Inductive Models of User Preferences for Semantic Web

Alan Eckhardt

Charles University, Faculty of Mathematics and Physics
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
alan.eckhardt@mff.cuni.cz

Abstract. User preferences became recently a hot topic. The massive use of internet shops and social webs require the presence of a user modelling, which helps users to orient them selfs on a page. There are many different approaches to model user preferences. In this paper, we will overview the current state-of-the-art in the area of acquisition of user preferences and their induction. Main focus will be on the models of user preferences and on the induction of these models, but also the process of extracting preferences from the user behaviour will be studied. We will also present our contribution to the probabilistic user models.

1 Introduction

The user preference modelling plays an important role in the current web. Internet shops need to help the user to find the product he/she searches for, social webs may suggest a contact that the user wants. The process of acquisition of user's preferences starts with the acquisition of known preferences (e.g. from the user behaviour) and then using these known preferences to get the user's preferences of other objects. In this paper, an example of a user who is buying a digital camera will be used. In Section 2, several user models will be presented and in Section 3, some of current methods of induction of user preferences will be described, including our own probabilistic model.

1.1 A use case for the induction of user preferences

We will present a typical use case for the induction of user preferences. We will describe a complex system for the extraction of information from the web and for the presentation of collected information to the user. This system will be aimed to help the user to find a camera that fits best his needs. Whole system proposition is in Figure 1.

The first task of this system is to collect data from various sources from the internet. Information is stored in various forms, most often in HTML format, and it has to be transformed to a computer-readable form (Semantic data). The typical computer-readable form is RDF [2] - a language of triples of the form (subject, predicate, object). Extension of RDF is OWL [1] which is one of standard ontology languages. Ontologies can be also used to annotate raw text data from the web. This transformation is called the ‘semantic annotation’ and there are many methods performing semantic annotation, though their accuracy and universality may be questioned. However, studying the semantic annotation is not the aim of this work, it is one of components in whole process.

With semantically annotated data or with online RDF sources, we may present the integrated information to the user. Several works (e.g. [8]) concern the graphical user interface that represents the semantic data. The task is to present the most important objects to the user in a such way that he/she will notice them before noticing the other objects.

The inductive methods enable to determine which objects are important and which are not. The process of determining the importance of an object is iteratively executed, it may be triggered e.g. by some user behaviour, for example by clicking on some object that is not considered important. Interpreting user behaviour is another part of whole process.

The interpreted behaviour is then used to generate a user preference model. This can be done by an already known inductive method. This user preference model will be then used to alter the appearance of the web page, for presenting preferred objects etc. This information can be also used by other servers, in case of a distributed computation.

In our work, we will focus on the induction process - the induction of preference model from some rated objects.

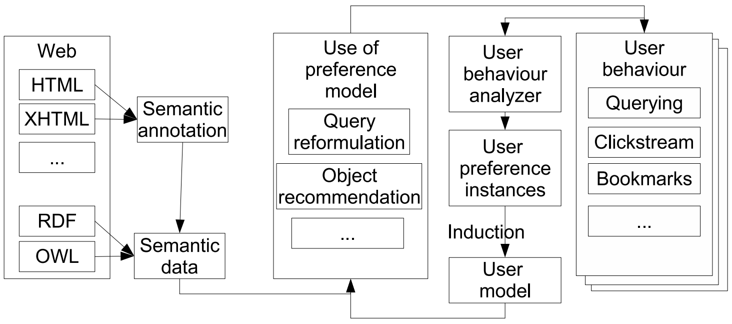

Fig. 1. Complex system for user preferences on the web

2 Models of user preferences

We will use the following notation - $u_1, ..., u_k \in U$ for the users, $o_1, ..., o_n \in O$ for the objects, $a_1^1, ..., a_1^m$ for the attribute values of the object o_1 and $A^1, ..., A^m \in A$ for the attributes of objects. When speaking only of one object, we will also use only o for object and $a^1, ..., a^m$ for its attribute values. Often, all objects are from the same domain and they will have the same set of attributes. For that reason we will be interested mainly in the attribute values. We will denote the user's preference of an object as $P(o)$, meaning how much object o is preferred. We will use the notation $P_1(o)$, which means the preference of the user u_1 on the object o, when we want to explicitly denote the user. The range of P depends on user model used, some of them do not have direct preference, e.g. the preference relations studied in Section 2.2.

