ON TWO INTERPRETATIONS OF COMPETITIVE CONDITIONAL FUZZY PREFERENCES

Patrick Bosc and Olivier Pivert
IRISA/ENSSAT, University of Rennes 1, Lannion, France

Keywords: Database flexible querying, Conditional preferences, Fuzzy conditions, Hierarchical queries.

Abstract: This paper introduces a new type of database queries involving fuzzy preferences. The idea is to consider competitive conditional preference clauses structured as a tree, where each child's set of a node corresponds to a disjunction of non mutually exclusive fuzzy predicates (thus the notion of competition). The paper defines two possible interpretations of such queries and outlines two evaluation techniques which follow from them.

1 INTRODUCTION

The last two decades have witnessed an increasing interest in expressing preferences inside database queries. Motivations for such a concern are manifold (Hadjali et al., 2008). First, it has appeared to be desirable to offer more expressive query languages that can be more faithful to what a user intends to say. Second, the introduction of preferences in queries provides a basis for rank-ordering the retrieved items, which is especially valuable in case of large sets of items satisfying a query. This research trend has motivated several distinct lines of research, in particular in fuzzy set applications to information systems.

In this paper, we particularly focus on hierarchical preference queries, whose basic form is: find the items satisfying \( C \) among which prefer those satisfying \( P_1 \), among which prefer those satisfying \( P_2 \) ... among which prefer those satisfying \( P_n \). The present paper extends this pattern and provides a new approach to the handling of database hierarchical preference queries. The idea is to deal with queries where preferences are modelled as a tree (and not anymore as a list) of fuzzy conditions, which implies a notion of competition between the "children conditions" of a node. Consider for instance the query: "preferably young (preference degree 1) or well paid (preference degree 0.8); if young then preferably tall (preference degree 1) or educated (preference degree 0.6); if well paid then preferably lives in Paris (preference degree 1)". Since a person can be both (somewhat) young and (somewhat) well paid, these criteria "compete" at level 1, and the same phenomenon may occur at the other levels of the hierarchy. It is thus necessary to devise an interpretation that takes into account the fuzziness of the predicates, the weights attached to them, the hierarchical aspect linked to conditional statements, and the competition between the preference conditions. The aim of the paper is to propose such an interpretation.

The remainder is organized as follows. Section 2 provides some background on a fuzzy approach previously proposed for handling hierarchical preferences. Section 3 presents two approaches to the handling of competitive conditional fuzzy preferences (CCFPs). Section 4 deals with implementation aspects for both methods introduced in the previous section. Section 5 concludes the paper and outlines some perspectives for future work.

2 RELATED WORK

In (Dubois and Prade, 1996), the authors propose to model fuzzy hierarchical preference conditions as follows. The type of clause considered is: "if \( P_1 \) (assigned the preference level 1) then \( P_2 \) (with priority level \( \alpha_2 \)), and if \( P_1 \) and \( P_2 \), then \( P_3 \) (with priority level \( \alpha_3 \)) [...]". It is assumed that \( \alpha_3 < \alpha_2 < 1 \). To interpret such a statement, the authors extend the definition of the weighted minimum operation that they originally introduced in (Dubois and Prade, 1986).

The following formula shows how the fuzzy set \( P^* \) of answers is computed in the case of a hierarchy involving three predicates (the generalization to any
The number of predicates is straightforward:

\[ \mu_p(t) = \min(\mu_p(t), \max(\mu_p(t), 1 - \min(\mu_p(t), \alpha_2)), \max(\mu_p(t), 1 - \min(\mu_p(t), \mu_p(t), \alpha_2))). \]

