RICHER Decisions – A Decision Support Software Tool

Helsinki University of Technology
Systems Analysis Laboratory
Ville Brummer 51613L
Department of Engineering Physics and Mathematics
## Contents

Contents ............................................................................................................................................. 2

1 Introduction ........................................................................................................................................ 3

2 Multi-attribute Value Theory and Decision Support Software ...................................................... 4
   2.1 Value Tree Analysis .................................................................................................................. 4
   2.2 Examples of MCDA Methods ................................................................................................. 5
   2.3 Dominance Relations and Decision Rules .............................................................................. 6
   2.4 Examples of MCDA Software ............................................................................................... 7

3 The RICHER-Method ......................................................................................................................... 9
   3.1 Formulation ............................................................................................................................. 9
   3.2 Implementation of the RICHER Method ............................................................................... 12

4 Software Architecture ..................................................................................................................... 15
   4.1 Software Implementation ........................................................................................................ 15
   4.2 Classes Implementing Decision Variables .............................................................................. 16
   4.3 Session .................................................................................................................................... 16
   4.4 Session Handlers ..................................................................................................................... 17
   4.5 Architecture of the Graphical User Interface (GUI) ............................................................. 18
   4.6 XPress Objects ....................................................................................................................... 19

5 Graphical User Interface (GUI) ....................................................................................................... 21
   5.1 Common Considerations .......................................................................................................... 21
   5.2 Adding and Removing Objects ............................................................................................... 22
   5.3 Defining Incomplete Ordinal Information .............................................................................. 24
   5.4 Defining Linear Equalities and Inequalities .......................................................................... 25
   5.5 Visualizing Results .................................................................................................................. 27

6 Discussion ....................................................................................................................................... 28
1 Introduction

When making decisions, decision support methods and tools act a major role as a supporting element, especially if the decision maker (DM) finds difficulty to picture the whole system linked to the decision. One of the most common approaches is multi-attribute value theory (MAVT, Keeney and Raiffa 1976), where decision problem is approached by structuring it into attributes and alternatives. Nowadays, MAVT is a commonly used approach in multi-criteria decision making (MCDM) with wide application areas (see, e.g. Keefer et. al. 2004) and along with models of MAVT, decision maker (DM) can support and consider decisions with multidimensional and uncertain consequences. Due to incompleteness of present information many methods have been developed to support the elicitation. Furthermore, considering the uncertainty of effects of the decision, many methods have been developed to support the construction of decision recommendations based on the DM’s preferences and moreover software tools to support the usage of these methods.

This paper introduces RICHER Decisions, decision support software based on the RICHER method (Rank Inclusion in Criteria Hierarchies with Extended Rankings, Punkka and Salo 2005). The method is an extension to the RICH method (Salo and Punkka 2005). Whereas RICH allows a decision maker to use incomplete ordinal information in attribute weight elicitation, the RICHER method allows incomplete ordinal preference information also in evaluation of alternatives. Furthermore the incomplete ordinal information is formulated through linear constrains and thus the method allows the DM to define also other constrains in form of linear inequalities.

The paper is structured as follows. Section 2 introduces key concepts of MAVT and some related methods. Section 3 reviews the formulation of the RICHER method. Section 4 discusses software architecture, and Section 5 presents the graphical user interface. Section 6 concludes.
2 Multi-attribute Value Theory and Decision Support Software

2.1 Value Tree Analysis

Value tree analysis is a widely used method in multi-criteria decision making (see, e.g. Keefer et. al. 2004). In value tree analysis, the decision problem is approached by i) defining the relevant alternatives and attributes, denoted by \( A = \{a_1, \ldots, a_n\} \) and \( X = \{x_1, \ldots, x_m\} \), respectively, ii) specifying the performance of alternatives on the defined attributes and assessing them through scores, iii) measuring relative importance of attributes through weights and iv) computing the overall value for each alternative.

