10100-016-0457-4)
- Bohanec M, Žnidaršič M (2008) Supporting decisions about the introduction of genetically modified crops. In: Zaraté P, Belaud JP, Camilleri G, Ravat F (eds) Collaborative decision making: perspectives and challenges. *Frontiers in artificial intelligence and applications*, vol 176. IOS Press, Amsterdam, pp 404–415
- Boose JH, Bradshaw JM, Koszarek JL, Shema DB (1993) Knowledge acquisition techniques for group decision support. *Knowl Acquis* 5:405–448
- Bouyssou D, Marchant T, Pirlot M, Tsoukiàs A, Vincke P (2006) Evaluation and decision models with multiple criteria. Springer, New York
- Brans JP, Vincke P (1985) A preference ranking organisation method: the PROMETHEE method for MCDM. *Manag Sci* 31:647–656
- Cafilisch RE (1998) Monte Carlo and quasi-Monte Carlo methods. *Acta Numer* 7:1–49
- Clemen RT, Reilly T (2001) Making hard decisions with decisiontools. Duxbury/Thomson Learning, Pacific Grove
- Corrente S, Greco S, Słowiński R (2012) Multiple criteria hierarchy process in robust ordinal regression. *Decis Support Syst* 53(3):660–674
- Dembczyński K, Greco S, Słowiński R (2009) Rough set approach to multiple criteria classification with imprecise evaluations and assignments. *Eur J Oper Res* 198:626–636
- Durbach IN, Stewart TJ (2012) Modeling uncertainty in multi-criteria decision analysis. *Eur J Oper Res* 223:1–14
- Feller W (1968) An Introduction to probability theory and its applications, vol 1, 3rd edn. Wiley, New York
- Figueira J, Greco S, Ehrgott M (2005) Multiple criteria decision analysis: state of the art surveys. Springer, New York
- French S (1986) Decision theory: an introduction to the mathematics of rationality. Halsted Press, New York
- Gomes LFAM, Moshkovich HM, Torres A (2010) Marketing decisions in small business: how verbal decision analysis can help. *Int J Manag Decis Mak* 11:19–36
- Greco S, Matarazzo B, Słowiński R (2001) Rough sets theory for multicriteria decision analysis. *Eur J Oper Res* 129:1–47
- Greco S, Matarazzo B, Słowiński R (2002) Rough sets methodology for sorting problems in presence of multiple attributes and criteria. *Eur J Oper Res* 138:247–259
- Greco S, Matarazzo B, Słowiński R (2010) Dominance-based rough set approach to decision under uncertainty and time preference. *Ann Oper Res* 176:41–75
- Hadar J, Rusell WR (1969) Rules for ordering uncertain prospects. *Am Econ Rev* 59:25–34
- Hewitt E, Stromberg K (1965) Real and abstract analysis. Springer, Berlin
- Holt J et al (2013) Eliciting and combining decision criteria using a limited palette of utility functions and uncertainty distributions: illustrated by application to pest risk analysis. *Risk Anal* 34:4–16
- Ishizaka A, Nemery P (2013) Multi-criteria decision analysis: methods and software. Wiley, Chichester
- Jacquet-Lagrèze E, Siskos Y (1982) Assessing a set of additive utility functions for multicriteria decision making: the UTA method. *Eur J Oper Res* 10:151–164
- Jacquet-Lagrèze E, Siskos Y (2001) Preference disaggregation: 20 years of MCDA experience. *Eur J Oper Res* 130(2):233–245
- Kadziński M, Greco S, Słowiński R (2014) Robust ordinal regression for dominance-based rough set approach to multiple criteria sorting. *Inf Sci* 283:211–228
- Kahraman C (2008) Fuzzy multi-criteria decision making. In: Pardalos PM, Du D-Z (eds) Springer optimization and its applications, vol 16. Springer, New York
- Kontić B, Bohanec M, Trdin N, Kontić D, Zagorc-Kontić S, Matko M (2014) Comparative evaluation of various energy options using qualitative multi-attribute models. In: 20th conference of the international federation of operational research societies, Barcelona, Spain, p 239
- Kuzmanovski V, Trajanov A, Leprince F, Džeroski S, Debeljak M (2015) Modeling water outflow from tile-drained agricultural fields. *Sci Total Environ* 505:390–401

- Larichev OI (2001) Ranking multicriteria alternatives: the method ZAPROS III. *Eur J Oper Res* 131:550–558
- Larichev OI, Moshkovich HM (1994) An approach to ordinal classification problems. *Int Trans Oper Res* 1:375–385
- Larichev OI, Moshkovich HM (1995) ZAPROS-LM—a method and system for ordering multiattribute alternatives. *Eur J Oper Res* 82:503–521
- Larichev OI, Moshkovich HM (1997) Verbal decision analysis for unstructured problems. In: Herings JJ, Jackson MO, Kaneko M, Peters H, Peleg B, Puppe C, Roth AE, Schmeidler D, Thomson W, Vohra R, Wakker PP, Tijs SH (eds) *Theory and decision library C*, vol 17. Springer, US, New York