In this expression, the priority level of \( P_2 \) for tuple \( t \) is \( \min(\mu_2(t), \alpha_2) \), i.e., is \( \alpha_2 \) if \( P_1 \) is at least satisfied at a level \( \alpha_2 \) and \( \mu_2(t) \) otherwise. Similarly, the priority level of \( P_2 \) for \( t \) is \( \min(\mu_2(t), \mu_2(t), \alpha_3) \). Notice that it is zero if \( P_1 \) is not at all satisfied even if \( P_2 \) is satisfied. Notice also that as soon as \( P_1 \) or \( P_2 \) is not satisfied, the satisfaction of \( P_3 \) or its violation makes no difference (\( P_3 \) is not “seen”). The “guaranteed level of satisfaction” associated with \( P_i(t) \) depends on the satisfaction degree obtained by \( t \) for the predicates of higher priority. The problem, with this approach, is that \( P_2 \) (and even \( P_3 \)) may modify the order obtained through \( P_1 \).

**Example 1.** Let \( \alpha_2 = 0.7 \), \( \alpha_3 = 0.4 \) and the two tuples \( t \) and \( t' \) such that:

\[ \mu_{P_1}(t) = 0.9, \mu_{P_2}(t) = 0.4, \mu_{P_3}(t) = 0.6 \]
\[ \mu_{P_1}(t') = 0.8, \mu_{P_2}(t) = 1, \mu_{P_3}(t) = 1 \]
We get: \( \mu_{P_2}(t) = 0.4 \) and \( \mu_{P_2}(t') = 0.8 \) whereas \( t \) is better than \( t' \) with respect to predicate \( P_1 \).

The main limitations of this approach, with respect to the goal we pursue in this study, concern:

- the semantics of the hierarchical mechanism, as illustrated by the previous example;
- the structure of the hierarchical preference clauses considered (a “cascade” with no disjunction, which implies the absence of any competitive behavior).

### 3 TWO VIEWS OF CCFPs

The idea consists in using disjunctive preference clauses (for instance: preferably age < 30 or salary > 2000; if age < 30 then preferably tall; if salary > 2000 then preferably lives in Paris). Using disjunctive conditions implies, when these conditions are not exclusive, that a tuple can satisfy several conditions at the same time (for instance, one may be younger than 30 and have a salary over 2000 euros). Therefore, a tuple can “progress” in several branches of the preference tree. This implies to define a mechanism for determining the “best” evaluation of a tuple relatively to the preference tree considered. The query format corresponding to this specification is:

\[(w_1) P_1 \text{ or } (w_2) P_2 \text{ or } ... \text{ or } (w_n) \text{ anything}_0;\]

\[\forall j \in [2..n], V'_t[j] = \min_{u=1..j} V_t[u]\]
The score $\mu(t)$ is computed as follows:

$$
\mu(t) = \frac{\sum_{u=1}^{n_i} V_i(t)[u]}{n_i}
$$

(1)

The operator underlying this computation is the hierarchical aggregation operator proposed in (Bosc and Pivert, 1993). It captures the notion of hierarchy by means of the transformation of $V$ into $V'$ which expresses that a predicate cannot contribute more to the final degree than its ancestors. Finally, the best score associated with $t$ is:

$$
\mu(t) = \max_{i=1..b} \mu_i(t).
$$

(2)

One may then order the tuples the following way:

$$
t \geq t' \iff \mu(t) \geq \mu(t').
$$

Obviously, $\geq$ is a complete preorder.

**Example 2.** Let us consider the following query:

(1) young, or (0.7) (around 50 y.o. and well-paid), or (0.2) anything;

if young, then preferably (1) (tall and blond), or (0.5) well-paid;

if (around 50 y.o. and well-paid), then preferably (1) liberal profession;

if anything then preferably (1) close to Lyon or (0.7) close to Paris;

if (young and well-paid) then preferably (1) liberal profession.

Let us consider the relation represented in Table 1.