If certain terms are fulfilled (see Dyer and Sarin 1979), the DM’s preferences can be modeled through an additive value function

\[
V(x^i) = \sum_{k=1}^{n} v_k(x_k^i),
\]

(1)

where \( v_k(x_k^i) \) is the value of alternative \( x^i \) with regard to attribute \( a_k \) (i.e. score) and \( V(x^i) \) is overall value of the alternative \( x^i \). The most and least preferred achievement level, with regard to the \( k \)-th attribute, are denoted by \( x_k^* \) and \( x_k^0 \), and because the additive value function is unique up to positive affine transformations, we can choose

\[
v_k(x_k^0) = 0 \ \forall \ k \in \{1, \ldots, n\} \ \text{and} \ \sum_{k=1}^{n} v_k(x_k^*) = 1.
\]

Thus, the overall value of alternative \( x^i \) can also be written as
\[ V(x^i) = \sum_{k=1}^{n} \frac{v_k(x^*_k) - v_k(x^0_k)}{v_k(x^*_k) - v_k(x^0_k)} = \sum_{k=1}^{n} w_k v^N_k(x^i_k), \quad (2) \]

where \( w_k = v_k(x^*_k) - v_k(x^0_k) \) is the weight of the attribute \( a_k \) and \( v^N_k(x^i_k) \in [0,1] \) is the normalized score of alternative \( x^i \) with regard to attribute \( a_k \).

### 2.2 Examples of MCDA Methods

Even though the structure of the MAVT is quite straightforward, elicitation of the preferences can be difficult. Many reasons can be found, such as urgency of decision, lack of resources and presence of intangible attributes (see, e.g. Weber 1987) and in consequence, several methods for modeling incomplete information in value trees have been developed, where the model profiles not only preferences but also uncertainty of different alternatives’ consequences (see, e.g. Arbel 1989, Salo and Hämäläinen 1992, Park and Kim 1997).

Early contribution in modeling of incomplete information is a method where the DM defines statements of relative importance of weights in form \( w_i / w_j \in [a,b] \) (Arbel 1989). The PAIRS method (Salo and Hämäläinen 1992) processes such statements in hierarchical value trees and obtains overall value intervals and dominance relations by solving a series of hierarchically structured LPs (Linear Program).

Park and Kim (1997) model incomplete information on weights in form of linear inequalities and feasible regions. They define five different kinds of rankings: 1) weak ranking \( w_i \geq w_j \), 2) strict ranking \( w_i - w_j \geq \alpha_i \), 3) ranking with multiples \( w_i \geq \alpha w_j \), 4) interval form \( \alpha_i \leq w_i \leq \alpha_i + \varepsilon_i \) and 5) ranking of differences \( w_i - w_j \geq w_k - w_j, j \neq k \neq l \). Már mol et al. (1998) consider computational aspects of such statements and develop an algorithm for solving the corresponding feasible regions’ extreme points.
The PRIME method (Preference Ratios in Multi-Attribute Evaluation, Salo and Hämäläinen 2001) is designed to support analysis, where the DM can provide many kinds of incomplete information such as holistic comparisons between alternatives $V(x^i) \geq V(x^j)$ and ratios of value differences $L \leq \frac{|v_j(x_i) - v_j(x_j)|}{v_j(y_j) - v_j(y_j)} \leq U$.

The RICH method (Rank Inclusion in Criteria Hierarchies, Salo and Punkka 2005) allows the DM to specify incomplete ordinal information of the relative importance of attributes. The DM can make statements such as attributes $a_1$, $a_4$ and $a_5$ are the three most important and thus define ordinal preference information without taking a stance on their mutual orderings or the complete rank-ordering.

### 2.3 Dominance Relations and Decision Rules

If scores and weights are given as intervals, overall value of an alternative is also an interval. For example, consider a case with two attributes and two alternatives: If we define weights as intervals; $0.1 \leq w_1 \leq 0.9$, $0.1 \leq w_2 \leq 0.9$ and normalized scores as exact values; $v_1^N(x^1) = 0.3$, $v_2^N(x^1) = 0.7$, $v_1^N(x^2) = 0.2$ and $v_2^N(x^2) = 0.8$, the overall values are within intervals $V(x^1) \in [0.34, 0.66]$ $V(x^2) \in [0.26, 0.74]$ and alternative maximizing the value function is not unambiguous. In such cases we need other methods to determine which alternative should be recommended to the DM.

