Automatic Learning of Multiple Extended Boolean Queries by Multiobjective GA-P Algorithms*

O. Cordón[1], F. Moya[2], and C. Zarco[3]

[1] Dept. of Computer Science and Artificial Intelligence
University of Granada. 18071 - Granada (Spain)
ocordon@decsai.ugr.es
[2] Dept. of Information Sciences. Faculty of Information Sciences
University of Granada. 18071 - Granada (Spain)
felix@ugr.es
[3] PULEVA Salud S.A.
Camino de Purchil, 66. 18004 - Granada (Spain)
czarco@puleva.es

Summary. In this contribution, a new Inductive Query by Example process is proposed to automatically derive extended Boolean queries for fuzzy information retrieval systems from a set of relevant documents provided by a user. The novelty of our approach is that it is able to simultanously generate several queries with a different precision-recall tradeoff in a single run. To do so, it is based on an advanced evolutionary algorithm, GA-P, specially designed to tackle with multiobjective problems by means of a Pareto-based multiobjective technique. The performance of the new proposal will be tested on the usual Cranfield collection and compared to the well-known Kraft et al.'s process.

Key words: Fuzzy Information Retrieval, Inductive Query by Example, Evolutionary Algorithms, Genetic Algorithm-Programming, Multiobjective Evolutionary Algorithms.

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 a user [28, 2]. Information retrieval systems (IRSs) are a

* This work has been supported by CICYT under Project TIC2002-03276 and by the University of Granada under Project "Mejora de Metaheurísticas mediante Hibridación y sus Aplicaciones".

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 appropriate time interval.

The underlying retrieval model of most of the commercial IRSs is the Boolean one [34], 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 [28] or the fuzzy information retrieval (FIR) models [3, 12].

However, the increase in the power of the retrieval model also comes with a high complexity augment in the query language, what makes difficult for the user to represent his information needs in the form of a valid query. This is especially significant in the case of fuzzy IRSs, whose query language allows us to formulate weighted Boolean (fuzzy) queries where the query terms are joined by the logical operators AND and OR. If it is difficult for a human user to formulate a classical Boolean query due to the need to know how to properly connect the query terms together using the Boolean operators, it will be even more difficult to both define the query structure and specify the query term weights to retrieve the desired documents.

Hence, the paradigm of Inductive Query by Example (IQBE) [5], where queries describing the information contents of a set of documents provided by a user are automatically derived, can be useful to solve this problem and assist the user in the query formulation process. Focusing on the FIR model, the most known existing approach is that of Kraft et al. [24], which is based on genetic programming [23]. Moreover, several other approaches have been proposed based on the use of more advanced evolutionary algorithms [1], such as genetic algorithm-programming (GA-P) [21] or simulated annealing-programming (SA-P) [30], in order to improve the Kraft et al.'s one [8, 9, 10].

On the other hand, it is well known that the performance of an IRS is usually measured in terms of two different criteria, precision and recall [34]. This way, the optimization of any of its components, and concretely the automatic learning of fuzzy queries, is a clear example of a multiobjective problem. Usually, the application of evolutionary algorithms in the area has been based on combining both criteria in a single scalar fitness function by means of a weighting scheme [7]. However, there is a kind of evolutionary algorithms specially designed for multiobjective problems, *multiobjective evolutionary algorithms*, which are able to obtain different non-dominated solutions to the problem in a single run [14, 6].

In [11], it was proposed an extension of Smith and Smith's IQBE algorithm to learn Boolean queries [33] transforming it into a Pareto-based multiobjective evolutionary algorithm. The proposed process obtained very good results in one of the most known IR benchmarks, the Cranfield document collection.

In this chapter, the same idea will be applied to learn extended Boolean queries for FIRSs. This way, an IQBE process similar to that proposed in [8] – based on a GA-P algorithm – will be transformed in a Pareto-based

multiobjective evolutionary algorithm to deal with this problem. The proposal will be compared to the previous approach by Kraft et al. in the well-known Cranfield documentary base.

To do so, this paper is structured as follows. In Section 2, some preliminaries are introduced by reviewing the basis of IRSs and FIRSs, IQBE and multiobjective evolutionary algorithms. Then, two single-objective IQBE processes are discussed in Section 3, Kraft et al.'s genetic programming-based algorithm and our previous proposal based on the use of the more advanced GA-P technique. Section 4 is devoted to extend the latter process to deal with the multiobjective problem of simultaneously optimizing both precision and recall by means of the Pareto-based approach. The experiments developed on the Cranfield collection are presented and analyzed in Section 5. Finally, Section 6 summarizes several concluding remarks.

