Focus

Soft Computing 6 (2002) 308 – 319 © Springer-Verlag 2002
DOI 10.1007/s00500-002-0184-8

A new evolutionary algorithm combining simulated annealing and genetic programming for relevance feedback in fuzzy information retrieval systems

O. Cordón, F. Moya, C. Zarco

Abstract Relevance feedback techniques have demonstrated to be a powerful means to improve the results obtained when a user submits a query to an information retrieval system as the world wide web search engines. These kinds of techniques modify the user original query taking into account the relevance judgements provided by him on the retrieved documents, making it more similar to those he judged as relevant. This way, the new generated query permits to get new relevant documents thus improving the retrieval process by increasing recall. However, although powerful relevance feedback techniques have been developed for the vector space information retrieval model and some of them have been translated to the classical Boolean model, there is a lack of these tools in more advanced and powerful information retrieval models such as the fuzzy one. In this contribution we introduce a relevance feedback process for extended Boolean (fuzzy) information retrieval systems based on a hybrid evolutionary algorithm combining simulated annealing and genetic programming components. The performance of the proposed technique will be compared with the only previous existing approach to perform this task, Kraft et al.'s method, showing how our proposal outperforms the latter in terms of accuracy and sometimes also in time consumption. Moreover, it will be showed how the adaptation of the retrieval threshold by the relevance feedback mechanism allows the system effectiveness to be increased.

Keywords Fuzzy information retrieval, Relevance feedback, Evolutionary algorithms, Genetic programming, Simulated annealing

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 [30]. 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. Nowadays, the world wide web constitutes the main example of an IRS.

The performance of an IRS is usually measured in terms of *precision* and *recall* [30, 36]. When a user submits a query to the system, a set of documents is retrieved that can really match or not the user's information needs. Recall is the ratio of the number of relevant documents retrieved in that set by the total number of documents in the data base that are relevant for the user. On the other hand, precision is the ratio of the number of relevant documents retrieved to the number of documents retrieved. The ideal goal would be to maximize both recall and precision, which are contradictory requirements in practice.

With this aim in mind, several IR models have been proposed. On the one hand, most of the commercial IRSs used in corporate intranet, as well as many Internet search engines, are based on the classical Boolean IR model [36], which presents some limitations. Due to this fact, some paradigms have been designed to extend this retrieval model and overcome its problems, such as the vector space [30] or the fuzzy IR (FIR) models [4, 6].

FIRSs are based on the fuzzy set theory as enunciated by L.A. Zadeh in 1965 [39]. In this framework, fuzzy sets are used as a means to deal with the imprecision and vagueness which is always present in the IR activity. This fact is making the use of fuzzy tools more recognized in the IR field in the last few years.

However, even the best of the IRSs have a limited recall, i.e., the formulated queries allow the users to retrieve some relevant documents but almost never all the relevant documents in the data base being relevant to their information needs. This is a very common fact in the Internet, where usual search engines such as Altavista have low degrees of precision and recall [17].

To solve this problem, relevance feedback techniques have been proposed to improve the retrieval process by iteratively refining the original user query in order to get new relevant documents in subsequent searches. Relevance feedback methods have demonstrated their power specially in vector space models, whose queries are composed of arrays of weighted terms. However, they are less developed in Boolean and FIRSs, based on the use of Boolean and weighted Boolean (fuzzy) queries where the

O. Cordón (✉)
Department of Computer Science and A.I. E.T.S. de Ingeniería Informática,
University of Granada, 18071 – Granada, Spain
e-mail: ocordon@decsai.ugr.es

F. Moya
Department of Librarianship,
University of Granada, 18071 – Granada, Spain

C. Zarco
PULEVA Salud S.A.,
Camino de Purchil, 66. 18004 – Granada, Spain

query terms are joined by the logical operators AND and OR. This is due to the fact that there is a need to know how to connect the query terms together using the Boolean operators in order to perform the relevance feedback mechanism, which is a difficult process even for human users.

Focusing on the FIR model, where the relevance feedback process also involves the query weight redefinition, the only existing approach is that of Kraft et al.'s [22], which is based on genetic programming [21]. Nevertheless, although this process achieves good results, it is not as accurate as desired sometimes.

In this paper, a new relevance feedback technique for FIRSs based on a hybrid simulated annealing-genetic programming evolutionary algorithm will be introduced with the aim of improving the performance of Kraft et al.'s proposal in terms of retrieval accuracy. Two different variants of this new approach will be introduced, showing how the adaptation of the retrieval threshold allows the effectiveness of the system to be increased. Moreover, the fact of being based on a neighborhood search technique will make the proposed algorithm to have a quicker convergence than Kraft et al.'s algorithm, thus making the relevance feedback process sometimes consume less processing time.

As other relevance feedback approaches based on the use of evolutionary algorithms [8, 9, 12, 13, 22, 35], the proposed technique will be useful for users that have a persistent need for the same type of information and are able to sacrifice a quick on-line response in order to increase the retrieval effectiveness.

This paper is structured as follows. Section 2 is devoted to the preliminaries, including the basis of Boolean IRSs and FIRSs, a short review of relevance feedback techniques, a brief overview of evolutionary algorithms and a review of their use in IR. Then, Kraft et al.'s proposal is introduced in Sect. 3. Section 4 presents the composition of the new algorithm proposed while the experiments developed to test it are showed in Sect. 5. Finally, several concluding remarks are pointed out in Sect. 6.

2 Preliminaries

2.1 Boolean information retrieval systems

An IRS is basically constituted of three main components, as showed in Fig. 1:

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, , the so called *retrieval status value* (RSV), 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). 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 Boolean IRS defines an indexing function:

$$F : D \times T \rightarrow \{0, 1\}$$

where $F(d, t)$ takes value 1 if term t appears in document d and 0 otherwise.

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 [30, 36]. 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 (RSV = 1) or not to be (RSV = 0) 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 as:

- 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

FIRSs make use of the fuzzy set theory [39] to deal with the imprecision and vagueness that characterizes the IR process. As stated in [4], the use of fuzzy sets in IR is suitable due to two main reasons:

1. It is a formal tool designed to deal with imprecision and vagueness.
2. It facilitates the definition of a superstructure of the Boolean model, so that the existing Boolean IRSs can be modified without redesigning them completely.