One way to model this problem is dominance structure (see, e.g. Hazen 1986) where mutual superiorities of alternatives can be defined by calculating pairwise dominance relations by minimizing value differences between alternatives’ values:

$$
\mu(x^k, x^l) = \min \sum_{j=1}^{n} (v_j(x^k_j) - v_j(x^l_j)). \quad (3)
$$
If pairwise bound \( \mu(x^k, x^l) \geq 0 \), alternative \( x^k \) is preferred to \( x^l \) and hence \( x^k \) dominates \( x^l \).

There are also several other decision rules that help the DM to decide which alternative to choose (see, e.g. Salo and Hämäläinen 2001). The DM can choose for example the alternative maximizing the maximum overall value (maximax) or the minimum overall value (maximin). One can also choose the alternative with maximum sum of maximum and minimum overall value (central value) or alternative minimizing the maximum difference between its overall value and the overall value of any other alternative (minimax regret). Formally, these rules can be represented as follows:

maximax: \[ \arg \max_{x^l} \left\{ \max_{x^i} V(x^i) \right\} \]
maximin: \[ \arg \max_{x^l} \left\{ \min_{x^i} V(x^i) \right\} \]
central value: \[ \arg \max_{x^l} \left\{ \max_{x^i} V(x^i) + \min_{x^i} V(x^i) \right\} \]
minimax regret: \[ \arg \min_{x^l} \left\{ \max_{x^i} \left( V(x^i) - V(x^l) \right) \right\} \]

### 2.4 Examples of MCDA Software

As we saw in the Sections 2.3 and 2.4 there are several decision support methods and decision rules to use with. Still, mathematical analysis does not necessary produce unambiguously comparable information and often decision making process with mathematical support methods is iterative and many methods and decision rules are used in paraller. To assist this work, software which helps the decision maker to perceive and control the whole process has been developed (see, e.g. Hämäläinen 2003).

One of the simplest models of decision support software is a framework, where the user can define normalized scores and attribute weights and gets the overall values of the alternatives as a result. An example of this kind of software is Web-HIPRE...
(www.hipre.hut.fi), where the user can create value trees and calculate overall values of alternatives (Mustajoki and Hämäläinen 2000). In Web-HIPRE, scores and weights are given as numerical point estimates which are solicited through methods such as SMART (Edwards 1977), SMARTER (Edwards and Barron 1994), SWING (von Winterfeldt and Edwards 1986) and AHP (Saaty 1980). Furthermore, Web-HIPRE provides several ways to visualize the results.

There are also other software tools that are designed to use one certain method. PRIME-Decisions (Gustafsson et. al. 2001, www.decisionarium.hut.fi) allows the DM to give preferences as intervals and offers visual presentation of value intervals and dominance structures. VIP Analysis (Dias and Climaco 2000) is a decision support software tool that allows the DM to use incomplete information about attribute weights in form of linear inequalities. Smart Swaps (www.smart-swaps.hut.fi) software supports the Even Swaps method (Hammond et. al. 1998, 1999). RICH decisions (www.rich.hut.fi) supports the use of the RICH method (Salo and Punkka 2005), where the DM can specify incomplete ordinal information about the relative importance of attributes and intervals to scores and offers visual presentation of value intervals and dominance structure.
3 The RICHER-Method

3.1 Formulation

One of the forms of incomplete information is ordinal preference information, where the DM can make statements such as “attribute $a_3$ is more important than $a_1$” or “alternative $x^3$ is better than $x^4$” or “with regard to attribute $a_3$, alternative $x^1$ is better than $x^2$”. Formally these clauses mean that if the DM prefers $x^i$ to $x^j$ with regard to attributes $A$, then $\sum_{a \in A} v_k(x^i) \geq \sum_{a \in A} v_k(x^j)$.