We now briefly distinguish several types of attribute domains, as was done in [5]. The first domain type is nominal. These domains have no ordering and are mainly text based. Typical example may be the color or the manufacturer. Second type are ordinal domains, on which exists some ordering, but not unit. For example set Big, Medium, Small may form an ordinal domain. When a unit of measure is added, we get interval domains - for example {1, 0.2, 0, -0.4, -1}. Finally, ratio scales have also an element 0 explicitly defined. Example of ratio scale may be the price, the number of megapixels, the weight etc. We will refer to both ratio and interval domains as numerical domains. In real data, most frequent are nominal and interval/ratio domains. Ordinal domains are created with a user influence - a user will say that every object that weights more than 600g is heavy, above 200g is medium and less than 200g is light.

2.1 Boolean preference

Boolean user preference model is used in some methods, where the user preference model is not explicitly mentioned. This model distinguish only two states of an object - either preferred or non-preferred. This is very simple approach with little semantic, but may be used when lot of computing power is required. In these cases, preference is represented by a vector of n bits, we will note this vector $v(u) : U \to [0,1]^n$. Operations on these vectors are fast when only binary operations like and, or, xor etc. are needed. These operations may be sufficient for some inductive methods but not for others. For example, this model can be successfully used in the collaborative filtering, which is described in Section 3.2.
Comparison of two user preferences.

Computing the similarity of two users may be computed in following way

$$s(u_1, u_2) = 1 - \sum_{i=1}^{n} (v(u_1) \text{ xor } v(u_2))[i]/n.$$

The sum in equation expresses the number of ratings, where both user disagree, e.g. object is preferred for u_1 and non-preferred for u_2. If we based the similarity on common preferences of u_1 and u_2, it will be influenced by the number of rated objects, which is not desirable. If this fact is of no relevance, alternative way for computing similarity may be

$$s(u_1, u_2) = \sum_{i=1}^{n} (v(u_1) \text{ and } v(u_2))[i]/n.$$

Computing the similarity of two users is essential for methods like collaborative filtering. Surprisingly, there are not many articles concerning this problematic. Further investigation and research in this area may reveal interesting ideas.

2.2 Preference relations

Preference relations are the oldest model of user preferences models, its description may be found e.g. in [15]. Basic idea behind preference relations is to characterize the relation between objects o_1 and o_2. We can say that o_1 is more preferred than o_2, o_1 is equal to o_2, o_1 is incomparable to o_2 or that o_1 is a little better than o_2 but not much. For the strict preference, we traditionally denote this relation as P. Then $P(o_1, o_2)$ states that o_1 is more preferred than o_2. For equivalence of two objects, we use I, e.g. $I(o_1, o_2)$ means that o_1 is as preferred as o_2. Finally, relation R is created as union of P and I, $R(o_1, o_2)$ meaning o_1 is equal or preferred to o_2. For incomparability, relation J is introduced. Then $J(o_1, o_2)$ means that o_1 and o_2 are incomparable.

We left out the case when o_1 is a little better than o_2. We may create new relation Q, so that $Q(o_1, o_2)$ states that o_1 is a little better than o_2. By a simple extension, set of relation $Q_1, ..., Q_n$ will represent the fact that o_1 is a little better than o_2, with Q_1 representing the lowest difference of preference of o_1 and o_2 and Q_n the highest difference.

Properties of relations determine properties of preferences. There are several properties, such as the existence of a minimum or the completeness (linearity) of the relation. For deeper insight in these properties, we again recommend [15], which is specialized survey of preference relations.

All these structures may be extended to valued structures. One special case is many valued logic, studied more in depth in 2.3. An example of a valued structure is $\mu(P(a, b)) : O^2 \rightarrow [0, 1]$. The interval $[0, 1]$ may be replaced by any other linear numerical structure, and it represents the degree (truth value) of the relation. Valued relations may successfully replace relations $Q_1, ..., Q_n$, which are a middle step between standard relations and valued relations.

Comparison of two user preferences.

When we want to compare preferences of two users, we have to compare two preference relations. When relation is ordered linearly, we compare two ordered lists of objects. In that case, we may count the number of permutation between both lists, which is traditional measure of computing the similarity (or the distance, in this case). However, this may be not the best distance used, because it makes no difference of distance between permuted objects. Switching two neighbour objects makes less change than switching first and last object.