Hence, trying to solve the previously introduced problems of the Boolean IR model, 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.

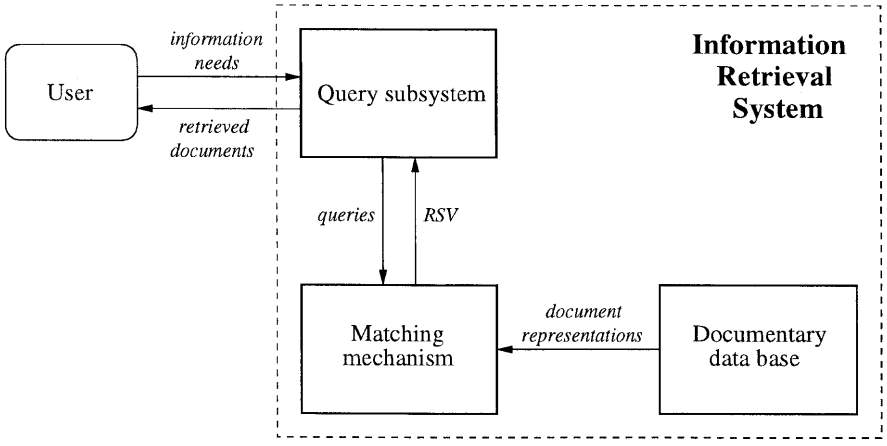

Fig. 1. Generic structure of an information retrieval system

2. Numeric weights (and in recenty proposals, linguistic terms [4,18]) are considered in the query with different semantics (a review of them all is to be found in [4]), 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, i.e., the RSV is defined as a real value in [0, 1]. 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 showed as follows.

2.2.1 Indexer Module

The indexer module of the FIRS defines an indexing function which maps the pair document-term into the real interval [0, 1]:

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

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:

$$d_i = \{\langle t, \mu_{d_i}(t)\rangle | t \in T\}; \quad \mu_{d_i}(t) = F(d_i, t)$$
$$t_j = \{\langle d, \mu_{t_j}(d)\rangle | d \in D\}; \quad \mu_{t_j}(d) = F(d, t_j)$$

There are different ways to define the indexing function F. In this paper, we will work with the normalized *inverse document frequency* [30]:

$$w_{d,t} = f_{d,t} \cdot \log(N/N_t); \quad F(d,t) = \frac{w_{d,t}}{\mathrm{Max}_d w_{d,t}}$$

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.

2.2.2 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 [4, 6] 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 [31]. 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*):

$$A_w = \{\langle d, \mu_{A_w}(d)\rangle | d \in D\}; \quad \mu_{A_w}(d) = \mathrm{Min}(w, \mu_A(d))$$
$$A^w = \{\langle d, \mu_{A^w}(d)\rangle | d \in D\}; \quad \mu_{A^w}(d) = \mathrm{Max}(1 - w, \mu_A(d))$$

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

$$\bar{A} = \{\langle d, \mu_{\bar{A}}(d)\rangle | d \in D\}; \quad \mu_{\bar{A}}(d) = 1 - \mu_A(d)$$

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:

$$A \ AND \ B = \{\langle d, \mu_{A \ AND \ B}(d)\rangle | d \in D\};$$
$$\mu_{A \ AND \ B}(d) = \mathrm{Min}(\mu_A(d), \mu_B(d))$$
$$A \ OR \ B = \{\langle d, \mu_{A \ OR \ B}(d)\rangle | d \in D\};$$
$$\mu_{A \ OR \ B}(d) = \mathrm{Max}(\mu_A(d), \mu_B(d))$$

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.

2.2.3 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:

$$q = \{\langle d, \mu_q(d)\rangle | d \in D\}; \quad \mu_q(d) = RSV_q(d)$$

Thus, one of the advantages of the FIRSs is that documents can be ranked in order to the membership degrees of relevance – as in IRSs based on the vector space model [30] – 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).

Focusing on the latter approach, which will be the one considered in this paper, the final set of documents retrieved would be:

$$R = \{d \in D | RSV_q(d) \geq \sigma\}$$

2.3 Relevance feedback

The basis of relevance feedback lie in the fact that either users normally formulate queries composed of terms that do not match the terms used to index the documents that are relevant to their needs or they do not provide the appropriate weights for the query terms. The operation mode involving modifying the previous query – adding and removing terms or changing the weights of the currently existing query terms – taking into account the relevance judgements of the documents retrieved by it constitutes a good way to solve the latter two problems and to improve the precision and especially the recall of the previous query [36]. This operation mode is represented in Fig. 2.

The difficulty found by non-expert users to express their retrieval needs in the form of a query makes necessary the design of automatic methods to perform the said task. The most of the research in automatic query modification has been developed in the field of vector space IRSs, where significantly good results have been obtained with very simple methods such as the Ide dec-hi [20].

However, it is not so easy to apply relevance feedback in Boolean and extended Boolean (fuzzy) IRSs as the automatic method must know how to connect the query terms together using the Boolean operators [35]. The existing approaches for Boolean systems link together terms based on their frequencies in documents and collections [10, 11]. On the other hand, in [35], it was proposed the use of genetic programming to perform this task, as we will see in Sect. 2.4.2. As regards FIRSs, up to our knowledge, the only approach developed to change the whole query is that of Kraft et al.'s [22], also based on genetic programming, that will be introduced in Sect. 3.

2.4 Evolutionary algorithms and its application to information retrieval

2.4.1 Introduction to evolutionary algorithms

Evolutionary computation uses computational models of evolutionary processes as key elements in the design and implementation of computer-based problem solving systems. There are a variety of evolutionary computational models that have been proposed and studied which are referred as *evolutionary algorithms* (EAs) [2]. There have been four well-defined EAs which have served as the basis for much of the activity in the field: *genetic algorithms* (GAs) [25], *evolution strategies* [34], *genetic programming* (GP) [21] and *evolutionary programming* [14].