If we define $m' = |X'|$ (cardinality of $X'$), following Punkka and Salo (2005) rank-orderings can be denoted by convention $r(X') = \{r_1, \ldots, r_m\}$ where $r_k = r(x_j; X')$ is the ranking of the alternative with k-th smallest index in $X'$. Because rank-orderings can be defined with regard to all attributes or just a few, we specify rank-orderings $r_A(X')$, so that they are defined with regard to attributes $A'$.

Punkka and Salo (2005) model incomplete ordinal information through statements that associate subsets of alternatives (denoted by $I$) to subsets of rankings (denoted by $J$). For example, if we say that “considering attributes $A' = \{a_1, a_3\}$, alternatives $x^1$ and $x^3$ are the two most preferred ones among alternatives $X' = \{x^1, x^2, x^3, x^4\}$”, we get sets $I_{A',X'} = \{x^1, x^3\}$ and $J_{A',X'} = \{1, 2\}$. By these statements we can define the set of compatible rank orderings i.e. all the rank-orderings that fulfill these statements. In this case we get four compatible rank-orderings: $(1,3,2,4)$, $(1,4,2,3)$, $(2,3,1,4)$ and $(2,4,1,3)$.

For defining compatible rank-orderings, $I$ and $J$ need not to be equal size. Punkka and Salo (2005) interpret the statements as follows:
**Definition 1** (Punkka and Salo 2005) Let $I \subset X' \subseteq X$ and $J \subset \{1,\ldots,m'\}$ where $m'=|X'|$ and $|I|, |J| > 0$. The set of compatible rank-orderings is

$$R_{X'}(I,J) = \begin{cases} \{ r \in R(X') \mid r^{-1}(j) \in I \ \forall \ j \in J \} & \text{if } |I| \geq |J| \\ \{ r \in R(X') \mid r(x^k) \in J \ \forall \ x^k \in I \} & \text{if } |I| < |J| \end{cases}$$

where $R(X')$ is the set of all rank-orderings defined on $X'$.

Rank-orderings imply linear constraints to the feasible set of scores $S$ in a following way:

$$S(r_x(X')) = \{ s \in S_0 \mid v_x(x^i) \geq v_x(x^j) \text{ if } r_x(x^i; X') < r_x(x^j; X'), x^i, x^j \in X' \},$$

where $S_0$ is the set of all the possible values of the scores.

Punkka and Salo (2005) formulate the feasible set $S(I,J)$ corresponding to statement $(I, J)$ through linear constraints. In formulation, we use $v(x^i)$ to represent a value of $x^i$, which can be subjected to any attribute set by setting $v(x^i) = v_x(x^i) \in [0, \sum_{a \in A} w_a]$.

Now, let us consider a statement where there are more alternatives than rankings ($|I| > |J| > 0$). Set of feasible scores can be defined by linear constraints (Punkka and Salo 2005):

$$z_j \leq v(x^i) + (1 - y_j(x^i))M \quad \forall \ j \in \{1,\ldots,m'-1\}, \forall x^i \in X'$$

$$v(x^i) \leq z_j + y_j(x^i)M \quad \forall \ j \in \{1,\ldots,m'-1\}, \forall x^i \in X'$$

$$y_{j-1}(x^i) \leq y_j(x^i) \quad \forall \ j \in \{2,m'-1\}, \forall x^i \in X'$$

$$\sum_{x^i \in X'} y_j(x^i) = j \quad \forall \ j \in \{1,m'-1\}$$

$$\sum_{x^i \in I} (y_j(x^i) - y_{j-1}(x^i)) = 1 \quad \forall \ j \in J,$$
where $M \gg 0$.

In the formulation, $z_j \geq 0$, $j \in \{1, \ldots, m\}$, is a “milestone” variable which defines an upper bound for the value of the alternative whose ranking exceeds $j$ and $y_j(x^i)$, $j \in \{1, \ldots, m-1\}$, is a binary variable indicating whether $v(x^i)$ is greater than or equal to milestone $z_j$ or not.

On the other hand, if $|J| \geq |I|$, according to Punkka and Salo (2005), $S(I, J) = S(J)$.

