

USING A SOFT COMPUTING TECHNIQUE FOR DATA MINING APPLICATIONS

Ramón Alberto Carrasco, María Amparo Vila

Dpto de Ciencias de la Computación e I.A., Universidad de Granada, Granada, Spain

Email: rcarrasco@caja-granada.es, vila@decsa.ugr.es

José Galindo

Dpto. de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Spain

Email: ppgg@lcc.uma.es

Keywords: Flexible Queries, Fuzzy SQL, Data Mining, Fuzzy Databases, Artificial Neural Networks

Abstract: At present we have a FSQL server available for Oracle© Databases, programmed in PL/SQL. This server allows us to query a Fuzzy or Classical Database with the FSQL language (Fuzzy SQL). The FSQL language is an extension of the SQL language, which permits us to write flexible (or fuzzy) conditions in our queries to a fuzzy or traditional database. In this paper we show an extension of FDBR architecture of FSQL for fuzzy handling of different types of data. The main advantage is that any user can to define his own fuzzy comparator for any specific problem. As example a method of ranking fuzzy numbers using Neural Networks to compare fuzzy quantities in FSQL is shown in this paper. We consider that this model satisfies the requirements of Data Mining systems (handling of different types of data, high-level language, efficiency, certainty, interactivity, etc) and this new level of personal configuration makes the system very useful and flexible.

1. INTRODUCTION

In the last years, the management applications have been popularised and widely extended. In these applications, the database management is very important. Thus, the final users use many times the DBMS (Database Management Systems) directly or through an interface program (front-end).

On the other hand, the fuzzy databases have been developed in the last years, rising up different models [14], among which they highlight the Prade-Testemale model, the Umano-Fukami model, the Buckles-Petry model, the Zemankova-Kaendel model and the GEFRED model by Medina-Pons-Vila [13]. This last model represents an eclectic synthesis of the different models, which have appeared to deal with the problem of the representation and management of fuzzy information in relational databases (FRDB). Besides, for the GEFRED model, the FSQL language [8,9,10], a fuzzy (or flexible) query language based on the SQL language, has been defined. This language allows us to express sentences taking into account the characteristics of imprecise information. Thus, as we will see, the FSQL queries allow expressing fuzzy conditions, to calculate fulfilment degrees, to establish fulfilment thresholds... Some applications for FSQL Server can be found, for example, in [1,2,3,4,10,11].

First, we include a brief explanation of the main advantages of the FSQL SELECT sentence, in order to express fuzzy queries. Besides, we explain the FRDB architecture for the FSQL Server (a more detailed description of this can be found in [8,10]). Then, we extend the FRDB architecture for fuzzy handling of different types of

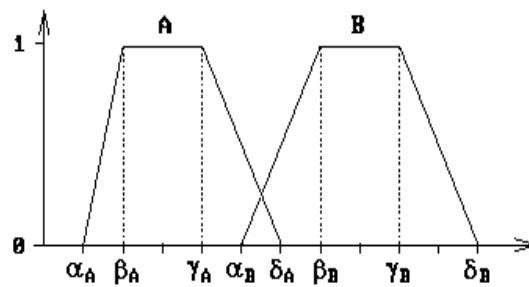


Figure 1. Trapezoidal possibility distributions: A, B.

data. After, we show some examples using the improved FRDB architecture on different types of data or with different fuzzy comparison functions. Thus, we define a method of ranking fuzzy numbers using Artificial Neural Networks (ANN). Therefore, we use the defined comparison method to compare fuzzy quantities in FSQL, allowing any user to train his own fuzzy ANN comparator. Thus, the trained ANN will be a special comparator for any specific problem. A more extend information on this can be found in [4].

We consider that this model satisfies the requirements of Data Mining (DM) systems [5,6,7] (handling of different types of data, high-level language, efficiency, certainty, interactivity, etc) and this new level of personal configuration makes the system very useful and flexible. Finally, we present an example used in our test, and we suggest some conclusions.

