GENERATING HASSE TREES OF FUZZY PREORDER CLOSURES: AN ALGORITHMIC APPROACH

Helga Naessens, Bernard De Baets, Hans De Meyer
Department of Applied Mathematics and Computer Science
University of Ghent
Krijgslaan 281 (S9), B-9000 Gent, Belgium
{Helga.Naessens,Bernard.DeBaets,Hans.DeMeyer}@rug.ac.be

Summary

A top-down algorithm for generating the Hasse tree of the fuzzy preorder closure of an arbitrary reflexive binary fuzzy relation is presented.

Keywords: fuzzy preorder, transitive closure, Hasse tree.

1 INTRODUCTION

Mathematical theories are pervaded with the use of partial orders, and more general, preorders (also called quasi-orders), i.e. reflexive and transitive binary relations. Preorders also play an important role in applications, for instance as preference relations in multicriteria decision aid techniques. The observation that classical relations do not allow to express partial or graded relationships has led to the introduction of fuzzy relations. In preference modelling, for instance, fuzzy relations are used to express graded preferences, indifferences and incomparabilities [4, 10].

In this paper, we will consider this decision-theoretic framework in which fuzzy preorders, i.e. reflexive and transitive binary fuzzy relations, are used as large preference relations. An important issue is that of the transitivity of the large preference relation. Transitivity is not always ensured (or even not desired) but can always be enforced by considering the transitive closure, which in this case is called fuzzy preorder closure.

Fuzzy preorders admit a nice graphical representation by means of a Hasse tree, i.e. a partition tree with Hasse diagrams drawn on top of it. Such a Hasse tree forms a suitable basis for the interpretation of preferential relationships in the set of alternatives. For more details about the exploitation of these Hasse trees, we refer to [3, 4]. Applications can be found in various disciplines such as cognitive psychology [1], finance [6], nuclear engineering [9] and geography [5].

In this paper, we focus on the problem of generating the Hasse tree. We present a top-down algorithm that allows to produce the Hasse tree level by level for decreasing degrees of confidence. Another advantage of the algorithm is its applicability for any binary reflexive fuzzy relation, in the sense that it produces at the same time the fuzzy preorder closure.

2 BASIC NOTIONS

We first recall some definitions that will be used throughout. A binary fuzzy relation R on a universe X is called a fuzzy preorder if it is reflexive and transitive, i.e. for any (x, y, z) in X^3 it holds that $R(x, x) = 1$ and

$$\min(R(x, y), R(y, z)) \leq R(x, z).$$

For α in $[0, 1]$, the α-cut R_α of a binary fuzzy relation R on X is the crisp binary relation on X defined by

$$(x, y) \in R_\alpha \Leftrightarrow R(x, y) \geq \alpha\,.$$

An important theorem states that R is a fuzzy preorder if and only if every α-cut R_α is a preorder [11]. This implies that for each α, R_α induces an equivalence relation E_α on X and a partial order $\leq_\alpha$ on the quotient set X/E_α. Formally,

$$(x_i, x_j) \in E_\alpha \Leftrightarrow (x_i, x_j) \in R_\alpha \wedge (x_j, x_i) \in R_\alpha\,,$$

and

$$[x_i]_\alpha \leq_\alpha [x_j]_\alpha \Leftrightarrow (x_i, x_j) \in R_\alpha \wedge (x_j, x_i) \notin R_\alpha\,,$$

where $[x_i]_\alpha$ and $[x_j]_\alpha$ are the equivalence classes of x_i and x_j w.r.t. E_α. It is clear that with decreasing α, the equivalence classes tend to merge together, or, in decision-theoretic terms: *"incomparability disappears at the cost of increasing indifference"* [3].

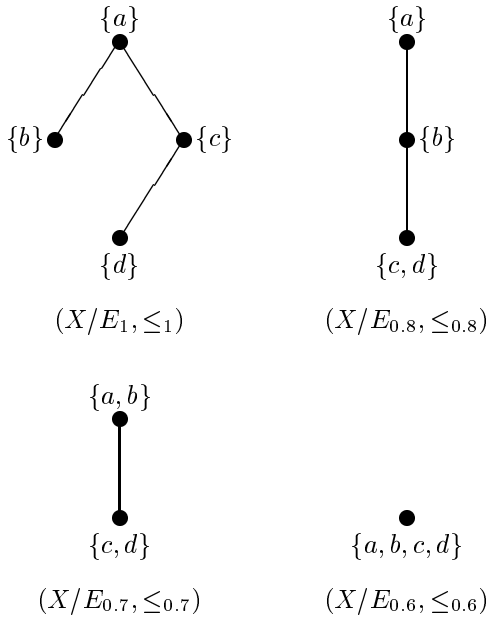

Figure 1: Hasse diagrams

As an illustration, consider the fuzzy preorder R on $X = \{a, b, c, d\}$ given by:

R	a	b	c	d
a	1	0.7	0.6	0.6
b	1	1	0.6	0.6
c	1	0.8	1	0.8
d	1	0.8	1	1

(1)

The Hasse diagrams of the four corresponding partial orders are depicted in Figure 1. These diagrams can be summarized in the Hasse tree drawn in Figure 2. In this tree, the broken lines in each level (identified by the α-cut R_α) represent the arcs of the Hasse diagram of $(X/E_\alpha, \leq_\alpha)$, while the nodes represent the equivalence classes of E_α.

For a non-transitive reflexive binary fuzzy relation R, the Hasse tree is drawn from its fuzzy preorder closure $\hat{R}$, which is defined as the smallest fuzzy preorder containing R, i.e. a fuzzy preorder $\hat{R}$ such that any fuzzy preorder E satisfying $R \subseteq E \subseteq \hat{R}$ coincides with $\hat{R}$ [2].

3 GENERATING HASSE TREES

In this section, we present a novel algorithm (Algorithm 1) for generating the Hasse tree of the fuzzy preorder closure of an arbitrary reflexive binary fuzzy relation R. A reflexive binary fuzzy relation R on X can be diagrammed by a *weighted* reflexive directed graph, whereby the vertices represent the elements of X and whereby the weights of the arcs are the corresponding values of R. Then, a reflexive binary fuzzy relation is

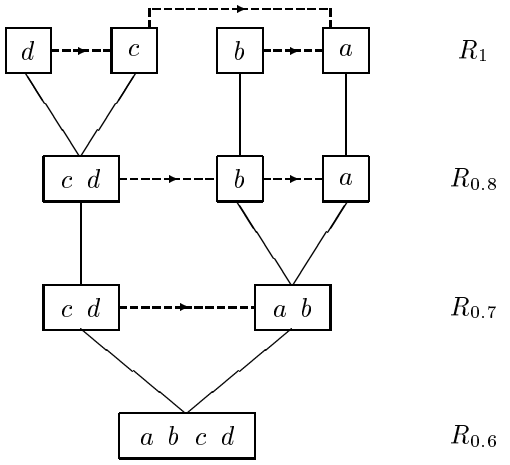

Figure 2: Hasse tree for R

a fuzzy preorder if and only if all the triangular subgraphs of its corresponding graph are transitive.

Algorithm 1 is inspired on a new algorithm we have developed for the computation of the T-transitive closure of a proximity relation, whereby T can be any triangular norm [7, 8]. It acts on a universe X with cardinality n, the elements of which are simply referred to as 1, 2, ..., n.

On execution of Algorithm 1 the arcs of the directed graph are visited in descending order of their actual weights (double repeat loop: an outer loop running over the different weights in the graph and an inner loop running over all the arcs with the same weight) and used as pivot, whereby it is taken into account that weights can change during execution of the algorithm. At the stage whereby arc (i, j) is the pivot, all non-transitive triangles containing arc (i, j) (for loop) and in which w_{ij} is the maximum weight (if tests) are made transitive by raising the smallest weight in the triangle.

Next, we add arc (i, j) to the set NewArcs. Then the algorithm checks if the class containing element i and the class containing element j merge. This is the case when arc (j, i) has previously been used as pivot. If the two classes merge, we add all the elements of the class containing j to the class containing i and we update the sets NewArcs and OldArcs. This means that we add arc (i, k), for all $k \in S \setminus \{i, j\}$, to NewArcs if OldArcs or NewArcs contains (j, k), but none of them contains (i, k). Analogously, we add arc (k, i) to NewArcs, if required. Then we remove all the arcs through vertex j from OldArcs and NewArcs to prevent them from being selected as pivot later on. Next, vertex j is removed from the set S, which will contain, upon termination of the inner loop, the representatives of the classes at the corresponding level.