Constraints define the feasible set $S(I, J)$, but for improving computational properties, Punkka and Salo (2005) redefine model through sequential sets of rankings.

**Definition 2** (Punkka and Salo 2005) Let $J \subset \{1,2, \ldots, m\}$. Set $J$ is sequential when $a, b \in J, a \leq b$ implies that $k \in J$ for any integer $k$ such that $a \leq k \leq b$. Otherwise, $J$ is non-sequential.

For sequential ranking sets we can define the feasible set $S(I, J)$ as follows (Punkka and Salo 2005):

\[
\begin{align*}
z_j & \leq v(x^i) + (1 - y_j(x^i))M & \forall j \in \{j^{-}, j^{+}\}, \forall x^i \in X' \quad (11) \\
v(x^i) & \leq z_j + y_j(x^i)M & \forall j \in \{j^{-}, j^{+}\}, \forall x^i \in X' \quad (12) \\
\sum_{x^i \in X'} y_j(x^i) & = j & \forall j \in \{j^{-}, j^{+}\} \quad (13) \\
\sum_{x^i \in X'} (y_{j^{-}}(x^i) - y_{j^{-1}}(x^i)) & = |J| \quad (14) \\
y_{j^{-1}}(x^i) & \leq y_j(x^i) & \forall x^i \in X', \quad (15)
\end{align*}
\]
where $j^- = \min J$ and $j^+ = \max J$

If $J$ is non-sequential, it can be divided into sequential parts $J_i$ so that $J = \bigcup_i J_i$ and

$\bigcap_i J_i = \emptyset$ and the feasible set can be defined as intersection

$$S(I, J) = \bigcap_i S(I, J_i).$$

(16)

More generally, if we have more than one statement $R(I_i, J_i)$, the feasible set can be defined as intersection

$$S(I_1, \ldots, I_k, J_1, \ldots, J_k) = \bigcap_{i=1}^k S(I_i, J_i).$$

(17)

### 3.2 Implementation of the RICHER Method

One advantage of the RICHER method is that it can be combined with all the other Preference Programming methods that capture incomplete information through linear inequalities (e.g. Park and Kim 1997). When implementing the RICHER method as software, it is useful to build the software so that in addition to incomplete ordinal information, the DM can give preference statements also in form of other linear inequalities.

Normalized scores and weights can be given as intervals, denoted by $VI_v(x^i_k) \subseteq [0,1]$ and $VI_w(w_k) \subseteq [0,1]$. From these intervals, we get inequalities
\[
\begin{align*}
    w_k &\geq \min \langle I \rangle \big( w_k \big) \quad (18) \\
    w_k &\leq \max \langle I \rangle \big( w_k \big) \quad (19) \\
    v^N_k \big( x^i_k \big) & = \frac{v_k \big( x^i_k \big)}{w_k} \geq \min \langle I \rangle \big( v^N_k \big( x^i_k \big) \big) \Rightarrow v_k \big( x^i_k \big) \geq \min \langle I \rangle \big( v^N_k \big( x^i_k \big) \big) w_k \quad (20) \\
    v^N_k \big( x^i_k \big) & = \frac{v_k \big( x^i_k \big)}{w_k} \leq \max \langle I \rangle \big( v^N_k \big( x^i_k \big) \big) \Rightarrow v_k \big( x^i_k \big) \leq \max \langle I \rangle \big( v^N_k \big( x^i_k \big) \big) w_k \quad (21)
\end{align*}
\]

Incomplete ordinal information on attributes is given as clauses denoted by \( R_n(I,J) \) and incomplete ordinal information on alternatives as clauses denoted by \( R_v(A,I,J) \), where \( I \) signifies the set of variables (alternatives or attributes) associated to set of rankings \( J \) and when handling alternatives, \( A \) signifies the set of attributes.

Equalities and inequalities for weights are given in form

\[
\sum_{i=1}^{i} c_i w_i \leq b, \quad c_i, b \in \mathbb{R} \quad (22)
\]

and especially through weight ratios

\[
a \leq w_i / w_j \leq b, \quad a, b \in \mathbb{R}_+ . \quad (23)
\]

Equation (23) implements two inequalities \( bw_j - w_i \geq 0 \) and \( aw_j - w_i \leq 0 \) and therefore all the linear inequalities are essentially in the form of (22).