**Table 1: Degrees attached to the tuples of a relation Person.**

<table>
<thead>
<tr>
<th></th>
<th>$\mu_{ha}$</th>
<th>$\mu_{hp}$</th>
<th>$\mu_{cl}$</th>
<th>$\mu_{hp}$</th>
<th>$\mu_{ha}$</th>
<th>$\mu_{hp}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>0.7</td>
<td>0.6</td>
<td>blond</td>
<td>0.4</td>
<td>no</td>
</tr>
<tr>
<td>$t_2$</td>
<td>0.2</td>
<td>0</td>
<td>0.8</td>
<td>dark</td>
<td>0.5</td>
<td>no</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0</td>
<td>1</td>
<td>0.4</td>
<td>blond</td>
<td>0.8</td>
<td>no</td>
</tr>
<tr>
<td>$t_4$</td>
<td>0.5</td>
<td>0</td>
<td>0.9</td>
<td>blond</td>
<td>0.6</td>
<td>yes</td>
</tr>
<tr>
<td>$t_5$</td>
<td>0.7</td>
<td>0</td>
<td>0.9</td>
<td>dark</td>
<td>0.3</td>
<td>yes</td>
</tr>
<tr>
<td>$t_6$</td>
<td>0</td>
<td>0</td>
<td>0.8</td>
<td>red</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>$t_7$</td>
<td>0.6</td>
<td>0</td>
<td>0.2</td>
<td>blond</td>
<td>0.4</td>
<td>yes</td>
</tr>
</tbody>
</table>

It is assumed that very well-paid (vwp) is interpreted as: $vwp(x) = \text{well-paid}(x)^2$. The conjunction involved in conditions such as “around 50 y.o. and very well-paid” is interpreted with the t-norm minimum. The tuples from Table 1 get the following scores:

$$
\mu(t_1) = 0.2, \mu(t_2) = 0.4, \mu(t_3) \approx 0.22,
$$

$$
\mu(t_4) = 0.5, \mu(t_5) = 0.35, \mu(t_6) = 0.56, \mu(t_7) = 0.4.
$$

Finally, the ordered answer obtained is:

$$
t_6 \geq t_4 \geq \{t_2, t_7\} \geq t_5 \geq t_3 \geq t_1.
$$

**Comment about the Semantics.** With this approach, the transformation of $V_i(t)$ into $V'_i(t)$, reduces the discrimination power. For instance, $[0.6, 0.6, 0.6]$ is considered equivalent to $[0.6, 1, 1]$. In the following subsection, we propose an alternative solution based on the lexicographic order.

### 3.2 Lexicographic-order-based Method

Another approach for dealing with branches of different lengths is to complete each vector so that it contains exactly as many scores as there are nodes in the longest branch of the query tree. In order for the final ranking to be consistent, this completion is based on the following principle. Let $n_{\text{max}}$ be the length of the longest branch of the tree. Let $n$ be the size of vector $V_i(t)$. Vector $V'_i(t)$ is completed (i.e., transformed into a vector $V'_i(t)$ of length $n_{\text{max}}$) the following way:

the last degree of the original vector is propagated to the right so as to get the desired length. This choice seems the most reasonable, compared to a completion with 0’s (resp. with 1’s) which could appear excessively pessimistic (resp. optimistic). The best evaluation of $t$ corresponds to the best vector $V'_i(t)$ in the sense of the lexicographic order:

$$
\text{best}V'(t) = V'_i(t) \text{ such that } \forall i \in [1..b], \text{best}V'_i(t) \geq_{\text{lex}} V'_i(t)
$$

where $b$ denotes the number of branches in the query tree and $\geq_{\text{lex}}$ denotes the lexicographic order.

Then, two tuples $t$ and $t'$ can be compared using the lexicographic order on their best evaluations:

$$
t \geq t' \iff \text{best}V'(t) \geq_{\text{lex}} \text{best}V'(t')
$$

Since $\geq_{\text{lex}}$ is a complete pre-order, so is $\geq$.