The example is in the context of a bank. This area needs a Data Mining system tailored to its needs, because this area manages very large databases and these data has a very concrete meaning. Thus, data must be treated according to this meaning.

2. FSQL A LANGUAGE FOR FLEXIBLE QUERIES

The FSQL language [8,9,10] extends the SQL language to allow flexible queries. We have extended the `SELECT` command to express flexible queries and, due to its complex format, we only show an abstract with the main extensions added to this command:

- **Linguistic Labels:** If an attribute is capable of undergoing fuzzy treatment then linguistic labels can be defined on it. These labels will be preceded with the symbol \$ to distinguish them easily. Every label has an associated trapezoidal possibility (Figure 1) or it is a scalar if there is a similarity relationship defined between each two labels in the same domain.
- **Fuzzy Comparators:** In addition to common comparators (`=`, `>`, etc), FSQL includes some different kinds of fuzzy comparators as shown in Table 1. The most important is used to compare two trapezoidal possibility distributions A, B with $A=[\alpha_A, \beta_A, \gamma_A, \delta_A]$ $B=[\alpha_B, \beta_B, \gamma_B, \delta_B]$ (see Figure 1). In the same way as in SQL, fuzzy comparators can compare one column with one constant or two columns of the same type. More information can be found in [9,10]. These definitions are based in typical possibility comparators in fuzzy set theory.

Fuzzy Comparator (fcomp) for:		Significance
Possibility	Necessity	
FEQ	NFEQ	Fuzzy Equal
FGT	NFGT	Fuzzy Greater Than
FGEQ	NFGEQ	Fuzzy Greater or Equal
FLT	NFLT	Fuzzy Less Than
FLEQ	NFLEQ	Fuzzy Less or Equal
MGT	NMGT	Much Greater Than
MLT	NMLT	Much Less Than

Table 1. Fuzzy Comparators for FSQL

- **Fulfilment Thresholds γ :** For each simple condition a Fulfilment threshold may be established with the format *<condition> THOLD γ* , indicating that the condition must be satisfied with a minimum degree γ in $[0,1]$ fulfilled.
- **CDEG(<attribute>) function:** This function shows a column with the Fulfilment degree of the condition of the query for a specific attribute, which is expressed in brackets as the argument.
- **Fuzzy Constants:** In FSQL we can use and store all of the fuzzy constants, which appear in Table 2.

F. Constant	Significance
UNKNOWN	Unknown value but the attribute is applicable
UNDEFINED	The attribute is not applicable or it is meaningless
NULL	Total ignorance: We know nothing about it
$A=[\alpha_A, \beta_A, \gamma_A, \delta_A]$	Fuzzy trapezoid ($\alpha_A \leq \beta_A \leq \gamma_A \leq \delta_A$): See Figure 1
\$label	Linguistic Label: It may be a trapezoid or a scalar (defined in FMB)
[n, m]	Interval "Between n and m" ($\alpha_A = \beta_A = n$ and $\gamma_A = \delta_A = m$)
#n	Fuzzy value "Approximately n" ($\beta_A = \gamma_A = n$ and $\alpha_A = \delta_A = \text{margin}$)

Table 2. Fuzzy Constants of FSQL

2.1 FRDB ARCHITECTURE OF FSQL

At present we have a FSQL Server available for Oracle© Databases, programmed in PL/SQL. Basically, the architecture of the FRDB with the FSQL Server is made up by:

1. **Data:** Traditional Database and Fuzzy Meta-knowledge Base (FMB).
2. **FSQL Server.**

2.1.1 Data: Traditional Database and FMB

The data can be classified in two categories:

- **Traditional Database:** They are data from our relations with a special format to store the fuzzy attribute values. The fuzzy attributes are classified by the system in 3 types:
 - Fuzzy Attributes **Type 1:** These attributes are totally crisp (traditional), but they have some linguistic trapezoidal labels defined on them, which allow us to make the query conditions for these attributes more flexible. Besides, we can use all constants in Table 2 in the query conditions with these fuzzy attributes.
 - Fuzzy Attributes **Type 2:** These attributes admit crisp data as well as possibility distributions over an ordered underlying domain. With these attributes we can store and use all the constants that we see in Table 2.
 - Fuzzy Attributes **Type 3:** These attributes have not an ordered underlying domain. On these attributes some labels are defined and on these labels a similarity relation has yet to be defined. With these attributes we can only use the fuzzy comparator FEQ, as they have no relation of order. Obviously we cannot store or use the constants fuzzy trapezoid, interval and approximate value of Table 2.
- **Fuzzy Meta-knowledge Base (FMB):** It stores information about the FRDB in a relational format. It stores attributes which admit fuzzy treatment and it will store different information for each one of them, depending on their type:
 - Fuzzy Attributes **Type 1:** In order to use crisp attributes in flexible queries we will only have to declare them as being a fuzzy attribute type 1 and store the following data in the FMB:
Trapezoidal linguistic labels: Name of the label and $\alpha_A, \beta_A, \gamma_A$ and δ_A values (as in Figure 1).
Value for the margin of the approximate values (see Table 1). Minimum distance in order to consider two values very separated (used in comparators MGT/NMGT and MLT/NMLT).
 - Fuzzy Attributes **Type 2:** As well as declare them as being a fuzzy attribute type 2, these attributes have to store the same data in the FMB as the fuzzy attributes Type 1.

- Fuzzy Attributes **Type 3**: They store in the FMB their linguistic labels, the similarity degree amongst themselves and the compatibility between attributes of this type, i.e., the attributes that use the same labels and that can be compared amongst themselves.

2.1.2 FSQ Server

It has been programmed entirely in PL/SQL and it includes 3 kinds of functions:

- **Translation Function**: It carries out a lexical, syntactic and semantic analysis of the FSQ query. If errors, of any kind whatsoever, are found, it will generate a table with all the found errors. If there are no errors, the FSQ query is translated into a standard SQL sentence. The resulting SQL sentence includes reference to the following kinds of functions.
- **Representation Functions**: These functions are used to show the fuzzy attributes in a comprehensible way for the user and not in the internally used format.
- **Fuzzy Comparison Functions**: They are utilized to compare the fuzzy values and to calculate the compatibility degrees (CDEG function).

3. EXTENDING FRDB ARCHITECTURE OF FSQ

We have already applied FSQ to DM processes [1,2,3,4]. Handling of different types of data is a necessary property for DM systems. Our objective is increasing the potential of FSQ in order to complete this property. Therefore we have extended the FRDB architecture at two levels:

1. **Data**: Traditional Database and FMB.

2. **FSQ Server**.

3.1.1 Data: Traditional Database and FMB

- **Traditional Database**: There are different kinds of data in a database used in diverse applications (relational data, objects, hypertext, etc.), therefore it would be desirable that a Data Mining system would carry out its work in an effective way. In order to manage these data we defined:
 - **Attributes Type 4**: These attributes are a generic type (fuzzy or crisp), which admit some fuzzy treatment. We permitted this attribute is formed by more than a column of the table. With these attributes we can store and use the constants linguistic label in Table 2.
- **Fuzzy Meta-knowledge Base (FMB)**: We have modified the FMB, such it stores information for the fuzzy treatment of the attributes Type 4.
 - **Fuzzy Comparison Functions**: The user can define the functions of comparison (Table 1) for the treatment of each attribute of Type 4. The format is: $CDEG(A \text{ } fcomp \text{ } B) \rightarrow [0,1]$ with $CDEG$ the compatibility degrees, A, B two attributes or linguistic labels Type 4 and $fcomp$ any fuzzy comparator in Table 1. The user can associate each attribute functions already defined in the FMB
 - **Representation Functions**: The user can optionally define it to show the attributes in a more comprehensible way. Of course the user can associate each attribute functions already defined in the FMB
 - **Linguistic labels**: They represent a concrete value of the attribute.
 - **Complex attributes**: We permitted this attribute is formed by more than a column of the table. Therefore the FMB stores information on structure of the attributes Type 4.