- Lavrač N, Džeroski S (1994) *Inductive logic programming: techniques and applications*. Ellis Horwood, New York
- Leben A, Kunstelj M, Bohanec M, Vintar M (2006) Evaluating public administration e-portals. *Inf Polity Dev e-Gov Cent East Eur* 11:207–225
- Mihelčić M, Bohanec M (2016) Approximating incompletely defined utility functions of qualitative multi-criteria modeling method DEX. *Cent Eur J Oper Res*. doi:[10.1007/s10100-016-0451-x](https://doi.org/10.1007/s10100-016-0451-x)
- Mileva-Boshkoska B, Bohanec M (2012) A method for ranking non-linear qualitative decision preferences using copulas. *Int J Decis Support Syst Technol* 4:42–58
- Moshkovich HM, Mechitov AI (2013) Verbal decision analysis: foundations and trends. *Adv Decis Sci* 2013:9
- Nagel SS (1993) *Computer-aided decision analysis: theory and applications*. Praeger, Westport
- Omero M, D'Ambrosio L, Pesenti R, Ukovich W (2005) Multiple-attribute decision support system based on fuzzy logic for performance assessment. *Eur J Oper Res* 160:710–725
- Rajkovič V, Bohanec M, Efstathiou J (1987) Ranking multiple options with DECMAX. In: Hagwood J, Humphreys P (eds) *Effective decision support systems*. Gower Technical Press, Aldershot, pp 49–60
- Roy B (1991) The outranking approach and the foundations of ELECTRE methods. *Theory Decis* 31:44–73
- Saaty TL (2008) Decision making with the analytic hierarchy process. *Int J Serv Sci* 1:83–98
- Saaty TL, Vargas LG (2012) Models, methods, concepts & applications of the analytic hierarchy process. In: Price CC, Zhu J, Hillier FS (eds) *International series in operations research & management science*, vol 175. Springer, US, New York
- Shachter RD, Peot MA (1992) Decision making using probabilistic inference methods. In: Dubois D, Wellman MP, D'Ambrosio B, Smets P (eds) *Eighth conference on uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc, Stanford
- Skinner DC (2009) *Introduction to decision analysis*, 3rd edn. Probabilistic Publishing, Gainesville
- Trdin N, Bohanec M (2012) Extending the multi-criteria decision making method DEX. In: Petelin D, Tavčar A, Kaluža B (eds) *4th Jožef Stefan international postgraduate school students conference*. Ljubljana, Slovenia, pp 182–187
- Trdin N, Bohanec M (2013) Relational multi-attribute models in DEX methodology. In: Ruiz F (ed) *22nd international conference on multiple criteria decision making*. Málaga, Spain, p 317
- Trdin N, Bohanec M (2014a) New generation platform for multi-criteria decision making with method DEX. In: Phillips-Wren G, Carlsson S, Burstein F, Respício A, Brézillon P (eds) *17th conference for IFIP WG8.3 DSS*, Paris, France. *DSS 2.0—Supporting decision making with new technologies*. IOS Press
- Trdin N, Bohanec M (2014b) Numerical relational multi-attribute models in qualitative multi-attribute method DEX. In: *20th conference of the international federation of operational research societies*, Barcelona, Spain, p 240
- Wang J, Zions S (2008) Negotiating wisely: considerations based on MCDM/MAUT. *Eur J Oper Res* 188:191–205
- Yang JB, Wang YM, Xu DL, Chin KS (2006) The evidential reasoning approach for MADA under both probabilistic and fuzzy uncertainties. *Eur J Oper Res* 171:309–343
- Zadeh LA (1965) Fuzzy sets. *Inf Control* 8:338–353
- Žnidaršič M, Bohanec M, Bratko I (2003) Categorization of numerical values for DEX hierarchical models. *Informatica* 27:405–409
- Žnidaršič M, Bohanec M, Lavrač N, Cestnik B (2009) Project self-evaluation methodology: the healththreats project case study. In: Bohanec M et al (eds) *Information Society—IS 2009*. Ljubljana, Slovenia, Institut Jožef Stefan

- Žnidaršič M, Bohanec M, Trdin N (2012) Qualitative assessment of data-mining workflows. In: Respício A, Burstein F (eds) *Fusing decision support systems into the fabric of the context*. *Frontiers in artificial intelligence and applications*. IOS Press, Amsterdam, pp 75–86
- Žnidaršič M, Bohanec M, Zupan B (2006a) Higher-order uncertainty approach to revision of probabilistic qualitative multi-attribute decision models. In: Adam F, Brézillon P, Carlsson S, Humphreys P (eds) *IFIP WG8.3 international conference on creativity and innovation in decision making and decision support*. Ludic Publishing, London
- Žnidaršič M, Bohanec M, Zupan B (2006b) proDEX—a DSS tool for environmental decision-making. *Environ Model Softw* 21:1514–1516
- Žnidaršič M, Bohanec M, Zupan B (2008) Modelling impacts of cropping systems: demands and solutions for DEX methodology. *Eur J Oper Res* 189:594–608