Mathware & Soft Computing 7 (2000) 309-322

A GA-P Algorithm to Automatically Formulate Extended Boolean Queries for a Fuzzy Information Retrieval System

O. Cordón[1], F. de Moya[2], C. Zarco[3]
[1]Dept. of Computer Science and A.I. E.T.S. de Ingeniería Informática
University of Granada. 18071 - Granada (Spain)
[2]Dept. of Librarianship. Faculty of Librarianship and Documentation
University of Granada. 18071 - Granada (Spain)
[3]PULEVA Salud S.A.
Camino de Purchil, 66. 18004 - Granada (Spain)
ocordon@decsai.ugr.es, felix@ugr.es, czarco@puleva.es

Abstract

Although the fuzzy retrieval model constitutes a powerful extension of the boolean one, being able to deal with the imprecision and subjectivity existing in the Information Retrieval process, users are not usually able to express their query requirements in the form of an extended boolean query including weights. To solve this problem, different tools to assist the user in the query formulation have been proposed.

In this paper, the genetic algorithm-programming technique is considered to build an algorithm of this kind that will be able to automatically learn weighted queries —modeling the user's needs— for a fuzzy information retrieval system by applying an off-line adaptive process starting from a set of relevant documents.

Keywords: Fuzzy information retrieval, GA-P algorithms, weighted queries, automatic query learning.

1 Introduction

Information retrieval (IR) may be defined, in general, as the problem of the selection of documentary information from storage in response to search questions provided by an user [15]. Information retrieval systems (IRSs) are a kind of information system that deal with data bases composed of information items —documents that may consist of textual, pictorial or vocal information— and process user queries

trying to allow the user to access to relevant information in an appropiate time interval.

Most of the commercial IRSs are based on the Boolean IR model [18], which presents some limitations. Due to this fact, some paradigms have been designed in the last few years to extend this retrieval model and overcome its problems without a need of a complete redesign. One of the main drawbacks is that Boolean IRSs lack the ability to deal with imprecision and subjectivity, both of which are inherently present in the IR process. This is the reason why fuzzy information retrieval (FIR) [2, 4] has emerged as an active research area in the past decade.

FIRSs present a query language that allows the user to build queries composed of statements of numericaly or linguisticaly weighted terms (that can be negated by means of the operator NOT) joined by the Boolean operators AND and OR. This query structure —and other aspects that will be reviewed in the next section— allow the system to improve the retrieval activity solving the problems associated with the Boolean model.

Nevertheless, the use of weights requires a clear knowledge of the semantics of the query in order to translate a fuzzy concept into a crisp numeric value in [0,1]. On the other hand, both in Boolean and FIRSs, it is difficult for non-expert users to express their retrieval needs in the form of a query involving different statements joined by the logical operators AND and OR. To solve these problems in different kinds of IRSs, several approaches have been applied to assist the user in the query formulation process [3]. One of them is based on automatically generating the query best describing the user's needs —represented in the form of an initial set of relevant (and optionally non relevant) documents— by means of an off-line process in which no user interaction is required. This operation mode is included in the usual *machine learning paradigm* and has been called *inductive query by example* (IQBE) by Chen [3].

Genetic algorithms (GAs) [14] have been succesfully applied to formulate queries for different IR models (see [6] for a review including this and other different applications of GAs to IR). Focusing on the FIR model, one of the most known approaches to learn fuzzy queries is Kraft et al.'s proposal [13], which is an IQBE algorithm based on genetic programming.

In this paper, an automatic learning process based on the genetic algorithm-programming (GA-P) paradigm [11], which is an extension of the previous proposal, will be introduced. To do so, the paper is structured as follows. Section 2 is devoted to introduce the preliminaries, including the basis of boolean IRSs and of FIRSs. Kraft et al.'s proposal is reviewed in Section 3. Section 4 presents the composition of the new algorithm proposed while the different experiments developed to test it are shown in Section 5. Finally, several concluding remarks and suggerences for future work are pointed out in Section 6.

2 Preliminaries: Boolean and Fuzzy Information Retrieval Systems

2.1 Boolean Information Retrieval Systems

An IRS is basically constituted of three main components:

1. A *documentary data base*, which stores the documents and the representation of their information contents. It is associated with the *indexer module*, which automatically generates a representation for each document by extracting the document contents. Textual document representation is typically based on index terms (that can be either single terms or sequences) which are the content identifiers of the documents.