3.1.2 FSQ Server

As we have seen, Translation and Representation Functions are now in FMB for the attributes Type 4. Therefore the FSQ Server have been modified only in the next functionality:

- **Translation Function**: The lexical and syntactic analysis of the FSQ query for attributes Type 4 is the same that rest of attributes. We have modified the semantic analysis for such attributes. The FSQ query is

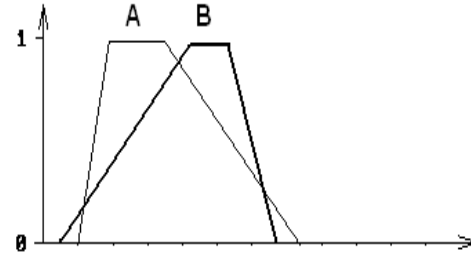


Figure 2. **Conflictive case to compare fuzzy numbers.**

translated into a standard SQL sentence using the enhanced FMB above explained. Therefore we can use new type of data only including information about it (functions, labels, etc.) in the FMB.

4. USING THE EXTENSION OF FRDB ARCHITECTURE OF FSQ

Now, we can use FSQ for any data type. We need only to have the comparison functions and to fill the FMB for this attribute (Type 4).

Thus by example, for an attribute Type 4 defined as several numeric columns of a Table we can defined on it FEQ comparators following: Euclidean Distance, Mahalonobi Distance, Sokal&Michener Distance (if all columns of the attribute have only binary values), etc.

Besides non-structured attributes (as strings, documents, graphics, etc.) can be used with FSQ if we can to define the comparison functions on it.

This also allows us have a more subjective vision of original FSQ comparators for possibility distributions. Following we have a new definition of this FSQ comparators for fuzzy attributes Type 2 based on Artificial Neural Networks (ANN). More information can be found in [4].

4.1 Fuzzy Comparator in FSQ using ANN

There are several methods to compare fuzzy attributes because the process of comparison of imprecise data is a diffuse process. Often this problem is not solved satisfactorily because it is difficult for a decision-maker to accept that some of the algorithmic methods in the literature [17] could adequately perform his own way to decide. Hence, the decision-maker would like to have a personal method, which implements his own criteria to compare fuzzy numbers especially in the face of conflictive cases like in Figure 2.

Solving this problem seems a very appropriate field for ANN because for most people it is easy to see that the trained ANN has learnt with examples given by him. That is not so clear with the algorithmic methods. Requena et al. [15] propose a method of ranking fuzzy numbers using ANN. Next we describe such method shortly.

The problem is to compare two trapezoidal possibility distributions A, B with $A=[\alpha_A, \beta_A, \gamma_A, \delta_A]$ $B=[\alpha_B, \beta_B, \gamma_B, \delta_B]$. Therefore, the inputs to the network are 8 (4 for each fuzzy number) and 1 output with 3 possible values: $A < B$, $A = B$ y $A > B$. The initial objective is to select an ANN structure capable to learn, and therefore to reproduce different comparison fuzzy numbers methods, considering for them the same topology and learning algorithm methods have coherent behaviour which may not be found in the human decision-makers, and because it is a simpler and more direct way of obtaining learning examples. If the learning is good from different comparison methods [12,16] with these characteristics, then we can deduce that this model can be used to learn the behaviour of any human decision-maker with a minimal level of coherence.

With the basic structures of 16 neurons with one hidden layer, 12+6 neurons for 2 hidden layers, initial values of 0.5 and 0.1 for the learning rate, backpropagation algorithm, and considering a basic learning set (with conflictive cases included) the authors obtain a sufficiently good learning of the behaviour of any real decision-maker, when comparing fuzzy numbers.