An EA maintains a population of trial solutions, imposes random changes to these solutions, and incorporates selection to determine which ones are going to be maintained in future generations and which will be removed from the pool of the trials. But there are also important differences between them. GAs emphasize models of genetic operators as observed in nature, such as crossover (recombination) and mutation, and apply these to abstracted chromosomes with different representation schemes according to the problem being solved. Evolution strategies and evolutionary programming only apply to real-valued problems and emphasize mutational transformations that maintain the behavioral linkage between each parent and its offspring.

As regards GP, it constitutes a variant of GAs, based on evolving structures encoding programs such as expression trees. Apart from adapting the crossover and mutation operators to deal with the specific coding scheme considered, the remaining algorithm components remain the same.

GP has obtained succesful results in different applications such as symbolic regression but it suffers from a key limitation: 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, good trees solving the problem can be discarded by the selection procedure as the parameters involved in them are not well adjusted.

Several solutions have been proposed for this problem. On the one hand, one can use a local search algorithm to learn the coefficients associated to each tree in the population. On the other hand, the GA-P [19] paradigm, an hybrid algorithm combining traditional GAs with the GP technique can be considered to concurrently evolve the tree and the coefficients used in them, both of them encoded in the individual being adapted.

2.4.2 Previous applications of evolutionary algorithms to information retrieval

There have been an increasing interest in the application of artificial intelligence tools to IR in the last few years. Concretely, the machine learning paradigm, whose aim is

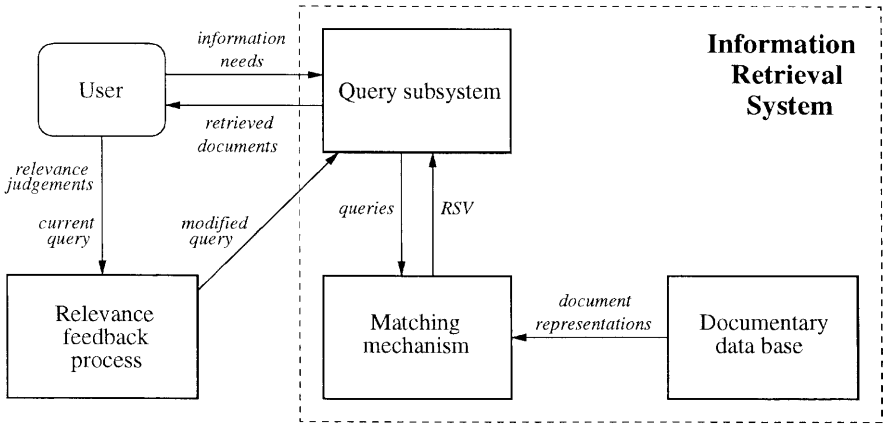

Fig. 2. The relevance feedback process

the design of systems able to automatically acquire knowledge by themselves, seems to be interesting in this topic [5].

EAs are not specifically learning algorithms but they offer a powerful and domain independent search ability that can be used in many learning tasks, as learning and self organization can be considered as optimization problems in many cases. Due to this reason, the application of EAs to IR has increased in the last decade. Among others, EAs have been applied to solve the following problems [7]:

1. *Automatic document indexing*, either by learning the relevant terms to describe them [15] or their weights [37]. Both approaches are based on a GA adapting the document descriptions in order to make them match more easily with querys to which they must be relevant.

On the other hand, in [12, 13], a GP algorithm to design a customized term weigthing function for web pages in the Internet is proposed obtaining very good results. The proposal constitutes a different approach to relevance feedback where the term weighting function is adapted to match the user's needs instead of the user query.

2. *Clustering of documents* [16, 27] *and terms* [28]. In the three cases, a GA is considered to obtain the cluster configuration. The first two approaches deal with the problem of obtaining user-oriented clusters of documents, i.e., groups of documents that should be relevant to the same user's information needs. The latter looks for groups of terms which appear with similar frequencies in the documents of a collection.

3. *Query definition*, by means of an on-line relevance feedback or and off-line inductive query by example [5] process. This is the most extended group of applications of EAs to IR. Among the different proposals, we can find:

- a GA to learn the query terms that best describe a set of relevant documents provided by an user [5],
- several GAs to perform relevance feedback by adapting the query term weights in vector space [23, 29, 38] and fuzzy [32] IRSs. Notice that the latter approach deals with extented Boolean queries as the proposal in this paper but it maintains fixed the structure of the query and only adapts its weights. However, the GA considered also adapts the retrieval threshold, which is an interesting idea that will also be used in this paper,
- GP-based relevance feedback processes for Boolean [35] and fuzzy [22] IRSs, and
- inductive query by example algorithms following the GA-P paradigm to learn the fuzzy query best describing a set of documents provided by an user [8, 9].

4. *Design of user profiles for IR in the Internet*. In [24], it was proposed an agent to model the user's information needs for searches in the web by an adaptive process based on a GA with fuzzy genes.

For a review of several of the previous approaches, see [7].

3 The Kraft et al.'s genetic programming-based relevance feedback algorithm for fuzzy information retrieval systems

In [22], Kraft et al. proposed a relevance feedback process to deal with extended Boolean queries in FIRSs. The algorithm is based on GP and its 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 [3], together with the elitist selection.

Genetic operators: The usual GP crossover is considered [21], 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 showed 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 ($p = 0.4$).

[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.

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* [25] 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 w be the query weight selected for mutation (the domain of w is $[0, 1]$), the new value for it is:

$$w' = \begin{cases} w + \triangle(t, 1 - w), & \text{if } a = 0 \\ w - \triangle(t, w), & \text{if } a = 1 \end{cases}$$

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.

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 [36]):

$$F_1 = \frac{\sum_d r_d \cdot f_d}{\sum_d r_d}; \quad F_2 = \alpha \cdot \frac{\sum_d r_d \cdot f_d}{\sum_d f_d} + \beta \cdot \frac{\sum_d r_d \cdot f_d}{\sum_d r_d} \quad (1)$$

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.

Moreover, as simple queries are always prefered by the user, a selection criterion has been incorporated to the algorithm in order to consider more fitted those queries with a lesser complexity among a group of chromosomes with the same fitness value.