After having performed these actions for pivots with the same weight (inner repeat loop), Algorithm 1 removes redundant arcs from the sets NewArcs and OldArcs, which requires two steps. In both steps, we make use of the fact that an arc (i, j) may be removed if there exists an indirect path from i to j in OldArcs $\cup$ NewArcs, which can be shown to be equivalent with $w_{ij} \geq \alpha$.

Algorithm 1 Generate the Hasse tree of the fuzzy preorder closure $\hat{R}$ of a fuzzy relation R

Input :
Vertex Set $= \{1, 2, \ldots, n\}$;
Weight Set $= \{w_{ij} = R(i, j) \mid \forall\, i, j \in$ Vertex Set$\}$;
Output :
Description of Hasse tree of $\hat{R}$;

begin
  $V := \{(i, j) \mid \forall\, i \neq j \in$ Vertex Set$\}$;
  $S := \{1, 2, \ldots, n\}$;
  for $k := 1$ **to** n **do** class$[k] := \{k\}$;
  OldArcs $:= \emptyset$;
  repeat
    Select an arc $(i, j) \in V$ such that
      $w_{ij} \geq w_{lm}\ \forall (l, m) \in V$;
    level $:= w_{ij}$;
    NewArcs $:= \emptyset$;
    repeat
      for $k := 1$ **to** n **do**
        if $k \in S$ **and** $w_{jk} \leq w_{ij}$ **and** $w_{ik} < w_{jk}$ **then**
          $w_{ik} := w_{jk}$;
        if $k \in S$ **and** $w_{ki} \leq w_{ij}$ **and** $w_{kj} < w_{ki}$ **then**
          $w_{kj} := w_{ki}$
      endfor;
      NewArcs $:=$ NewArcs $\cup \{(i, j)\}$;
      $V := V \setminus \{(i, j)\}$;
      if class$[i]$ and class$[j]$ merge **then**
        class$[i]$:= class$[i] \cup$ class$[j]$;
        class$[j] := \emptyset$;
        Update NewArcs and OldArcs;
        $V := V \setminus \{(j, 1), \ldots, (j, n), (1, j), \ldots, (n, j)\}$;
        $S := S \setminus \{j\}$
      endif;
      Select an arc $(i, j) \in V$ such that
        $w_{ij} \geq w_{lm}\ \forall (l, m) \in V$
    until $w_{ij} \neq$ level;
    Remove redundant arcs from NewArcs;
    Remove redundant arcs from OldArcs;
    OldArcs := OldArcs $\cup$ NewArcs;
    Write level, classes and arcs of OldArcs
  until $V = \emptyset$
end

The first step (Procedure 1) removes redundant arcs from the set NewArcs, for which it investigates in pairs the arcs of NewArcs. Suppose that arcs (i, j) and (k, l) are selected. Then, arc (i, j) may be removed if $w_{ik} \geq \alpha$ and $w_{lj} \geq \alpha$, since in that case there exists a path from i to j, containing arc (k, l).

Procedure 1 Remove redundant arcs from NewArcs

begin
  $X :=$ NewArcs;
  repeat
    Select an arc $(i, j) \in X$;
    $Y :=$ NewArcs $\setminus \{(i, j)\}$;
    repeat
      Select an arc $(k, l) \in Y$;
      if $w_{ik} \geq \alpha$ **and** $w_{lj} \geq \alpha$ **then**
        NewArcs := NewArcs $\setminus \{(i, j)\}$;
      $Y := Y \setminus \{(k, l)\}$
    until $Y = \emptyset$;
    $X := X \setminus \{(i, j)\}$
  until $X = \emptyset$
end

The second step (Procedure 2) removes redundant arcs from the set OldArcs, for which it investigates in pairs the arcs of OldArcs and NewArcs. Suppose that arcs (i, j) from OldArcs and (k, l) from NewArcs are selected. Then, arc (i, j) may be removed if $w_{ik} \geq \alpha$ and $w_{lj} \geq \alpha$, since in that case there again exists a path from i to j, containing arc (k, l).