**Example 3.** Let us come back to the query and data from Example 2. With this new approach, we get the vectors:

$$
\text{best}V'(t_1) = [0.2, 1, 1],
$$

$$
\text{best}V'(t_2) = [0.8, 0.25, 0],
$$

$$
\text{best}V'(t_3) = [0.45, 0, 0], \text{best}V'(t_4) = [0.5, 0.6, 0.6],
$$

$$
\text{best}V'(t_5) = [0.7, 0.15, 1], \text{best}V'(t_6) = [0.56, 1, 1],
$$

$$
\text{best}V'(t_7) = [0.6, 0.2, 0.2].
$$

and the ordered obtained i

$$
t_6 \geq t_4 \geq \{t_2, t_7\} \geq t_5 \geq t_3 \geq t_1.
$$

Notice that this order differs from that obtained with the previous approach (cf. results from Example 2).
Comments about the semantics. Two points are worthy of comments. First, it must be noticed that the lexicographic order has a somewhat “brutal” way of discriminating, which can be felt as counter-intuitive in some cases. For instance, with the approach proposed, the vector \([\alpha, 0, 0, \ldots, 0]\) is preferred to \([\alpha - \epsilon, 1, \ldots, 1]\) even for a very small \(\epsilon\). A second point concerns the fact that zeros do not “propagate” to their descendants in the tree. Thus, vector \([\alpha, 0, \beta]\) is preferred to \([\alpha, 0, \gamma]\) as soon as \(\gamma\) is smaller than \(\beta\), even though the parent node of \(\beta\) and that of \(\gamma\) yield the score zero. Again, this can be considered debatable since one might think that a hierarchical behavior would impose to take a node into account only if its parent is somewhat satisfied.

4 IMPLEMENTATION ASPECTS

For the score-based approach, the query processing method is based on a depth-first scan of the query tree. The algorithm includes two pruning criteria:

1. when the local score obtained by tuple \(t\) for the predicate associated with the current node equals 0, the evaluation of a branch stops. This is due to the fact that the score obtained for a node acts as a threshold (upper bound) on the scores obtained through the descendants of this node (cf. the transformation of vector \(V\) into \(V'\));

2. for the same reason, if a child \(x\) of the root returns a score \(e\) which is less than the current best score attached to \(t\), there is no need to check the descendants of \(x\). Indeed, the final score obtained along every branch originating in \(x\) cannot be greater than \(e\).

As to the approach based on the lexicographic order, query processing is based on a breadth-first scan of the query tree which computes the best vector (according to the lexicographic order) associated with tuple \(t\).

The important point to notice is that the data complexity of conditional competitive queries is linear, which implies that they are perfectly tractable. However, with respect to regular selection queries (involving conditions of the form \(\text{attribute} \ \theta \ \text{constant}\) or \(\text{attribute}_1 \ \theta \ \text{attribute}_2\) where \(\theta\) denotes a comparator), the processing of an individual tuple is more expensive in the conditional competitive case since one has to deal with a tree of predicates instead of a “flat” condition which is in general more simple.

5 CONCLUSIONS

In this paper, a new type of database queries involving preferences is introduced. The idea is to allow for competitive conditional preference clauses structured as a tree, of the type “preferably \(P_1\) or \(\ldots\) or \(P_n\); if \(P_1\) then preferably \(P_{1,1}\) or \(\ldots\); if \(P_2\) then preferably \(P_{2,1}\) or \(\ldots\)”, where the \(P_i\)’s are not exclusive (thus the notion of competition). Two interpretations of such queries have been defined, one based on the hierarchical aggregation operator previously proposed in (Bosc and Pivert, 1993), the other on the lexicographic order. These approaches lead to different complete pre-orders and the choice of the most suitable interpretation depends on the semantics that the user wants to give to the notion of hierarchy in his/her query.

Future works should thus notably aim at proposing some mechanisms that can help the user in this choice. In any case, the processing of such queries is perfectly tractable since their data complexity is linear. Among other perspectives for future work, the main one concerns the effective integration of such a query functionality into a regular database system.

REFERENCES