In [14] the distance of two interval fuzzy preference relations is described. However, it can't be simply used as a generalization of simple preference relation, because it will degrade into simple 'equal' or 'not equal' semantics.

2.3 Many valued logic

Many valued logic is an extension of the traditional two valued logic. In the two valued logic, a variable may be either true or false, in the many valued logic a whole set of possible truth values is introduced, often denoted as T. T should form a lattice, typical case is a linear structure and the most used is interval $[0, 1]$. The set T will represent the set of preference values where 1 is most preferred and 0 is least preferred. Other structures than $[0, 1]$ are possible to use, for example discrete set {Worst, Worse, Neutral, Better, Best} or {One star, Two stars, Three stars, Four stars, Five stars} may be relevant in some cases. We will interpret the truth value as a degree of preference.

When creating this extension of the two valued logic, we must define a new semantics for logical operators, predicates and quantifiers. This definition can be found for example in [13], but this is not of major interest in this work.

We will use two structures from the many valued logic - the first are fuzzy functions of an attribute domain and the second is an aggregation function. A fuzzy function represents user's ordering and normalization of a domain. For example, consider the attribute price. Most people prefer low prices over high prices. In figure 2,a) we may see an example of the fuzzy function for price.

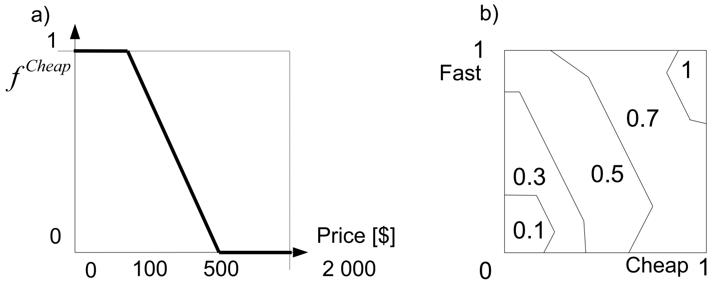

Fig. 2. Fuzzy function of price (a) and an example of a more complicated aggregation function (b)

An aggregation function is used to aggregate several truth values, or preferences, into one truth value, therefore it is a function $@ : T^m \rightarrow T$. There are few restrictions on an aggregation function - it must be monotone in all variables, and $@(0, .., 0) = 0$ and $@(1, .., 1) = 1$. An aggregation function is very suitable for the modeling of the user preferences of the complex objects. The user aggregates attributes of an object into the preference of the object itself. An example of an aggregation function may be

$$@_{U_1}(\text{MPix\_}U_1(x), \text{Fast\_}U_1(x), \text{Cheap\_}U_1(x)) = \frac{5 * \text{MPix\_}U_1(x) + 1 * \text{Fast\_}U_1(x) + 3 * \text{Cheap\_}U_1(x)}{9}$$

Symbols MPix, Fast and Cheap denotes fuzzy sets of particular attributes. E.g. Cheap_$U_1(D50)$ represents how camera D50 is cheap for U_1.

We consider the weighted average as a good example of a user fuzzy function. It has clear semantics, because we can see directly, which of the attributes are important for the user and which are not. Even more, from the weight we can deduce how much important an attribute is. However, many more aggregation functions that fits better to psychological aspects of human decision process may be represented. An example of a more complicated aggregation function is in Figure 2,b). These two mechanisms allow us to create very flexible model of user preferences and moreover, the aggregation function models also user decision process.

3 Inductive methods

In this section, we will examine several inductive methods that are used to create a user preference model from some input. The user preference model is often independent of the inductive method, the input may be represented in several ways but often the paradigm of object with some attributes is expected.

Most of the methods expect a training set of objects, which are supposed to belong to 'classes'. These classes of objects in the training set may have different forms, depending on the model of user preferences we are using. For example, when using many valued logic, one class may be $o : P(o) \in [0, 0.1]$. Some of the models of user preferences we have described above do not have a direct interpretation as classes. With preference relations, we have only comparison between two objects. We assume that a method will transform user preferences into several (possibly discrete) classes. For preference relations we may order objects and associate a weight corresponding to the position in the ordered set.