Procedure 2 Remove redundant arcs from OldArcs

begin
  $X :=$ OldArcs;
  repeat
    Select an arc $(i, j) \in X$;
    $Y :=$ NewArcs;
    repeat
      Select an arc $(k, l) \in Y$;
      if $w_{ik} \geq \alpha$ **and** $w_{lj} \geq \alpha$ **then**
        OldArcs := OldArcs $\setminus \{(i, j)\}$;
      $Y := Y \setminus \{(k, l)\}$
    until $Y = \emptyset$;
    $X := X \setminus \{(i, j)\}$
  until $X = \emptyset$
end

Next, Algorithm 1 adds the arcs of NewArcs to the set OldArcs. As mentioned above, the elements of S are the representatives of the classes at the corresponding level, and the graph $(S,$ OldArcs$)$ represents the Hasse diagram of this level. Finally, we write out the level, the classes and the arcs of OldArcs.

4 EXAMPLE

To illustrate our algorithm, let us consider the fuzzy relation S given by:

S	a	b	c	d
a	1	0.7	0.4	0.5
b	1	1	0.6	0.4
c	1	0.8	1	0.8
d	1	0.7	1	1

Obviously, S is not transitive since for example $\min(S(a,b), S(b,c)) > S(a,c)$. The fuzzy preorder closure of S is nothing else but the fuzzy relation R (1). If we run Algorithm 1 on the fuzzy relation S, the following output is generated:

```
Level 1.0 :
Class 1 : a
Class 2 : b
Class 3 : c
Class 4 : d
Arc from Class 2 to Class 1
Arc from Class 3 to Class 1
Arc from Class 4 to Class 3

Level 0.8 :
Class 1 : a
Class 2 : b
Class 3 : c d
Arc from Class 2 to Class 1
Arc from Class 3 to Class 2

Level 0.7 :
Class 1 : a b
Class 2 : c d
Arc from Class 2 to Class 1

Level 0.6 :
Class 1 : a b c d
```

As expected, this is a description of the Hasse tree shown in Figure 2.

Acknowledgements

H. De Meyer is a Research Director and B. De Baets is a Post-Doctoral Fellow of the Fund for Scientific Research - Flanders.

References

[1] W. Bandler and L.J. Kohout, *Fuzzy relational products as a tool for analysis and synthesis of the behaviour of complex natural and artificial systems*, Fuzzy Sets: Theory and Application to Policy Analysis and Information Systems (S. Wang and P. Chang, eds.), Plenum Press, New York and London, 1980, pp. 341–367.

[2] W. Bandler and L.J. Kohout, *Special properties, closures and interiors of crisp and fuzzy relations*, Fuzzy Sets and Systems **26** (1988) 317–331.

[3] B. De Baets, *Ordering alternatives in MCDM problems*, Fuzzy Logic and Intelligent Technologies in Nuclear Science (D. Ruan et al., eds.), Proc. First International FLINS Workshop (Mol, Belgium), World Scientific Publishing, Singapore, 1994, pp. 48–53.

[4] J.C. Fodor and M. Roubens, *Fuzzy Preference Modelling and Multicriteria Decision Support*, Kluwer Academic Publishers, Dordrecht, 1994.

[5] R. Groenemans, E. Kerre and E. Van Ranst, *Fuzzy relational calculus in land evaluation*, GEODERMA **77** (1997) 283–298.

[6] E. Haven, *The fuzzy multicriteria analysis method: an application on NPV ranking*, Int. J. Intell. Sys. Acc. Fin. Mgmt **7** (1998) 243–252.

[7] H. Naessens, H. De Meyer and B. De Baets, *Novel algorithms for the computation of T-transitive closures and openings of proximity relations*, Proc. EUROFUSE-SIC '99 (Budapest, Hungary) (B. De Baets, J. Fodor, L. Kóczy, eds.), 1999, pp. 200-203.

[8] H. Naessens, H. De Meyer and B. De Baets, *Algorithms for the computation of the T-transitive closure of a fuzzy relation*, in preparation.

[9] B. Van de Walle, B. De Baets and E. Kerre, *Fuzzy multi-criteria analysis of cutting techniques in a nuclear reactor dismantling project*, Fuzzy Sets and Systems, Special Issue "Nuclear Engineering" (D. Ruan, ed.), **74** (1995) 115–116.

[10] B. Van de Walle, B. De Baets and E. Kerre, *Characterizable fuzzy preference structures*, Ann. Oper. Res., Special Issue "Preference Modelling" (D. Bouyssou and Ph. Vincke, eds.), **80** (1998) 105-136.

[11] L. Zadeh, *Similarity relations and fuzzy orderings*, Inform. Sci. **3** (1971) 177–200.