4 A relevance feedback technique for fuzzy information retrieval systems based on simulated annealing-programming

In this section, the main proposal of this paper is introduced. First, the EA considered is reviewed. Then, its adaptation to solve the relevance feedback problem in FIRSs is described in detail. Finally, an extension of the previous approach based on also adapting the retrieval threshold σ during the feedback process is presented.

4.1 The SA-P algorithm

EAs, and more concretely GP, were introduced in Sect. 2.4.1 as powerful global search and optimization techniques. On the other hand, simulated annealing (SA) [1] is a neighborhood search algorithm which modifies the usual acceptance criteria of the basic local search sometimes permitting accepting a worse solution than the current one to avoid getting trapped in local optima.[2] SA starts from an initial solution and then generates a new candidate solution (close to it) by applying random changes on it. If the candidate solution is better than the current one, then the former replaces the latter. Otherwise, it still could be randomly accepted with a probability that depends on the difference between both solutions and on a parameter called temperature. This temperature is initiated to a high value (meaning that significantly worse candidate solutions are likely to be accepted) and then this value is decreased by a procedure called cooling strategy each time a number of neighbors are generated.

In [33], Sánchez et al. proposed a hybrid EA between SA and GP. The algorithm was based on encoding both a expresional part (the parse tree) and a value string (the coefficients involved in the expression) – as done in the GA-P (see Sect. 2.4.1) – and adapt it within an usual SA search scheme by a neighborhood operator based on the classical GP crossover and a string value mutation operator. As we will see in the next section, the SA-P algorithm combines both the high performance of GA-P with the quickness usually associated to neighborhood search algorithms.

4.2 Application of the SA-P algorithm to relevance feedback in FIRSs

To adapt the previous algorithm to the problem of relevance feedback in FIRSs, the extended Boolean query is encoded by storing the query structure – terms and logical operators – in the expresional part, and the term weights in the value string using a real coding scheme, as showed in Fig. 3.

The neighboorhood operator – called *macromutation* in [33] – generates the candidate fuzzy query by either changing the expresional part – the query structure – or the value string – the query weights – of the current individual I. This decision is randomly made with respect to a value string mutation probability p ($p = 1$ means "only weight mutation" while $p = 0$ means "only query structure mutation"). The query structure is mutated by selecting an edge of its parse tree and substituting the subtree located at it by a randomly generated parse tree (which is the same that crossing the original tree with a randomly generated one and taking one of the two offspring). The weight vector is mutated by applying intermediate recombination [26] between the current values ($weights(I)$) and a randomly generated vector W with an amplitude parameter that depends on the current temperature T by a constant K_1 as follows:

$$weights(I) = weights(I) \cdot (T/K_1) + (1 - (T/K_1)) \cdot W$$

The SA-P algorithm considered in our case, which is an adaptation of the one proposed in [33], is showed as follows:

[2] For an application of SA to IR, see [5].

```
algorithm SAP−RF
needs : MaxEval /∗maximum number of evaluations∗/, MaxNeighs /∗maximum number of
        neighbors generated per temperature∗/, MaxSuccess /∗maximum number of
        neighbors accepted per temperature∗/, c /∗cooling factor∗/, T_0
        ∗initial temperature∗/, p /∗value string mutation probability∗/,
        K_1 /∗value string mutation parameter∗/
produces: Ibest

I = Ibest = random individual
T = T_0
Eval = 1
while (Eval <= MaxEval) do
    num_neighs = num_success = 0
    while (num_neighs < MaxNeighs) (num_success < MaxSuccess)
       Icand = macromutation(I, p, T, K_1)
       num_neighs = num_neighs + 1
       delta = F(I) − F(Icand)
       Eval = Eval + 1
       v = random value with uniform distribution U(0, 1)
       if (delta < 0) or (v < exp(−delta/T)) then
          I = Icand
          num_success = num_success + 1
          if (I > Ibest) then Ibest = I end if
       end if
       end while
       T = c ∗ T
end while
```


As seen in the algorithm, the initial solution is a random fuzzy query. The initial temperature T_0 is computed by means of the following expression:

$$T_0 = \frac{\mu}{-\ln(\phi)} \cdot F(I)$$

with I being the initial solution and ϕ being the probability of acceptance for a solution that can be μ per 1 worse than $F(I)$. Both parameters are defined in the interval $[0, 1]$.

Finally, notice that we consider the function F_2 showed in (1) as fitness function to measure the quality of a fuzzy query.

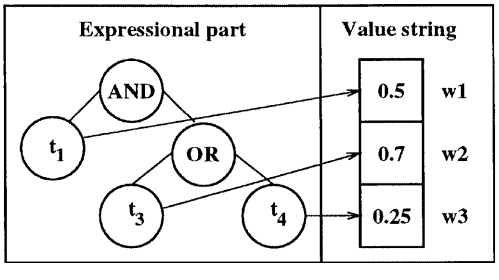

Fig. 3. Individual representing the fuzzy query 0.5 t_1 *AND* (0.7 t_3 *OR* 0.25 t_4)

4.3 An extension of the relevance feedback technique proposed to adapt the retrieval threshold

In [9, 32], several experiments were made in FIRS relevance feedback and inductive query by example environments showing that better effectiveness could be obtained if not only the fuzzy query but also the retrieval threshold σ (see Sect. 2.2.3), which is usually a difficult choice for the user, is adapted by the query modification process.

To do so, this value is encoded in the individual (see Fig. 4) and it is initialized to the value provided by the user. Then, it is adapted during the algorithm run each time the macromutation operator acts on the weight vector by the following operation:

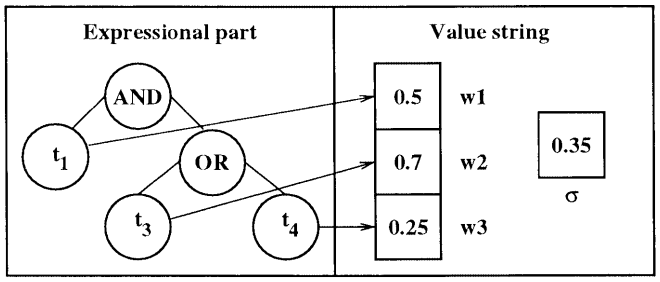

Fig. 4. Individual representing the fuzzy query 0.5 t_1 *AND* (0.7 t_3 *OR* 0.25 t_4) and the retrieval threshold $\sigma = 0.35$

$$threshold(I) = threshold(I) \cdot (T/K_1) + (1 - (T/K_1)) \cdot randthr$$

with *randthr* being a real value randomly generated in (0,1].