2. A *query subsystem*, which allows the users to formulate their queries and presents the relevant documents retrieved by the system to them. To do so, it includes a *query language*, that collects the rules to generate legitimate queries and procedures to select the relevant documents.

3. A *matching or evaluation mechanism*, which evaluates the degree to which the document representations satisfy the requirements expressed in the query and retrieves those documents that are judged to be relevant to it.

In the Boolean retrieval model, the indexer module performs a binary indexing in the sense that a term in a document representation is either significant (appears at least once in it) or not (it does not appear in it at all). On the other hand, user queries in this model are expressed using a query language that is based on these terms and allows combinations of simple user requirements with logical operators AND, OR and NOT [15, 18]. The result obtained from the processing of a query is a set of documents that totally match with it, i.e., only two possibilities are considered for each document: to be or not to be relevant for the user's needs, represented by the user query.

Thus, the Boolean model presents several problems that are located in the different Boolean IRS components such us:

- It does not provide the user with tools to express the degree of relevance of the index terms to the documents (*indexer module*).

- It has no method to express a user's judgement of the importance of the terms in the query (*query language*).

- There are no partial degrees of relevance of documents to queries possibly useful in ranking (*matching mechanism*).

2.2 Fuzzy Information Retrieval Systems

Trying to solve the problems of the Boolean IR model introduced at the end of the previous subsection, FIR mainly extends it in three aspects:

1. Document representations become fuzzy sets defined in the universe of terms, and terms become fuzzy sets defined in the universe of discourse of documents, thus introducing a degree of relevance (aboutness) between a document and a term.

2. Numeric weights (and in recenty proposals, linguistic terms [2, 10]) are considered in the query with different semantics (a review of them all is to be found in [2]), thus allowing the user to quantify the "subjective importance" of the selection requirements.

3. Since the evaluation of the relevance of a document to a query is also an imprecise process, a degree of document relevance is introduced, the so called *retrieval status value* (RSV). To do so, the classical complete matching approach and Boolean set operators are modeled by means of fuzzy operators appropriately performing the matching of queries to documents in a way that preserves the semantics of the former.

Thus, the operation mode of the three components of an FIRS is shown as follows.

Indexer Module: Let D be a set of documents and T be a set of unique and significant terms existing in them. The indexer module of the FIRS defines an indexing function:

$$F : D \times T \rightarrow [0,1]$$

It can be seen that F is the membership function of a two-dimensional fuzzy set (a fuzzy relation) mapping the degree to which document d belongs to the set of documents "about" the concept(s) represented by term t. By projecting it, a fuzzy set can be associated to each document and term:

$$d_i = \{< t, \mu_{d_i}(t) > \mid t \in T\} \quad ; \quad \mu_{d_i}(t) = F(d_i, t)$$
$$t_j = \{< d, \mu_{t_j}(d) > \mid d \in D\} \quad ; \quad \mu_{t_j}(d) = F(d, t_j)$$

There are different ways to define the indexing function F. In this paper, we will work with the normalized *inverted document frequency* [15]:

$$w_{d,t} = f_{d,t} \cdot log(N/N_t) \qquad ; \qquad F(d,t) = \frac{w_{d,t}}{Max_d\, w_{d,t}}$$

where $f_{d,t}$ is the frequency of term t in document d, N is the number of documents in the collection and N_t is the number of documents where term t appears at least once.

Matching mechanism: It operates in a different way depending on the interpretation associated to the numeric weights included in the query (the interested reader can refer to [2, 4] to get knowledge about the three existing approaches). In

this paper, we consider the *importance* interpretation, where the weights represent the relative importance of each term in the query.