Now, it is necessary to relate the FSQ environment above explained with this ANN method. To do it, we will introduce a general definition of FSQ operators using ANN. We suppose that we want to compare two trapezoidal possibility distributions A and B (Figure 1) with $A=[\alpha_A, \beta_A, \gamma_A, \delta_A]$ $B=[\alpha_B, \beta_B, \gamma_B, \delta_B]$. CDEG is

used in FSQ to show a column with the fulfilment degree of the fuzzy condition. Therefore we will proceed to define CDEG for each fuzzy comparator using a procedure based on the ANN above explained.

Previously, we define the function:

$fcomp_ann(A, B, fcomp, dpc_fcomp_i)$

where:

- **fcomp** is a fuzzy comparator for FSQ (see Table 1)
- **dpc_fcomp_i** is the decision personal criteria of the user *i* for the comparator *fcomp*. This parameter contains all information above the ANN trained by que user *i* (weights matrix) for the comparator *fcomp*.

Thus the user *i* trains a network (based on the ANN above explained) for the comparator *fcomp* thought a set of examples (fuzzy numbers with intersection with conflictive cases includes). For this examples he decides if A *fcomp* B is true except whether the *fcomp* is FEQ then the user must give the degree fulfilment in this set {0.2, 0.4, 0.6, 0.8}. Therefore the possible values of *fcomp_ann* are:

- If $fcomp \neq \{FEQ\}$ then
 $fcomp_ann(A, B, fcomp, dpc_fcomp_i) =$
1, if A *fcomp* B with *dpc_fcomp_i*
0, otherwise
- If $fcomp = \{FEQ\}$ then
 $fcomp_ann(A, B, fcomp, dpc_fcomp_i) =$
{0.2, 0.4, 0.6, 0.8} degree fulfilment A FEQ B with *dpc_fcomp_i*

How we have seen, CDEG returns the fulfilment degree in [0,1] of the fuzzy condition. But the function *fcomp_ann* defined (for the comparators of inequality) only allows us ranking the possibility distributions A and B. Therefore we needed a method in order to express such fulfilment degree. In

order to express the distance between two fuzzy numbers, Requena et al. [15] define the decision personal index DPI as an application from the fuzzy numbers set to ranking in \mathbb{R} (real line). However when FSQ compare A and B it does not have information about the rest of fuzzy numbers which it is a dynamic information (possible values of the fuzzy attributes). In order to solve this problem we have developed the following method.

Firstly we define the function $df_ann(A, fcomp, dpc_fcomp_i)$ which returns a value in [0,1] which is the crisp value that represents to A with the criteria of the user *i* learnt by the ANN for the *fcomp* comparator:

$$df_ann(A, fcomp, dpc_fcomp_i) = (a_1 + a_2) / 2$$

where

$$a_1 = \inf \{a \in [\alpha_A, \delta_A] / fcomp_ann(a, A, fcomp, dpc_fcomp_i) = 1\}$$

$$a_2 = \sup \{a \in [\alpha_A, \delta_A] / fcomp_ann(A, a, fcomp, dpc_fcomp_i) = 1\}$$

For the implementation of this function we have used a method of binary search in order to find a_1 and a_2 . In practice in order to be more precise than the third decimal (for a_1 and a_2) place does not produce any improvement to represent A.

We have that $Ham(a, b)_{max, min}$ is a function which returns the absolute difference between two real numbers *a* and *b* previously convert to [0,1]:

$$Ham(a, b)_{max, min} =$$

$$|[(1 - \max/(\max - \min)) + a * (1 / (\max - \min))]$$

$$- [(1 - \max/(\max - \min)) + b * (1 / (\max - \min))]|$$

Now we proceed to define each one of the FSQ comparators considering that $\max = \max(\delta_A, \delta_B)$ and $\min = \min(\alpha_A, \alpha_B)$. The following definition can be incorporate without problems to the enhanced FMB of FSQ above explained:

$$CDEG(A \text{ FGT } B) =$$

$$\begin{cases} 1, & \text{if } \gamma_A \geq \delta_B \\ 0, & \text{if } \gamma_B \geq \delta_A \text{ or } fcomp_ann(A, B, FGT, dpc_FGT_i) = 0 \\ Ham_{\max, \min}(df_ann(A, FGT, dpc_FGT_i), \\ & df_ann(B, FGT, dpc_FGT_i)), & \text{otherwise} \end{cases}$$

$$\text{CDEG (A FGEQ B)} = \begin{cases} 1, & \text{if } \gamma_A \geq \beta_B \\ 0, & \text{if } \beta_B \geq \delta_A \\ & \text{or } fcomp_ann(A, B, FGEQ, dpc_FGEQ_i) = 0 \\ Ham_{\max, \min} [(df_ann(A, FGEQ, dpc_FGEQ_i), \\ & (df_ann(B, FGEQ, dpc_FGEQ_i)], \text{ otherwise} \end{cases}$$

$$\text{CDEG (A FEQ B)} = \begin{cases} 1, & \text{if } \alpha_A = \alpha_B \text{ and } \beta_A = \beta_B \text{ and } \gamma_A = \gamma_B \text{ and } \delta_A = \delta_B \\ 0, & \text{if } \delta_A < \alpha_B \text{ or } \delta_B < \alpha_A \\ fcomp_ann(A, B, FEQ, dpc_FEQ_i), & \text{ otherwise} \end{cases}$$

For the next comparator (MGT) we define $A_M = [\alpha_A - M, \beta_A - M, \gamma_A - M, \delta_A - M]$ with M is the minimum distance to consider two attributes as very separate. M is defined in FMB for each attribute.

$$\text{CDEG (A MGT B)} = \begin{cases} 1, & \text{if } \gamma_A \geq \delta_B + M \\ 0, & \text{if } \delta_A \leq \delta_B + M \\ & \text{or } fcomp_ann(A_M, B, FGT, dpc_FGT_i) = 0 \\ Ham_{\max, \min} [(df_ann(A_M, FGT, dpc_FGT_i), \\ & (df_ann(B, FGT, dpc_FGT_i)], \text{ otherwise} \end{cases}$$

This system (FSQL and ANN comparators) has been applied in a context of bank customers in a real life situation. The banking expert has identified relevant attributes: payroll, credit card use level and average account balance. Payroll is a binary attribute that indicates if the client receives payroll through the financial company (value Y) or not (value N). The level of use of the credit card is represented by fuzzy values obtained through an analytic study in the company data warehouse system. The average account balance store fuzzy values based on the average of the last 12 months.

In this context, the following kinds of queries are very useful and with a trained ANN for fuzzy comparators, the bank can obtain information from the database that it is not explicitly. Of course, the goodness of the resulting values depending on the training stage.

```
SELECT AccountNumber, CDEG(*)
FROM Customers
WHERE Payroll = 'N'
AND Credit_Card_Use FEQ $Medium
    THOLD 0.7
AND Balance FGT #300000
    THOLD 0.7
ORDER BY 2 DESC;
```

5. CONCLUSIONS

FSQL Server is an useful tool for DM process [1,2,3,4] and other applications [10,11]. However, it has a very important limitation, the handling of different types of data. In this paper we have showed an extension of FDBR architecture of FSQL that avoid this problem and make new definitions of operations and data very flexible. Thus, any user can define his own fuzzy comparator for any type of data. As example, we have defined fuzzy ANN comparator for any specific problem.