5 Experiments developed and analysis of results

This section is devoted to test the performance of both proposals of relevance feedback algorithms for FIRSs introduced in this paper. To do so, we have worked with two different document collections, a set of 359 documents extracted from the the *Library and Information Science Abstracts* (*LISA*) data base, and the well known *Cranfield* collection composed of 1400 documents about Aeronautics.

The next two subsections respectively report the results obtained with each of them. They both are divided into three parts: the first one describing the experimental test bed, and the other two analyzing the results obtained by both variants of our algorithm.

5.1 Experiments with a collection extracted from the LISA documentary base

5.1.1 Experimental test bed

In the first experimental study developed to test the performance of the proposed algorithm, we have followed a similar methodology as that in [22]. A documentary base, in this case composed of 359 abstracts taken from the LISA base, has been automatically indexed by first extracting the non-stop words, thus obtaining a total number of 2609 different indexing terms, and then using the normalized IDF scheme to generate the term weights in the document representations. A user has selected a set of 82 relevant documents that have been provided to Kraft et al.'s and both variants of our system (noted by SAP-RF), which have been run considering the second fitness function F_2. In order to make a fair comparison, the three algorithms have been run three times with different initializations during the same fixed number of fitness function evaluations (100,000). The parameters considered for them are showed in Table 1.

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 [22]). On the other hand, different values for the value string mutation probability p have been tested in order to study the robustness of our proposal.

5.1.2 Results obtained without adapting the retrieval threshold

The results obtained by Kraft et al.'s and our method (without considering the retrieval threshold adaptation) are respectively showed in Tables 2 and 3, where *Run* stands for the corresponding algorithm run (1 to 3), T for the run time (both algorithms have been run in a 350 MHz. Pentium II computer with 64 MB of memory, and the time is measured in minutes), *Sz* for

Table 1. Parameter values considered

Parameter	Decision
Common parameters	
Number of evaluations	100,000
Expression part limited to	10 nodes
Weighting coefficientes α, β in F_2	1.2, 0.8
(Initial) retrieval threshold σ	0.5
Kraft et al.'s algorithm	
Population size	1600
Crossover and Mutation probabilities	0.8, 0.2
SAP-RF	
Initial temperature computation parameters	$\mu = 0.5, \phi = 0.5$
Max. # neighbors generated per temperature	500
Max. # neighbors accepted per temperature	50
Cooling parameter c	0.9
Value string mutation probability p	$\{0.25, 0.5, 0.75\}$
Value string mutation parameter K_1	5

Table 2. Results obtained by Kraft et al.'s method in our LISA collection

Run	*T*	*Sz*	*Fit*	*P*	*R*	*#rr/#rt*
1	3:19	9	1.409214	0.962963	0.317073	26/27
2	3:24	7	1.395122	1.000000	0.243902	20/20
3	3:25	9	1.414634	1.000000	0.268293	22/22

Table 3. Results obtained by SAP-RF in LISA without adapting the retrieval threshold

Run	*p*	*T*	*Sz*	*Fit*	*P*	*R*	*#rr/#rt*
1	0.25	1:26	9	1.508130	0.972222	0.426829	35/36
2	0.25	1:22	7	1.424390	1.000000	0.280488	23/23
3	0.25	1:23	9	1.508130	0.972222	0.426829	35/36
1	0.5	1:25	9	1.453659	1.000000	0.317073	26/26
2	0.5	1:23	9	1.464939	0.968750	0.378049	31/32
3	0.5	1:28	9	1.497422	0.971429	0.414634	34/35
1	0.75	1:30	9	1.424390	1.000000	0.280488	23/23
2	0.75	1:29	9	1.497422	0.971429	0.414634	34/35
3	0.75	1:27	7	1.395122	1.000000	0.243902	20/20

Table 4. Results obtained by SAP-RF in LISA adapting the retrieval threshold

Run	*p*	*T*	*Sz*	*Thr*	*Fit*	*P*	*R*	*#rr/#rt*
1	0.25	1:26	9	0.323620	1.577094	0.956522	0.536585	44/46
2	0.25	1:26	9	0.287751	1.577094	0.956522	0.536585	44/46
3	0.25	1:25	9	0.259209	1.577094	0.956522	0.536585	44/46
1	0.5	1:26	9	0.126364	1.420558	0.964286	0.329268	27/28
2	0.5	1:28	9	0.291257	1.577094	0.956522	0.536585	44/46
3	0.5	1:28	9	0.149957	1.482927	1.000000	0.353659	29/29
1	0.75	0:49	5	0.605357	1.297561	1.000000	0.121951	10/10
2	0.75	1:25	9	0.155147	1.409214	0.962963	0.317073	26/27
3	0.75	1:27	9	0.209458	1.356098	1.000000	0.195122	16/16

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 query, and *#rr* for the number of relevant documents retrieved.

In view of these results, the performance of our proposal is significant as it overcomes Kraft et al.'s algorithm in the most of the cases. This improvement in retrieval performance is probably due to the better ability of our algorithm to learn the query weights, as Kraft et al.'s suffers from the difficulty of GP to derive the values of the constants in the evolved expresions (see Sect. 2.4.1). Moreover, and which is more important, our SAP-RF requires less than a half of the run time of Kraft's proposal, which is an important point in this field.

5.1.3
Results obtained adapting the retrieval threshold

The results obtained by our SAP-FIR method when adapting the retrieval threshold within the relevance feedback process are collected in Table 4, which considers the same equivalences than the previous two tables. Only a new column has been included, that labeled by *Thr*, containing the value of the learned threshold.