Thus, we get constraints (18)-(21) from value intervals, constraints (11)-(15) from clauses implementing incomplete ordinal information and some other constraints in form of (22). With these constraints we can define the feasible set of the scores and find the maximum and minimum values for every alternative. By minimizing (3) we also get the
pairwise dominance relations. Finally, these results can be used to provide decision recommendations based the different decision rules.
4 Software Architecture

4.1 Software Implementation

The software is programmed using Java language and solutions are computed with XPress-MP solver (www.dashoptimization.com). Java objects and solver are integrated by using classes of the BCL Java interface (www.dashoptimization.com) and the graphical user interface (GUI) is implemented mainly by using javax.swing package (http://java.sun.com/j2se/1.3/docs/api/javax/swing/package-summary.html).

![Software Architecture Diagram](image)

Figure 1 Software architecture

The architecture of the software can be divided into four independent parts: GUI, session, session handlers and XPress objects (see Fig. 1). GUI includes all graphic elements and implements all functionality that controls the software. Session is the object where all information is stored and GUI remolds session by using static functions of the session handlers. XPress objects handle computation.

On the other hand, one can find a bit different aspect on the architecture. In all parts of the software i.e. GUI, session, session handlers and XPress Objects, there are six
elements that create a framework: i) attributes, ii) alternatives, iii) rankings for attributes, iv) rankings for alternatives implementing incomplete ordinal information, v) linear constraints and iv) weight ratios implementing linear constraints of weights and every part of the software consists of corresponding classes that individually handle these elements.

4.2 Classes Implementing Decision Variables

There are six classes that implement objects corresponding with a decision problem: Attribute, Alternative, AlterRank, AttrRank, Equation and RelEquation. In this work we call them decision objects. They store the information that is needed for modeling the decision problem.

4.3 Session

All information is stored in session. Session includes six java.util.HashMap objects and six java.util.ArrayList objects that are used through java.util.Collection interface. HashMaps are used as stores for decision objects and Collections store Hashmap keys. Thus, behind the key in corresponding Hashmap decision objects are in a form that decision classes implement them and iteration through objects can be carried out by iterating through Collection. Beside these basic objects, session includes also HashMap for storing pairwise dominance relations between alternatives and object IdCreator that produces unique keys for handling decision objects. Figure 2 illustrates the objects in session.
This framework is useful because decision objects are not independent and there are a lot of cross-linkages and references between them. For example incomplete ordinal information consists of attributes and alternatives and alternatives’ scores are dependent of attribute weights. So when referring from one decision object to another we do not have to convey the whole object but we can just refer to the identifying key.

This same structure could be implemented also as tables and indices or by creating structures that imitate value trees. However, using Collections and HashMaps as a root makes the software more stable, reliable and faster. By this division (Alternative, Attribute, AttrRank, etc.) session corresponds with GUI, programming stays simple and information is easy to save for later use.

### 4.4 Session Handlers

Session handlers are classes with static functions. Through them, GUI and Xpress objects can handle session.
Every decision object has its own handler and they include two different kind of functions; setters and getters. Setters remold session according to function parameters and return a boolean variable describing if execution succeed or not. Getters return determined values, for example attributes’ score limits or alternatives’ names.

Basic forms of getter and setter functions are described below:

```java
public static boolean setValue(Session session, String key, String value)
public static String getValue(Session session, String key)
```

At the previous chapter we noticed that decision objects depend on each other. So remolding one decision object can make another decision object infeasible. For example deleting an alternative may cause infeasibility in rankings and deleting an attribute may pose weight ratios and linear inequalities infeasible. These functions not only remold one certain object but also remold all the other objects so that session stays feasible.

### 4.5 Architecture of the Graphical User Interface (GUI)

GUI architecture follows the same formula as session and session handlers; every decision object has corresponding classes implementing graphical objects. GUI is built on a base panel that includes four twin-panels organized by tabs. Every twin-panel consists of two panels corresponding with decision objects.