In this case, the RSV of each document to a fuzzy query q is computed as follows [16]. When a single term query is logically connected to another by means of the AND or OR operators, the relative importance of the single term in the compound query is taken into account by associating a weight to it. To maintain the semantics of the query, this weighting has to take a different form according as the single term queries are ANDed or ORed. Therefore, assuming that A is a fuzzy term with assigned weight w, the following expressions are applied to obtain the fuzzy set associated to the weighted single term queries A_w (in the case of *disjunctive queries*) and A^w (for *conjunctive ones*):

$$A_w = \{< d, \mu_{A_w}(d) > \mid d \in D\} \quad ; \quad \mu_{A_w}(d) = Min\ (w, \mu_A(d))$$

$$A^w = \{< d, \mu_{A^w}(d) > \mid d \in D\} \quad ; \quad \mu_{A^w}(d) = Max\ (1 - w, \mu_A(d))$$

On the other hand, if the term is negated in the query, a negation function is applied to obtain the corresponding fuzzy set:

$$\overline{A} = \{< d, \mu_{\overline{A}}(d) > \mid d \in D\} \quad ; \quad \mu_{\overline{A}}(d) = 1 - \mu_A(d)$$

Once all the single weighted terms involved in the compound query have been evaluated, the fuzzy set representing the RSV of the compound query is obtained by combining them into a single fuzzy set by means of the following operators:

$$A\ AND\ B = \{< d, \mu_{A\ AND\ B}(d) > \mid d \in D\} \quad ; \quad \mu_{A\ AND\ B}(d) = Min(\mu_A(d), \mu_B(d))$$

$$A\ OR\ B = \{< d, \mu_{A\ OR\ B}(d) > \mid d \in D\} \quad ; \quad \mu_{A\ OR\ B}(d) = Max(\mu_A(d), \mu_B(d))$$

We should note that all the previous expressions can be generalized to work with any other t-norm, t-conorm and negation function different from the usual minimum, maximum and one-minus function. In this contribution, we will consider these ones.

Query Subsystem: It affords a fuzzy set q defined on the document domain specifying the degree of relevance of each document in the data base with respect to the processed query:

$$q = \{< d, \mu_q(d) > \mid d \in D\} \quad ; \quad \mu_q(d) = RSV_q(d)$$

Thus, one of the advantages of the FIRSs is that documents can be ranked in order to the membership degrees of relevance —as IRSs based on the vector space model [15]— before being presented to the user as query response. The final relevant document set can be specified by him in two different ways: providing an upper bound for the number of retrieved documents or defining a threshold σ for the relevance degree (as can be seen, the latter involves obtaining the σ-cut of the query response fuzzy set q).

3 The Kraft et al.'s Genetic Programming-based Fuzzy Query Learning Algorithm

In [13], Kraft et al. proposed an IQBE process to learn the whole composition of extended Boolean queries (terms, weights and logical operators) for an FIRS. It is based on a variant of GAs, genetic programming (GP) [12], which evolves structures encoding programs such as expression trees. The algorithm components are described next[1].

Coding Scheme: The fuzzy queries are encoded in expression trees, whose terminal nodes are query terms with their respective weights and whose inner nodes are the Boolean operators *AND*, *OR* or *NOT*.

Selection Scheme: It is based on the classical generational scheme, where an intermediate population is created from the current one by means of Baker's stochastic universal sampling [1], together with the elitist selection.

Genetic Operators: The usual GP crossover is considered [12], which is based on randomly selecting one edge in each parent and exchanging both subtrees from these edges between the both parents.

On the other hand, the following three possibilities are randomly selected — with the shown probability— for the GP mutation:

a) Random selection of an edge and random generation of a new subtree that substitutes the old one located in that edge (p=0.4).

b) Random change of a query term for another one, not present in the encoded query, but belonging to any relevant document (p=0.1).

c) Random change of the weight of a query term (p=0.5).

For the latter case, *Michalewicz's non-uniform mutation operator* [14] is considered. It is based on making a uniform search in the initial space in the early generations, and a very local one in later stages. Let $C = (c_1, \ldots, c_k, \ldots, c_H)$ be the parent and c_k be the gene selected for mutation (the domain of c_k is $[c_{kl}, c_{kr}]$), the new value for this gene is

$$c_k^{'} = \begin{cases} c_k + \triangle(t, c_{kr} - c_k) & \text{if } a = 0, \\ c_k - \triangle(t, c_k - c_{kl}) & \text{if } a = 1 \end{cases}$$

where $a \in \{0, 1\}$ is a random number and the function $\triangle(t, y)$ returns a value in the range $[0, y]$ such that the probability of $\triangle(t, y)$ being close to 0 increases as the number of generations increases.

[1] Notice that the composition of several components is not the original one proposed by Kraft et al. but they have been changed in order to improve the algorithm performance. Of course, the basis of the algorithm have been maintained.