In this case, a deeper analysis is needed. On the one hand, it can be seen that the adaptation of the retrieval threshold allows the SAP-RF algorithm to obtain the best retrieval effectiveness when considering $p = 0.25$ (i.e., the weight vector and the retrieval threshold are adapted only in the twentyfive percent of the cases). The proposal shows to be very robust as the best result is obtained in the three runs developed. Moreover, it is important to notice that this performance improvement is obtained without increasing the computational time required as the time needed to adapt the threshold is not significant.

On the other hand, notice that when increasing the value of *p*, i.e., when augmenting the number of times where the weights and the threshold are adapted, the algorithm performance reduces significantly. This is due to the fact that a high number of adaptations of the retrieval threshold makes the SA-P algorithm converge excessively quickly, thus increasing the chances to get stuck in local optima. This assumption can be clearly corroborated in view of the result obtained in the first run with $p = 0.75$, which is the only one where the algorithm stopped because none of the 500 neighbors generated in an iteration were accepted.

Finally, as regards the specific values generated for the thresholds, we can see that they are very different from one run to the others (ranging from 0.126 to 0.605). This is because this value directly depends on the query generated, which is different in each case.

5.2
Experiments with the Cranfield collection

5.2.1
Experimental test bed

The second experimental study has been developed using the Cranfield collection. The 1400 documents composing this documentary base has been automatically indexed in the same way that our LISA collection considered in the previous section (obtaining 3857 different indexing terms). Among the 225 queries associated to the Cranfield collection, we have selected those presenting 20 or more relevant documents in order to have enough chances to show the performance advantage of one algorithm over the other. The resulting seven queries and the number of relevant documents associated to them are showed in Table 5.

Both of our proposals and Kraft et al.'s algorithm have been run three times for each query as done in the previous section. The same parameter values have been considered but the following two changes:

- The query size has been increased to 20 nodes in order to scale to a more complex problem presenting a higher number of documents and indexing terms than the previous one.

Table 5. Cranfield queries with 20 or more relevant documents

#query	#relevant documents
1	29
2	25
23	33
73	21
157	40
220	20
225	25

Table 6. Results obtained by Kraft et al.'s method in the Cranfield collection

#q	Run	T	Sz	Fit	P	R	#rr/#rt
	1	12:48	19	1.282759	1.000000	0.103448	3/3
1	2	13:01	19	1.282759	1.000000	0.103448	3/3
	3	12:49	17	1.255172	1.000000	0.068966	2/2
	1	12:58	17	1.328000	1.000000	0.160000	4/4
2	2	12:44	17	1.328000	1.000000	0.160000	4/4
	3	12:51	13	1.296000	1.000000	0.120000	3/3
	1	12:52	19	1.272727	1.000000	0.090909	3/3
23	2	12:49	17	1.272727	1.000000	0.090909	3/3
	3	12:48	17	1.272727	1.000000	0.090909	3/3
	1	12:36	19	1.466667	1.000000	0.333333	7/7
73	2	12:59	19	1.466667	1.000000	0.333333	7/7
	3	12:49	17	1.504762	1.000000	0.380952	8/8
	1	12:47	19	1.260000	1.000000	0.075000	3/3
157	2	12:45	19	1.260000	1.000000	0.075000	3/3
	3	12:49	17	1.240000	1.000000	0.050000	2/2
	1	12:40	17	1.400000	1.000000	0.250000	5/5
220	2	12:47	17	1.400000	1.000000	0.250000	5/5
	3	12:43	7	1.360000	1.000000	0.200000	4/4
	1	12:41	19	1.328000	1.000000	0.160000	4/4
225	2	12:51	17	1.328000	1.000000	0.160000	4/4
	3	12:45	17	1.328000	1.000000	0.160000	4/4

Table 7. Results obtained by SAP-RF in Cranfield without adapting the retrieval threshold

#q	Run	T	Sz	Fit	P	R	#rr/#rt
	1	13:38	17	1.365517	1.000000	0.206897	6/6
1	2	13:04	19	1.393103	1.000000	0.241379	7/7
	3	13:23	19	1.393103	1.000000	0.241379	7/7
	1	13:09	17	1.424000	1.000000	0.280000	7/7
2	2	12:39	17	1.488000	1.000000	0.360000	9/9
	3	12:53	19	1.488000	1.000000	0.360000	9/9
	1	13:29	19	1.369697	1.000000	0.212121	7/7
23	2	13:02	19	1.369697	1.000000	0.212121	7/7
	3	13:02	19	1.321212	1.000000	0.151515	5/5
	1	12:38	19	1.619048	1.000000	0.523810	11/11
73	2	13:06	19	1.619048	1.000000	0.523810	11/11
	3	13:05	19	1.619048	1.000000	0.523810	11/11
	1	13:43	19	1.380000	1.000000	0.225000	9/9
157	2	13:43	19	1.320000	1.000000	0.150000	6/6
	3	13:28	19	1.380000	1.000000	0.225000	9/9
	1	13:05	17	1.520000	1.000000	0.400000	8/8
220	2	13:22	19	1.560000	1.000000	0.450000	9/9
	3	13:40	19	1.640000	1.000000	0.550000	11/11
	1	13:18	19	1.520000	1.000000	0.400000	10/10
225	2	13:22	19	1.488000	1.000000	0.360000	9/9
	3	13:21	17	1.488000	1.000000	0.360000	9/9

- The retrieval threshold has been decreased to 0.1 in order to retrieve a significant number of documents as the query weights are lower in the Cranfield documentary base than in our LISA collection.

Finally, as regards the value string mutation probability p, we have made use of the knowledge obtained in the previous experimentation and has considered a single value for each case: $p = 0.5$ for the runs without learning the retrieval threshold and $p = 0.25$ for those where it is learned.

5.2.2 Results obtained without adapting the retrieval threshold

The results obtained by Kraft et al.'s and our method (without considering the retrieval threshold adaption) are respectively showed in Tables 6 and 7, both of them having the same nomenclature than in Sect. 5.1.2. The first column #q refers to the number of each of the seven Cranfield queries selected. The algorithms have been run again in the same 350 MHz. Pentium II computer with 64 MB of memory.