REFERENCES

- [1] R.A. Carrasco, J. Galindo, M.C. Aranda, J.M. Medina, M.A. Vila, "Classification in Databases using a Fuzzy Query Language". 9th International Conference on Management of Data, COMAD'98, Hyderabad (India), December 1998.
- [2] R.A. Carrasco, J. Galindo, M.A. Vila, J.M. Medina, "Clustering and Fuzzy Classification in a Financial Data Mining Environment". 3rd International ICSC Symposium on Soft Computing, SOCO'99, pp. 713-720, Genova (Italy), June 1999.
- [3] R.A. Carrasco, M.A. Vila, J. Galindo, J.C. Cubero, "FSQL: a Tool for Obtaining Fuzzy Dependencies". 8th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU'2000, pp. 1916-1919. Madrid (Spain), July 2000.
- [4] R. Carrasco, J. Galindo, A. Vila, "Using Artificial Neural Network to Define Fuzzy Comparators in FSQL with the Criterion of some Decision-Maker". In "Bio-inspired applications of connectionism.-2001", eds. J. Mira and A. Prieto, Lecture Notes in Computer Science (LNCS) 2085, pp. 587-594. Ed. Springer-Verlag, 2001, ISBN: 3-540-42237-4.
- [5] M. Chen, J. Han, P.S. Yu, "Data Mining: An overview from a Data Base Perspective". IEEE Transac. On Knowledge and Data Engineering, Vol 8-6 pp. 866-883, 1996.
- [6] M.A. Vila, J.C. Cubero, J.M. Medina, O. Pons, "On the use of Soft Computing Techniques in Data Mining Problems". Technical report #DECSAI-96-01-12. Department of Computer Science and Artificial Intelligence. Universidad de Granada. Spain (1997).
- [7] W.J. Frawley, G. Piatetsky-Shapiro, C.J. Matheus, "Knowledge Discovery in Databases: An Overview" in G. Piatetsky-Shapiro, W.J. Frawley eds. "Knowledge Discovery in Databases" pp. 1-31, The AAAI Press, 1991.
- [8] J. Galindo, J.M. Medina, O. Pons, J.C. Cubero, "A Server for Fuzzy SQL Queries", in "Flexible Query Answering Systems", eds. T. Andreasen, H. Christiansen and H.L. Larsen, Lecture Notes in Artificial Intelligence (LNAI) 1495, pp. 164-174. Ed. Springer, 1998.
- [9] J. Galindo, J.M. Medina, A. Vila, O. Pons, "Fuzzy Comparators for Flexible Queries to Databases". Iberoamerican Conference on Artificial Intelligence, IBERAMIA'98, pp. 29-41, Lisbon (Portugal), October 1998.
- [10] J. Galindo, "Tratamiento de la Imprecisión en Bases de Datos Relacionales: Extensión del Modelo y Adaptación de los SGBD Actuales". Ph. Doctoral Thesis, University of Granada (Spain), March 1999. www.lcc.uma.es
- [11] J. Galindo, J.M. Medina, J.C. Cubero, O. Pons, "Management of an Estate Agency Allowing Fuzzy Data and Flexible Queries". EUSFLAT-ESTYLF Joint Conference, pp. 485-488, Palma de Mallorca (Spain), September 1999.
- [12] R. Jain, "Tolerance Analysis using fuzzy sets". Internat. J. Systems Sci, 7, pp 1393-1401, 1976.
- [13] J.M. Medina, O. Pons, M.A. Vila, "GEFRED. A Generalized Model of Fuzzy Relational Data Bases". Information Sciences, 76(1-2), pp. 87-109, 1994.
- [14] F.E. Petry, "Fuzzy Databases: Principles and Applications" (with chapter contribution by Patrick Bosc). International Series in Intelligent Technologies. Ed. H.-J. Zimmermann. Kluwer Academic Publishers (KAP), 1996.
- [15] I. Requena, M. Delgado, J.L. Verdegay, "Artificial neural networks learn the criteria of a real decision-maker to compare fuzzy numbers". Fuzzy Sets and Systems 64, pp 1-19, 1994.
- [16] R.R. Yager, "Ranking fuzzy subsets over the unit interval". Proc. CDC, pp 1435-1437, 1978.
- [17] Q. Zhu, E.S. Lee, "Comparison and ranking of fuzzy numbers". Fuzzy Regression Analysis. J. Kacprzyk and Fedrizzi, eds. Omnitech Press, Warsaw, Poland, pp 21-44, 1991.