Generation of the Initial Population: A first individual is obtained by generating a random tree representing a query with a maximum predefined length and composed of randomly selected terms existing in the initial relevant documents provided by the user, and with all the term weights set to 1. The remaining individuals are generated in the same way but with random weights in [0,1].

Fitness function: Two different possibilities are considered based on the classical precision and recall measures (to get more information about them, see [18]):

$$F_1 = \frac{\sum_d r_d \cdot f_d}{\sum_d r_d} \quad ; \quad F_2 = \alpha \cdot \frac{\sum_d r_d \cdot f_d}{\sum_d f_d} + \beta \cdot \frac{\sum_d r_d \cdot f_d}{\sum_d r_d}$$

with $r_d \in \{0,1\}$ being the relevance of document d for the user and $f_d \in \{0,1\}$ being the retrieval of document d in the processing of the current query. Hence, F_1 only considers the recall value obtained by the query, while F_2 also takes its precision into account.

Although the algorithm proposed by Kraft et al. obtains good results, it suffers from one of the main limitations of the GP paradigm: while it performs really well in the generation of structures, adapting them both by crossover and mutation, the learning of the numeric values of the constants considered in the encoded structure —which are generated by the implementation program when the GP starts— can only be altered by mutation. Hence, in the problem of weighted query learning, the GP algorithm is able to find the positive, or negative, terms expressing the user's needs and to appropriately combine them by means of the logical operators AND and OR. However, it is very difficult for the algorithm to obtain the term weights, which constitutes an important drawback due to their importance in the query. In the next section, we will propose an extension of Kraft et al.'s proposal to solve this problem and improve the query formulation process.

4 A GA-P Algorithm to Learn Fuzzy Queries

The problem introduced in the previous section can be solved by means of another evolutionary algorithm, the GA-P [11]. This paradigm is based on combining traditional GAs with the GP technique to evolve complex expressions capable of handling numeric and symbolic data. Each population member will involve both a value string and an expression. While the GP part of the GA-P evolves the expressions, the GA part concurrently evolves the coefficients used in them.

Most of the GA-P's elements are the same as in either of the traditional genetic techniques. The GA-P and GP make selection and child generation similarly, except that the GA-P's structure requires separate crossover and mutation operators for the expression and coefficient string components. Mutation and crossover rates for the coefficient string (using traditional GA methods) are independent from the rates for the expression part (using standard GP methods).

The different components of the proposed algorithm are shown as follows.

Coding Scheme: When considering a GA-P to learn fuzzy queries[2], the expressional part (GP part) encodes the query composition —terms and logical operators— and the coefficient string (GA part) represents the term weights, as shown in Figure 1. In our case, a real coding scheme is considered for the GA part.

Figure 1: GA-P individual representing the fuzzy query $0.5\ t_1\ AND\ (0.7\ t_3\ OR\ 0.25\ t_4)$. Aditionally, the last string coefficient can be used to learn the retrieval threshold ($\sigma = 0.3$)

An important extension to Kraft et al.'s algorithm is that we also allow the system to automatically learn the relevance threshold for the retrieved documents, which is usually a difficult choice for the user. This value is encoded in the last string coefficient in case the user decides to consider this capability.

Selection, Niching and Replacement Scheme: In our algorithm, as opposed to Kraft et al.'s, the selection is based on the steady-state approach [14]. Hence, no intermediate population is needed and each generation involves selecting two parents, crossing and mutating them, and introducing the two offsprings in the current population only if they improve the individuals they are substituting.

On the other hand, although the GA-P is a powerful technique to learn the coefficients in the expression, the strong relationship existing between the value string and the expressional part make it perform better if the individuals involved in the crossover have a similar GP part. A previous version of the algorithm proposed in this paper did not consider this fact, thus obtaining fuzzy queries of medium quality [7].

In [17], it is shown that the said behaviour can be obtained by inducing niches in the GA-P population. The *niche* concept [8] was introduced to avoid having the GA converge to a single space zone when dealing with multimodal fitness functions — a phenomenon known as *genetic drift*—. The key idea is the formation of stable

[2] For other applications of the GA-P paradigm to the field of fuzzy systems, refer to [5, 17].

subpopulations of similar individuals —that evolve in parallel— in different space zones. This way, the GA has more chances to obtain the optimal solution to the problem.