The conclusion drawn from the results is even more clear this time as our SAP-RF algorithm significantly

Table 8. Results obtained by SAP-RF in Cranfield adapting the retrieval threshold

#q	Run	T	Sz	Thr	Fit	P	R	#rr/#rt
	1	12:45	19	0.297264	1.475862	1.000000	0.344828	10/10
1	2	12:43	19	0.294263	1.448276	1.000000	0.310345	9/9
	3	13:20	19	0.044080	1.420690	1.000000	0.275862	8/8
	1	13:09	19	0.046401	1.520000	1.000000	0.400000	10/10
2	2	12:36	17	0.166046	1.488000	1.000000	0.360000	9/9
	3	13:20	19	0.163329	1.520000	1.000000	0.400000	10/10
	1	13:20	19	0.094884	1.515152	1.000000	0.393939	13/13
23	2	13:09	19	0.038554	1.369697	1.000000	0.212121	7/7
	3	12:09	15	0.169519	1.345455	1.000000	0.181818	6/6
	1	12:58	19	0.093121	1.657143	1.000000	0.571429	12/12
73	2	13:18	19	0.098302	1.619048	1.000000	0.523810	11/11
	3	13:13	19	0.102439	1.542857	1.000000	0.428571	9/9
	1	13:12	19	0.172545	1.360000	1.000000	0.200000	8/8
157	2	12:45	19	0.115925	1.340000	1.000000	0.175000	7/7
	3	13:12	19	0.004288	1.320000	1.000000	0.150000	6/6
	1	13:45	19	0.049878	1.520000	1.000000	0.400000	8/8
220	2	13:19	19	0.076543	1.520000	1.000000	0.400000	8/8
	3	12:41	19	0.128209	1.680000	1.000000	0.600000	12/12
	1	12:52	17	0.036680	1.456000	1.000000	0.320000	8/8
225	2	13:10	19	0.112805	1.488000	1.000000	0.360000	9/9
	3	13:05	19	0.099069	1.488000	1.000000	0.360000	9/9

outperforms Kraft et al.'s proposal in all the cases. The minimum individual retrieval performance improvement is obtained in the third run of query 23 with a 67% in the recall value (0.151515 against 0.090909), whilst the maximum one is got in the third run of query 157 with a 350% (0.225 against 0.05). As regards the global results for each query, the lower improvement corresponds to query 73 with only a 37.5% of improvement (a recall value of 0.523810 obtained in the three runs of SAP-RF against to 0.380952 resulting from the third run of Kraft et al.'s algorithm), and the highest one to query 157 with a 300% (a recall value of 0.225 generated in the first and third runs of SAP-RF against another of 0.075 got from the first and second runs of Kraft et al.'s method).

However, in this case we can not find a reduction in the computational time needed to generate the queries by means of the SAP-RF technique. This time, our method requires approximately the same time than Kraft et al.'s proposal and usually a little bit more. We think that this is a consequence of the inclusion of a selection criterion to get simpler queries in the latter method (see Sect. 3). This criterion makes the population being composed of simpler queries while the EA is converging, thus making their evaluation – the more time consuming procedure in both algorithms – less demanding. As the SAP-RF process lacks of this characteristic due to its nature of neighborhood search technique allowing solutions worsening the current one to be accepted, this time it has had to evaluate more complex queries during its learning process – thus consuming more computational time – and finally has generated more complex (although significantly more accurate) queries. Notice that this fact has not happened in the previous experimentation with the LISA collection as the maximum query size considered (10) was so small to affect the time required to evaluate the encoded queries.

5.2.3
Results obtained adapting the retrieval threshold

The results obtained by our SAP-FIR method when adapting the retrieval threshold in the Cranfield collection are collected in Table 8. Column names stand for the same equivalences than in Table 4 (Sect. 5.1.3).

The extension proposed to adapt the retrieval threshold shows again very satisfactory results. In five of the seven queries – numbers 1, 2, 23, 73 and 220 – the best absolute retrieval result is improved when automatically learning the threshold, with the following respective percentages of improvement in the recall measure: 42.9 (0.344828 against 0.241379), 11.1 (0.4 against 0.36), 85.7 (0.393939 against 0.212121), 9.1 (0.571429 against 0.52381), and 9.1 (0.6 against 0.55). In the remaining two queries – numbers 157 and 225 – there is a recall loss of a 11.1 (0.2 against 0.225) and a 10% (0.36 against 0.4), respectively. However, the obtained values are still much more better than those generated from Kraft et al.'s method.

Moreover, in 10 of the 21 individual runs performed, the SAP-RF algorithm adapting the threshold outperforms the basic version that does not adapt this parameter, whilst in other 6 cases the same result is provided. Only in the remaining five cases (the third run of query 73, the first and third runs of query 157, the second run of query 220 and the first run of query 225), the joint adaption of query and retrieval threshold leads to a local optimum, thus resulting in a worser retrieval performance than in the previous case.

In this experimentation, the threshold values generated present a lower deviation than in that performed with our LISA collection, ranging from 0.004 to 0.297. This corroborates the fact that the weights of the indexing terms are lower in this collection and hence smaller values for the retrieval threshold are needed in this case.

Finally, notice that the performance improvements obtained does not again need of a increase of the computa-

tional time, showing similar values in this aspect to the basic algorithm.

6 Concluding remarks

A relevance feedback technique for FIRSs based on a hybrid SA-GP algorithm has been proposed. It has performed appropriately in two different documentary bases – a collection of 359 documents extracted from the LISA database and the well known 1400 document Cranfield collection –, outperforming the previous proposal by Kraft et al. both in terms of retrieval performance in both cases, and of the required computation time in the former.

Moreover, the proposed method has been extended by allowing it to adapt the retrieval threshold, which is usually a fixed value provided by the user. This new variant has increased even more the system effectiveness without augmenting the algorithm run time.