3.1 Inductive logic programming

Inductive logic programming is a method to obtain a logic program. This program may be very general, in our case, it will represent rules that user uses in decision process. After application of these rules to an object, the preference of that object should be obtained. We will describe only predicate logic programs, which are more expressive than sentential logic programs. An introduction to induction of logic programs may be found in [11].

Rules have a head and a body. The head of a rule is a single predicate and the body is a conjunction of predicates. When using fuzzy predicate logic, each predicate has also value that represents the truth value of the predicate. For simplicity (and because of space limitation), we will describe two-valued logic program.

For example, following rules may represent user preferences of cameras

```
GoodMPix(camera) <- MPix(camera)>5;
GoodWeight(camera) <- Weight(camera)<700 & Weight(camera)>300;
GoodCamera(camera) <- GoodMPix(camera) & GoodWeight(camera);
```

Inductive process works with this input

1. The background theory B.
2. Positive examples E^+.
3. Negative examples E^-.

Background theory is used to infer new statements H (hypothesis) about examples. Both E^+ and E^- are formulas, but E^- have empty head, e.g.

```
<- GoodCamera(D50);
<- GoodCamera(D40);
```

On the other side, positive examples have empty body, e.g.

```
GoodCamera(D200) <- ;
GoodCamera(D2x) <- ;
```

We present also an example of a background knowledge B:

```
Weight(D2x)=12 <- ;
Megapixels(D2x)=1150 <- ;
```

Four conditions must be fulfilled

1. Prior satisfiability B & $E^- \not\models \Box$
2. Posterior satisfiability B & E^- & $H \not\models \Box$.
3. Prior necessity $B \not\models E^+$.
4. Posterior sufficiency B & $H \models E^+$.

The symbol $\Box$ represent the contradiction or false. The meaning of these condition is clear - with B and E^- we should not get a contradiction, e.g. E^- should comply to the background knowledge. With the B, E^- and H we should not get a contradiction either. On the other hand, we want that examples are not deducible from B itself, only with addition of H.

Now we will describe a general algorithm of hypothesis construction, as proposed in [11].

```
QH = Initialize();
do
   Delete H from QH;
   Choose rules $r_1, ..., r_k \in R$ to be applied to H;
   Apply $r_1, ..., r_k$ to H, obtaining $H_1, ..., H_n$;
   Add $H_1, ..., H_n$ to QH;
   Prune QH;
while not Stop-criterion(QH)
```

QH is a set of candidates to hypothesis and R is a set of rules, which transform H. An example of a rule may be dropping a clause or adding a clause to the body of H. During each step, a hypothesis H and rules that will be applied to H are chosen. Result of the application of rules on H are then stored in

QH and candidate set is pruned. Pruning means that useless candidates are deleted. Implementation of each of methods Delete, Choose, Prune and Stop-criterion may be different. Also the set of available rules R may differ across implementations.

An example of application of a rule on a hypothesis `GoodCamera(D2x) <- ;` may be
`GoodCamera(camera) <- Megapixels(camera)=12;`
or
`GoodCamera(camera) <- Weight(camera)=1150;`
This is an example of a generalization rule, whose result must be verified on E^+ and E^-.

The hypothesis should be completely correct, i.e. it should have the Posterior sufficiency property. However, if we relax this property, a kind of probabilistic rules will be generated. The probabilistic approach is further studied in 3.4.

3.2 Collaborative filtering

Collaborative filtering represent widely used method for acquisition of user preferences. It is based on assumption, that the preference of user u_0 on object o will be the same as the preference of users $u_1, ..., u_k$ that are 'similar' to u_0. The similarity of users is based on similarity of ratings on objects. Many collaborative filtering methods are described in [10].

This method requires a lot of ratings of objects by a lot of users. For computing the similarity of users, we need a lot of object ratings, for accuracy of computing the rating of object o, we need a lot of users similar to u_0.

There are several different algorithms for collaborative filtering. The first and most simple is K-NN, the K nearest neighbor. This is most intuitive algorithm - for a user u_0, we find the K nearest users $u_1, ..., u_K$. The distance is computed by the similarity of users' ratings, for example

$$s(u_i, u_j) = \sqrt{\sum_{l=1}^{n} (P_i(o_l) - P_j(o_l))^2} \tag{1}$$

Having these K nearest neighbours, we may compute the rating of objects as average of users' ratings

$$P_0(o_i) = \sum_{j=1}^{K} (P_j(o_i))/K \tag{2}$$

Another method of computing new ratings is to use a naive Bayes classifier [6]. For each object o, we construct a separate Bayes classifier. Input of the network are the ratings of all objects other than o. Bayes network will answer the question "What is the value $P(o)$, when the user rated other objects this way?". Bayes network learns its parameters from the ratings of users that have rated o.