As in [17], every individual having the same expressional part in our algorithm belong to the same niche. Thus, all the individuals in a subpopulation share the same query composition, and we only encode different weights for the query terms involved in it.

Two different kinds of crossover are considered, depending on whether the two parents to be crossed belong or not to the same niche. *Intra-niche crossovers* are performed between individuals in the same niche, and in this case only the GA parts are crossed in order to look for better term weights for the fixed query expression encoded in the GP part. In this sense, their function is to exploit the GA search space zone where the niche is located. On the other hand, *inter-niche crossovers* are performed between individuals belonging to different niches, and thus both parts are crossed, creating individuals with a different GP parts (i.e., fuzzy queries with a different composition). Since new niches can be generated in this way (when a new expressional part —query composition— different from the ones in the existing niches appear), this crossover operator leads to an exploration of the whole GA-P search space by the introduction of diversity in the population.

The algorithm operation mode is as follows. First, a single individual is selected at random with a probability that grows with his adaption level —we consider the proportional probability assigment scheme– by means of the universal stochastic procedure —the usual roulete wheel— [14]. Then, the kind of crossover to be performed is randomly selected according to a probability P_{intra_cross}:

- If an intra-niche crossover is to be done, another individual belonging to the same niche and different from the previous one is selected in the same way. The crossover in the GA part is then performed and both GA parts are also mutated with probability P_m^{GA}. Finally, a competition is established among the two parents and the two offsprings, and the two best of them take the place of both parents in the population. It is important to note that when two individuals have the same fitness value, simpler queries are preferred.

- Otherwise, if an inter-niche crossover is chosen, another individual is selected that belongs to a different niche. The crossover is then performed on both parts (GA and GP) and later both GP parts and GA parts in the offsprings are mutated with probilities P_m^{GP} and P_m^{GA}, respectively. In this case, the two offsprings compete with the two worst individuals in the population for the places of the latter two. Thus, they can create new niches and cause existing niches to grow by incorporating new individuals or to decrease their size if their individuals are not well adapted enough.

Genetic Operators: A real-coded crossover operator —the BLX-α [9]— is considered for the GA parts in the intra-niche crossover. This operator generates an offspring, $C = (c_1, \ldots, c_n)$, from two parents, $X = (x_1, \ldots, x_n)$ and $Y =$

$(y_1, \ldots, y_n)$, with c_i being a randomly (uniformly) chosen number from the interval $[min_i - I \cdot \alpha, max_i + I \cdot \alpha]$, where $max_i = max\{x_i, y_i\}$, $min_i = min\{x_i, y_i\}$, and $I = max_i - min_i$ ($[min_i, max_i]$ is the interval where the $i-th$ gene is defined. In our case, $[min_i, max_i] = [0, 1]$, and the operator is applied twice to obtain two offsprings.

On the other hand, *Michalewicz's non-uniform mutation operator*, introduced in the previous section, is considered to perform mutation in the GA part.

As regards the operators for the GP part, the usual GP crossover described in the previous section is used to perform the inter-niche crossover, while the two first GP mutation operators (a) and b)) considered by the Kraft et al.'s algorithm are employed with probability 0.5 each.

Generation of the Initial Population and Fitness Function: These both have the same definition as that in Kraft et al.'s proposal, introduced in the previous section.

5 Experiments Developed and Analysis of Results

To test the performance of the proposed algorithm, we have followed a similar experimental methodology as that in [13]. A documentary base, composed of 359 abstracts taken from the *Library and Information Science Abstracts (LISA)* data base, has been automatically indexed, obtaining a total number of 2609 different indexing terms. A user has selected two sets of 82 and 8 relevant documents, *query 1* and *query 2*, respectively, that have been provided to both Kraft et al.'s and our system (noted by GAP-FIR), which have been run considering the second fitness function F_2. In order to make a fair comparison, both algorithms have been provided with the same parameter values (see Table 1) and have been run three times with different initializations till the same fixed number of fitness function evaluations have been performed.