References

1. **Aarts EHL** (1989) Simulated annealing and boltzman machines: a stochastic approach to combinatorial optimization and neural computing, Wiley
2. **Bäck T** (1996) Evolutionary Algorithms in Theory and Practice, Oxford University Press, Oxford
3. **Baker JE** (1987) Reducing bias and inefficiency in the selection algorithm, Proc. Second International Conference on Genetic Algorithms (ICGA'87), Hillsdale, pp. 14–21
4. **Bordogna G, Carrara P, Pasi G** (1995) Fuzzy approaches to extend Boolean information retrieval. In: Bosc P, Kacprzyk J (eds), Fuzziness in Database Management Systems, pp. 231–274
5. **Chen H** (1998) A machine learning approach to inductive query by examples: an experiment using relevance feedback, ID3, genetic algorithms, and simulated annealing, J Amer Soc Information Sci **49**(8): 693–705
6. **Cross V** (1994) Fuzzy information retrieval, J Intelligent Information Syst **3**: 29–56
7. **Cordón O, Moya F, Zarco C** (1999) 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, pp. 179–186
8. **Cordón O, Moya F, Zarco C** (1999) Learning queries for a fuzzy information retrieval system by means of GA-P techniques, Proc EUSFLAT-ESTYLF Joint Conference, Palma de Mallorca, Spain, pp. 335–338
9. **Cordón O, Moya F, Zarco C** (2000) A GA-P algorithm to automatically formulate extended Boolean queries for a fuzzy information retrieval system, Mathware & Soft Computing **7**(2–3) 309–322
10. **Dillon M, Desper J** (1980) Automatic relevance feedback in Boolean retrieval systems, J Documentation **36**: 197–208
11. **Dillon M, Ulmschneider J, Desper J** (1983) A prevalence formula for automatic relevance feedback in Boolean systems, Information Processing and Management **19(1): 27–36**
12. **Fan W, Gordon M, Pathak P** (2000) Personalization of search engine services for effective retrieval and knowledge management, Proc 2000 International Conference on Information Systems (ICIS)
13. **Fan W, Gordon M, Pathak P, Radev D** (2001) Automatic term weighting for effective retrieval: a genetic programming approach, submitted to SIGIR
14. **Fogel DB** (1991) System identification trough simulated evolution. A Machine Learning Approach, Ginn Press, USA
15. **Gordon M** (1988) Probabilistic and genetic algorithms for document retrieval, Commun ACM **31**(10): 1208–1218
16. **Gordon M** (1991) User-based document clustering by redescribing subject descriptions with a genetic algorithm, J Amer Soc Information Sci **42**(5): 311–322
17. **Gordon M, Pathak P** (1999) Finding information on the world wide web: the retrieval effectiveness of search engines, Information Processing and Management **35**(2): 141–180
18. **Herrera-Viedma E** (2001) Modelling the retrieval process of an information retrieval system using an ordinal fuzzy linguistic approach, J Amer Soc Information Sci **52**(6): 460–475
19. **Howard L, D'Angelo D** (1995) The GA-P: a genetic algorithm and genetic programming hybrid, IEEE Expert 11–15
20. **Ide E** (1971) New experiments in relevance feedback. In: Salton G (ed.), The SMART Retrieval System, Prentice Hall, pp. 337–354
21. **Koza J** (1992) Genetic programming. On the Programming of Computers by Means of Natural Selection, The MIT Press
22. **Kraft DH, Petry FE, Buckles BP, Sadasivan T** (1997) Genetic algorithms for query optimization in information retrieval: relevance feedback. In: Sanchez E, Shibata T, Zadeh LA (eds), Genetic Algorithms and Fuzzy Logic Systems, World Scientific, 155–173.
23. **López-Pujalte C, Guerrero VP, Moya F** (2002) A test of genetic algorithms in relevance feedback, Information Processing & Management. (to appear).
24. **Martín-Bautista MJ, Vila MA, Larsen HL** (1999) A fuzzy genetic algorithm approach to an adaptive information retrieval agent, J Amer Soc Information Sci **50**(9): 760–771
25. **Michalewicz Z** (1996) Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag
26. **Mühlenbein H, Schlierkamp-Voosen D** (1993) Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization, Evolut Comput **1**(1): 25–49
27. **Raghavan VV, Agarwal B** (1987) Optimal determination of user-oriented clusters: an application for the reproductive plan, Proc Second International Conference on Genetic Algorithms and Their Applications (ICGA'87), Hillsdale, NJ, USA pp. 241–246
28. **Robertson AM, Willet P** (1994) Generation of equifrequent groups of words using a genetic algorithm, J Documentation **50**(3): 213–232
29. **Robertson AM, Willet P** (1996) An upperbound to the performance of ranked-output searching: optimal weighting of query terms using a genetic algorithm, J Documentation **52**(4): 405–420
30. **Salton G, McGill MJ** (1989) Introduction to Modern Information Retrieval, McGraw-Hill, New York
31. **Sanchez E** (1989) Importance in knowledge systems, Information Syst **14**(6): 455–464
32. **Sanchez E, Miyano H, Brachet JP** (1995) Optimization of fuzzy queries with genetic algorithms. Application to a data base of patents in biomechanical engineering, Proc Sixth IFSA World Congress, Sao Paulo, Brazil, vol. II, pp. 293–296
33. **Sánchez L, Couso I, Corrales JA** (2001) Combining GP operators with SA search to evolve fuzzy rule based classifiers, Information Sci **136**(1–4): 175–191
34. **Schwefel H-P** (1995) Evolution and optimum seeking. Sixth-Generation Computer Technology Series, John Wiley and Sons
35. **Smith MP, Smith M** (1997) The use of genetic programming to build Boolean queries for text retrieval through relevance feedback, J Information Sci **23**(6): 423–431
36. **van Rijsbergen CJ** (1979) Information Retrieval (2nd edition), Butterworth, London
37. **Vrajitoru D** (1998) Crossover improvement for the genetic algorithm in information retrieval, Information Processing and Management **34**(4): 405–415
38. **Yen JJ, Korfhage RR** (1994) Query modification using genetic algorithms in vector space models, Int J Expert Syst **7**(2): 165–191
39. **Zadeh LA** (1965) Fuzzy sets, Information and Control **8**: 338–353