Other methods are considered as a content filtering methods - they work with the objects and their properties rather then with the preferences of other users. However, both approaches are often combined. Collaborative filtering can not be appropriately used for new objects, which have not yet been rated by any user. For this reason, some kind of the content filtering is also used and the results of both methods are combined together.

However, some of the presented methods may be used both on other users' ratings and the properties of objects, or both together.

3.3 Decision trees and rules

Decision trees are well known structure from the data mining theory, they are used to model user decision process (or any decision process). Decision trees are oriented trees with class names in leaves and a rule associated to each inner node. An example of a decision tree which models user preferences of digital cameras is in figure 3. We can see that cameras with at least 10 megapixels are good, cameras that costs more than 500$ are bad.

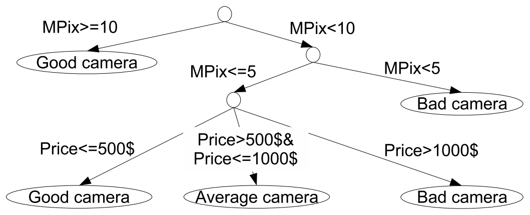

Fig. 3. An example of a decision tree

Theory of the induction of the decision trees may be found in [12]. Induction of a decision tree is discussed mainly for discrete attributes with few values. Decision trees were extended to fuzzy decision trees, one of a new contribution is in [5], older one is [3].

Basic induction procedure starts with a 'training set', which is a set of objects with known classes. From these objects, a tree is constructed and then it is verified on a test set of objects, which is also a set of objects with known attributes. The construction is typically a top-down algorithm. During each step, all possible splits are considered and the most appropriate one is chosen. The appropriateness is measured with an 'impurity measure'. Impurity measure measures how evenly the data are spread in classes. When all objects are only in one class, impurity measure is 0, when all classes contain same number of objects, impurity measure should be 1. Most used is entropy-based impurity measure ([12]) and Gini index ([4]). The structure of a tree depends mainly on the impurity measure used during its construction.

3.4 Probabilistic methods

Probabilistic methods use probability as a method of inducing user preference. There are several possible approaches to statistical interpretation of preference data, e.g. [9]. Usage of probability is reasonable when working with user preferences, because few users have consistent preferences. Often, an exception from a general rule occurs. This exception have to be handled explicitly in non-probabilistic methods, but it creates no problem in the probabilistic methods. For example in inductive logic programming, we can assign a probability to each rule, denoting how much the rule is true in general, or in decision trees the probability would be associated to each left node.

Probabilistic model for boolean preference model. The probabilistic preference model proposed in [9] is based on two measures - the first is the actual user preference and the second is the accessibility (or the frequency) of the object. The second one is important because when trying to induce user preferences from user behaviour it is apparent that user will rather examine frequent items than rare items just because they are more frequent.

The preference of an object is $Pref(x) = f(x|V)$, where V is a user profile or a user history and f returns the preference value of object x. Objects have again several attributes, in [9] called 'features'. The preference of an object is computed as

$$Pref(x) = 1/|X| \sum_{a \in X} Pref(a), \tag{3}$$

e.g. it uses the average of attributes values preferences. The problem of finding $Pref(a)$ is then analyzed, actually in the similar way as in our previous work [7]. We will compare these approaches in Section 3.4. The suggestion in [9] is to use formula

$$Pref(a) = I(X(a); V) = log \frac{P(X(a)|V)}{P(X(a))} \tag{4}$$

where $X(a)$ is the set of objects containing a. In other words, attribute value a is preferred, if the probability that the user selects an object with a is higher than the probability of the occurrence of an object with a in the whole set of objects.

Probabilistic model for many valued logic. A probabilistic model for the many valued logic is our contribution. It is aimed on nominal attributes and uses only weighted average as aggregation function. We concentrate on the case, when we know preferences of some objects and user aggregation function but not the preference of attribute values.