Table 1: Common parameter values considered

Parameter	Decision
Population size	1600
Number of evaluations	100000
Kraft et al.'s GA Crossover and Mutation probability	0.8, 0.2
GAP-FIR GA and GP Mutation probability	0.2, 0.2
GAP-FIR Intra-niche Cross. probability	0.25
Expression part limited to	10 nodes
Retrieval threshold σ (when not learned)	0.5
Weighting coefficientes α, β in F_2	(0.8,1.2), (1,1), (1.2,0.8)

Since simple queries are desired, the expressional part has been limited to 10

nodes in every case. For the sake of simplicity, only the experiments not considering the use of the NOT operator are reported (as done in [13]). On the other hand, different values for the parameters α and β weighting, respectively, the precision and recall measures in the F_2 fitness functions, are considered. For simplicity, only the best results are reported [3].

The results obtained are shown in Tables 2 to 5, where *Run* stands for the corresponding algorithm run (1 to 3), Sz for the generated query size, Fit for the fitness value, P and R for the precision and recall values, respectively, $\#rt$ for the number of documents retrieved by the learned query, and $\#rr$ for the number of relevant documents retrieved. In the tables associated to our GAP-FIR proposal (3 and 5), column Thr shows whether the automatic learning of the retrieval threshold is considered (Y) or not (N).

Table 2: Results obtained by Kraft et al.'s method in *query 1*

Run	α	β	Sz	Fit	P	R	$\#rr/\#rt$
1	1.2	0.8	7	1.800000	1.000000	0.750000	6/6
2	1.2	0.8	9	1.800000	1.000000	0.750000	6/6
3	1.2	0.8	9	1.800000	1.000000	0.750000	6/6
1	1.0	1.0	7	1.750000	1.000000	0.750000	6/6
2	1.0	1.0	9	1.750000	1.000000	0.750000	6/6
3	1.0	1.0	9	1.750000	1.000000	0.750000	6/6

Table 3: Results obtained by GAP-FIR method in *query 1*

Run	α	β	Thr	Sz	Fit	P	R	$\#rr/\#rt$
1	1.2	0.8	N	9	2.000000	1.000000	1.000000	8/8
2	1.2	0.8	N	9	2.000000	1.000000	1.000000	8/8
3	1.2	0.8	N	9	2.000000	1.000000	1.000000	8/8
1	1.2	0.8	Y	7	2.000000	1.000000	1.000000	8/8
2	1.2	0.8	Y	7	2.000000	1.000000	1.000000	8/8
3	1.2	0.8	Y	7	2.000000	1.000000	1.000000	8/8
1	1.0	1.0	N	9	2.000000	1.000000	1.000000	8/8
2	1.0	1.0	N	9	2.000000	1.000000	1.000000	8/8
3	1.0	1.0	N	9	2.000000	1.000000	1.000000	8/8
1	1.0	1.0	Y	7	2.000000	1.000000	1.000000	8/8
2	1.0	1.0	Y	7	2.000000	1.000000	1.000000	8/8
3	1.0	1.0	Y	7	2.000000	1.000000	1.000000	8/8

In view of these results, our proposal performs suitably, since it always obtains better results than Kraft et al.'s. Moreover, it can be clearly seen that the au-

[3] For example, every experiment developed with a weight combination different from $(\alpha, \beta) = (1.2, 0.8)$ always generates a query retrieving the whole document collection (359 documents).

Table 4: Results obtained by Kraft et al.'s method in *query 2*

Run	α	β	*Sz*	*Fit*	*P*	*R*	*#rr/#rt*
1	1.2	0.8	9	1.409214	0.962963	0.317073	26/27
2	1.2	0.8	7	1.395122	1.000000	0.243902	20/20
3	1.2	0.8	9	1.414634	1.000000	0.268293	22/22

Table 5: Results obtained by GAP-FIR method in *query 2*

Run	α	β	*Thr*	*Sz*	*Fit*	*P*	*R*	*#rr/#rt*
1	1.2	0.8	N	9	1.508130	0.972222	0.426829	35/36
2	1.2	0.8	N	9	1.508130	0.972222	0.426829	35/36
3	1.2	0.8	N	9	1.508130	0.972222	0.426829	35/36
1	1.2	0.8	Y	9	1.577094	0.956522	0.536585	44/46
2	1.2	0.8	Y	9	1.544186	0.953488	0.500000	41/43
3	1.2	0.8	Y	9	1.552781	0.920000	0.560976	46/50

tomatic learning of the retrieval threshold is a simple way to improve the system performance, since simpler queries are obtained for *query 1* and more accurate ones for *query 2* when this capability is considered. As regards the run time, both algorithms last more or less the same, approximately three minutes for *query 1* and four in the case of *query 2*.