Beside these graphical objects GUI includes objects that handle functionality of the program. These functions are stored in the GUI handlers and Figure 3 presents a diagram of attribute panel’s architectural implementation.
Thus, when the user does something, he communicates with the panel which sends information to the GUI handler. The GUI handler remolds panel and session and if necessary sends information to other GUI handlers so that they can remold their corresponding panels accordant with the new session.

## 4.6 XPress Objects

XPress objects operate as an interface between XPress-MP solver and session. The kernel of this ensemble is class XPressRicherProblem which includes functions that transfer Session to a mathematical model and calculate the minimum and the maximum overall values and the pairwise dominance relations between alternatives. All the other three classes are supporting these functions by creating and naming variables and constraints.
Figure 4 XPress Handler architecture

Figure 4 illustrates the interface between GUI and XPress-MP solver that is implemented by one object. The class implementing the object is build by using BCL Java library.
5 Graphical User Interface (GUI)

5.1 Common Considerations

Graphical user interface (GUI) consists of four different panels that lie on a tapped pane. The first panel (from left) is for handling attributes and alternatives. In the second panel the user can give incomplete ordinal information and the third panel is for defining linear constraints for attribute weights. In the fourth panel the user can view the results.

Every panel of the GUI includes buttons “OK” and “Delete” and mouse is used to select the object the user wants to handle. Clicking “OK” means that the user wants to save the changes he has made and clicking “Delete” removes the object that has been chosen. Pressing “Enter” on the keyboard means same as “OK”. Clicking button “Calculate” solves the defined problem.

Every list (attributes, alternatives, ranks, etc.) includes object “{new}” which has to be chosen in order to create a new object. If the user wants to edit objects, e.g. rename attributes or redefine incomplete ordinal preference information, he has to choose the object by mouse and click “OK” or press “Enter” afterwards. By pressing “Ctrl” or “Shift” simultaneously as using mouse, the user can choose more than one objects at once. This is very useful when creating or rewriting ordinal statements.

The system is integrated so that if the user changes the list of attributes or alternatives, all the components, e.g. rankings, weight ratios and constraints, that are not feasible anymore, are deleted. Deleting the unfeasible components is explained in detail in the section 5.2.
5.2 Adding and Removing Objects

In the panel “Alternatives/Attributes”, the user can add and remove attributes and alternatives. The left pane is for handling attributes and the right pane is for alternatives. When creating a new attribute or alternative, the user chooses “{new}”. The program offers a default name, but the object can be defined uniquely by writing a name on the box upper side the list. When renaming the object afterwards, the user can choose the object from the list and write a new name to the same box.

At the combo box under the attribute list, the user can define a lower limit to the attribute weights. When defining limits to the alternatives’ normalized scores, first the user has to choose an attribute and an alternative and then define the limits by using the scroll box under the alternative list or just by writing the limits to the boxes next to the scroll box.

Figure 5 Alternatives and attributes
Editing (i.e. removing and adding alternatives and attributes) the set of variables may cause infeasibility in other segments of the model. For example deleting an attribute makes weight ratios including the particular attribute inaccessible and deleting an alternative may cause disharmony in clauses implementing ordinal preference information. In order to maintain the model feasible, the following actions are executed whenever editing set of variables.

Adding and deleting alternatives:
- Clauses implementing ordinal preference information of alternatives are removed.

Adding and deleting attributes:
- Clauses implementing ordinal preference information of attributes are removed.
- Clauses implementing ordinal preference information of alternatives with regard to the deleted attribute are removed
- Constraints and weight ratios including the deleted attribute are removed.
5.3 Defining Incomplete Ordinal Information

Figure 6 Rankings

In the panel “Ranks” the user can define incomplete ordinal information on attributes and alternatives. The left pane is for attributes and the right pane for alternatives. By choosing attributes at the middle left and rankings at the middle right the user can create and rewrite statements. By pressing “Ctrl” or “Shift” simultaneously as using the mouse, the user can choose more than one attribute and ranking and thus define statements with several attributes and rankings. When creating a new statement, the user chooses “{new}” and if the user wants to rewrite a statement, it has to be chosen on the upper side the pane.