We are missing the preference of an attribute value a which is the value of attribute A_k. But we know the preferences of objects and the aggregation function. We will consider the set $X(a)$ of objects which have the attribute value a. We will look into the distribution of the preference of these objects. When most of the objects in $X(a)$ have high preference, attribute value a will also have high preference. Formally,

$$P(a) = \frac{\sum_{o \in X(a)} P(o)}{|\{o \in X(a)\}|}. \tag{5}$$

The ratio between the weight of the attribute A_k in aggregation function @ and the sum of the weights of all attributes represents the probability that the preference is computed correctly. It is denoted formally in the following equation

$$P(A_k) = \frac{W(@, A_k)}{\sum_{i=1,..,m} W(@, A_i)}, \tag{6}$$

where $W(@, A_j)$ is the weight of attribute A_j in @. The preference of an object is influenced more by an attribute with a big weight than by an attribute with a small weight. Therefore this method is useful mainly for the attributes with big weight.

Computing preference of one attribute value a may be costly when the number of objects with a is big. However, higher number of objects with a means also higher precision of this method. Naturally, this method is only useful for the domains with discrete values, especially non ordered domains like color or manufacturer. This method can't be successfully used for continuous domains, because there will be very few objects with the same attribute value. However, we may divide these continuous domains to a set of discrete intervals, and use the method proposed above on these intervals.

Comparison with our model. Our model is an extension over the model proposed in [9]. There are two aspects in which our approaches differ

1. In [9] the boolean user model is used (implicitly). We use many-valued logic model, which is more general.
2. The preference of an object is computed in [9] as a simple average of preference of its attributes. In our model, weighted average is used.

However, there is a similarity in our approaches - we use the preference of objects for acquiring the preference of attribute values. This is an inverse process to deduction, where the preference of an object is computed from the preference of its attribute values.

4 Conclusion

In this paper, we have reviewed some of the main user preference models. There are other models as well, their complete listing is beyond the scope of this paper, we recommend [15] to the interested reader. The user model is used in a web system to better present data to the user or to alter his/her query in order to the results of the query actually better fit his/her preferences.

The creation of the user model is often done by inductive methods, which were studied in this paper in Section 3. We presented methods that are used for induction of user preferences and one probabilistic model for boolean user preferences. We have developed a similar approach for many valued logic, which is more general and flexible than the method studied in section 3.4. The precision and the computational cost of our approach is still to be tested on real data. These experiments are however beyond the scope of this paper, which is an overview of methods used for the induction of the user preferences.

Acknowledgment

Supported by Czech projects MSM 0021620838, 1ET 100300517 and 1ET 100300419.

References

1. Owl, ontology web language . `http://www.w3.org/TR/owl-features/`.
2. Rdf data format. `http://www.w3.org/TR/rdf-primer/`.
3. B. Apolloni, G. Zamponi, and A. M. Zanaboni. Learning fuzzy decision trees. *Neural Networks*, 11(5):885–895, 1998.
4. L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Chapman & Hall, New York, 1993.
5. K. Cao-Van. *Supervised Ranking, from semantics to algorithms.* Ph.D. dissertation, Ghent University, 2003.
6. P. Domingos and M. J. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997.
7. A. Eckhardt. Methods for finding best answer with different user preferences, In Czech only, Master's thesis, Charles University in Prague. 2006.
8. Lars Hult, Magnus Irestig, Jonas Lundberg *Design Perspectives. Human-Computer Interaction.* Vol. 21, No. 1, Pages 5-48, 2006.
9. S. Y. Jung, J.-H. Hong, and T.-S. Kim. A statistical model for user preference. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):834– 843, June 2005.
10. B. Marlin. Collaborative filtering: A machine learning perspective. Master's thesis, University of Toronto, 2004.
11. S. Muggleton and L. D. Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.
12. J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, 1986.
13. P. Vojtáš. Fuzzy logic programming. Fuzzy Sets and Systems. 124,3 (2001) 361-370
14. Z. Xu. On compatibility of interval fuzzy preference relations. *Fuzzy Optimization and Decision Making*, 3(3):217–225, 2004.
15. M. Öztürké, A. Tsoukias, and P. Vincke. *Preference modelling. Multiple Criteria Decision Analysis: State of the Art Surveys.* Springer New York, 2006.