6 Concluding Remarks

An IQBE algorithm based on the GA-P paradigm has been proposed to assist the user in the design of weighted queries for FIRSs. Its performance has been tested by learning queries for two different document collections provided by an user and the results obtained were significantly good, outperforming the previous proposal by Kraft et al.

In the future, we are thinking of considering multi-objective evolutionary algorithms to extend the process proposed by avoiding the need to weight both fitness function criteria, precision and recall, thus generating different fuzzy queries with an appropriate balance between both criteria in each run.

Acknowledgements

The authors would like to thank Dr. Luciano Sánchez from Oviedo University for his valuable suggestions that have helped the proposed algorithm take its current form.

References

[1] J.E. Baker, Reducing bias and inefficiency in the selection algorithm, Proc. Second International Conference on Genetic Algorithms (ICGA'87), Hillsdale, 14-21 (1987).

[2] G. Bordogna, P. Carrara, G. Pasi, Fuzzy approaches to extend Boolean information retrieval, in: P. Bosc, J. Kacprzyk (Eds.), Fuzziness in database management systems 231-274 (1995).

[3] H. Chen, A machine learning approach to inductive query by examples: an experiment using relevance feedback, ID3, genetic algorithms, and simulated annealing, Journal of the American Society for Information Science **49:8** 693-705 (1998).

[4] V. Cross, Fuzzy information retrieval, Journal of Intelligent Information Systems **3** 29-56 (1994).

[5] O. Cordón, F. Herrera, L. Sánchez, Solving electrical distribution problems using hybrid evolutionary data analysis techniques, Applied Intelligence **10:1** 5-24 (1999).

[6] O. Cordón, F. Moya, M.C. Zarco, A brief study on the application of genetic algorithms to information retrieval (in spanish), Proc. Fourth International Society for Knowledge Organization (ISKO) Conference (EOCONSID'99), Granada, Spain, 179-186 (April, 1999).

[7] O. Cordón, F. Moya, M.C. Zarco, Learning queries for a fuzzy information retrieval system by means of GA-P techniques, Proc. EUSFLAT-ESTYLF Joint Conference, Palma de Mallorca, Spain, 335-338 (September, 1999).

[8] K. Deb, D.E. Goldberg, An investigation of niche and species formation in genetic function optimization, Proc. Second International Conference on Genetic Algorithms (ICGA), Hillsdale, EEUU, 42-50 (1989).

[9] L.J. Eshelman, J.D. Schaffer, Real-coded genetic algorithms and interval-schemata, in: L.D. Whitley (Ed.), Foundations of Genetic Algorithms 2, Morgan Kaufmann 187-202 (1993).

[10] E. Herrera-Viedma, Modelling the retrieval process of an information retrieval system using an ordinal fuzzy linguistic approach, Journal of the American Society for Information Science **52:6** 460-475 (2001).

[11] L. Howard, D. D'Angelo, The GA-P: a genetic algorithm and genetic programming hybrid, IEEE Expert, 11-15 (1995).

[12] J. Koza, Genetic programming. On the programming of computers by means of natural selection, The MIT Press (1992).

[13] D.H. Kraft, F.E. Petry, B.P. Buckles, T. Sadasivan, Genetic algorithms for query optimization in information retrieval: relevance feedback, in: E. Sanchez, T. Shibata, L.A. Zadeh, Genetic algorithms and fuzzy logic systems, 155-173 (1997).

[14] Z. Michalewicz, Genetic algorithms + data structures = evolution programs, Springer-Verlag (1996).

[15] G. Salton, M.J. McGill, Introduction to modern information retrieval, McGraw-Hill (1989).

[16] E. Sanchez, Importance in knowledge systems, Information Systems **14:6** 455-464 (1989).

[17] L. Sánchez, J.A. Corrales, A niching scheme for steady state GA-P and its application to fuzzy rule based classifiers induction, Mathware & Soft Computing, this issue.

[18] C.J. van Rijsbergen, Information Retrieval (2nd edition), Butterworth (1979).