Defining preference information for alternatives can be done almost in the same way as for attributes, but first the user has to choose attributes that he regards to. Holistic
identifications e.g. “alternative $x_1$ is the most preferred and alternative $x_2$ is the second most preferred” can be defined by choosing “{all}” from the attribute list on the upper pane. By choosing one specific attribute from the same list, the user can define statements such as “with regard to attribute $a_1$, alternative $x_2$ is the most preferred” (cf. Figure 2).

### 5.4 Defining Linear Equalities and Inequalities

![Figure 7 Linear equations](image)

In the panel “Equations” the user can define linear constraints on the attribute weights. The left pane is for handling linear equalities and inequalities such as $2w_2 + 3w_3 \geq 0.5$ (see Figure 7) and the pane on the right is for handling weight ratios for attributes such as $1 \leq \frac{w_i}{w_5} \leq 3$. 
On the left pane inequalities are created as follows. In the middle of the pane there is a box for the equation. On the left side of the pane the user can define the left side of the equation by writing attributes inside quotes and adding signs and coefficients between. For example the formula on Figure 7 can be formulated as follows: “2.0*"attribute2"+3.0*"attribute5"”. By clicking the attribute on the list underneath the text box, the user can select attributes without writing and add signs and coefficient afterwards. In the middle of the line the user can define the sign of the equation and the right side of the equation is defined on the right.

The right side of the panel is for defining weight ratios. At the bottom of the pane there are two lists where the user can define attributes to be compared. Lower and upper limits for the relation can be defined in the text boxes on the middle. Pressing enter or clicking “OK” saves the equation.

If the equation is not well-defined, for example there does not exist an attribute named the way that user has defined, clicking OK or pressing enter just empties all the boxes and the equation has to be rewritten.
5.5 Visualizing Results

Results are displayed in the “Results” panel, when the model has been calculated at least once. If the user has made changes, he has to calculate the model again (click “Calculate”). Otherwise, the results shown at the panel remain same as before changes. On the left of the panel the user can see overall value intervals and the pairwise dominance relations are viewed on the right. By clicking mouse on the result pane, the corresponding results are viewed on a bigger screen.
6 Discussion

Compared to previous methods, the RICHER method has two advantages; it enables use of incomplete ordinal information on both attributes’ importance and alternatives’ performance and second it can be combined with methods where information on weights and/or scores is given as linear inequalities. Both of these aspects have had a significant role in the implementation of the software. Having ability to define linear inequalities and value intervals makes it possible to use results also from other methods and software as input and moreover establishing the results as value intervals enables to use the results as input for other methods and software.

The greatest challenge in building useful software was to maintain the program as simple as possible and understandable for different type of users. One of the main reasons for implementing the method as software was to make the use of the method so simple, that it can be used, even if the user has no strong understanding of the actual theory behind. Minimum functionalities were implemented; i) handling attributes and alternatives, ii) defining relative ordinal information on attributes and alternatives and iii) defining linear inequalities for attributes. In addition to these basic functionalities, weight ratios were considered as a remarkably essential part of the weight elicitation process, and the only overlapping functionality of the program is the ability to define weight ratios directly as ratios, not through basic form linear inequalities.

Using a tapped pane as a bed for the actual working panels kept the visual face of the program simple and easy to use. Having separate panels for different type of elicitation procedures made it easy to perceive the different parts of the elicitation process. On the other hand, the user can remold the model only on one panel at a time and it may be difficult to see the cross-linking between the different panels and piece together the elicitation process in aggregate. For example removing an attribute affects to the whole decision problem and deleting an alternative may cause infeasibility in the previously defined information.
To sum up, the program is simple, easy to use and it implements the essential functionalities of the RICHER method. The program is stable and the model can be remolded iteratively. Due to the character of incomplete ordinal preference information, removing variables may cause major changes in the model and recommendation for the use of the program is that before defining relations between variables it is useful to try to define all the essential variables at the same time.
References