2 Preliminaries

2.1 Boolean Information Retrieval Systems

An IRS is basically constituted by three main components, as showed in Figure 1:

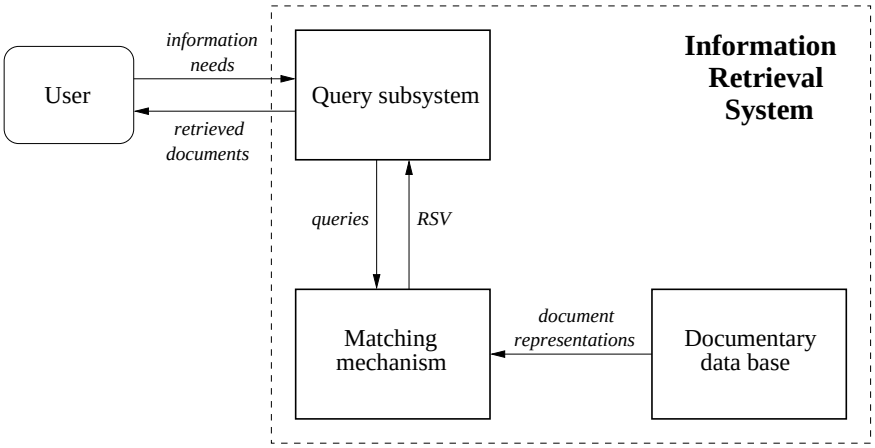

Fig. 1. Generic structure of an information retrieval system

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 considers combinations of simple user requirements with logical operators AND, OR and NOT [28, 34]. The result obtained from the processing of a query is a set of documents that totally match with the query, 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 correspond to 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 [35] to deal with the imprecision and vagueness that characterizes the IR process. As stated in [3], 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 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.
2. Numeric weights (and in recent proposals, linguistic terms [3, 20]) are considered in the query with different semantics (a review of them is to be found in [3]), 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.

Indexer Module

The indexer module of the FIRS defines an indexing function which maps the document-term pair 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:

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

There are different ways to define the indexing function F. In this paper, we will work with the normalized *inverse document frequency* [28]:

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

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 [3, 12] 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 [29]. 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*):

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

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

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

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

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

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

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

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 the former 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 FIRSs is that documents can be ranked in order to the membership degrees of relevance – as in IRSs based on the vector space model [28] – 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:

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

2.3 Inductive Query by Example

IQBE was proposed in [5] as "a process in which searchers provide sample documents (examples) and the algorithms induce (or learn) the key concepts in order to find other relevant documents". This way, IQBE is a process for assisting the users in the query formulation process performed by machine learning methods [26]. It works by taking a set of relevant (and optionally, non relevant documents) provided by a user – that can be obtained from a preliminary query or from a browsing process in the documentary base – and applying an off-line learning process to automatically generate a query describing the user's needs (as represented by the document set provided by him). The obtained query can then be run in other IRSs to obtain more relevant documents. This way, there is no need that the user interacts with the process as in other query refinement techniques such as relevance feedback [28, 2].

Several IQBE algorithms have been proposed for the different existing IR models. On the one hand, Smith and Smith [33] introduced a Boolean query learning process based on genetic programming. Besides, a similar idea to that proposed in this paper was applied in [11] in order to allow the Smith and Smith's algorithm to simultaneously derive multiple Boolean queries from the same document set. On the other hand, all of the machine learning methods considered in Chen et al.'s paper [5] (regression trees, genetic algorithms and simulated annealing) dealt with the vector space model. Moreover, there are several approaches for the derivation of weighted Boolean queries for FIRSs, such as the genetic programming algorithm of Kraft et al. [24], that will be reviewed in the next section, our niching GA-P method [9] and our SA-P method [10], based on a simulated annealing-genetic programming hybrid.

For descriptions of some of the previous techniques based on EAs refer to [8, 10].

2.4 Multiobjective 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) [1]. Concretely, four well-defined EAs have served as the basis for much of the activity in the field: *genetic algorithms* (GAs) [25], *evolution strategies* [32], *genetic programming* (GP) [23] and *evolutionary programming* [16].

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. Focusing on the two kinds of EAs considered on this paper, GAs and GP, the former 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. 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 rest of the algorithm components remain the same.

EAs are very appropriate to solve multiobjective problems. These kinds of problems are characterized by the fact that several objectives have to be simultaneously optimized. Hence, there is not usually a single best solution solving the problem, i.e. being better than the remainder with respect to every objective, as in single-objective optimization. Instead, in a typical multiobjective optimization problem, there is a set of solutions that are superior to the remainder when all the objectives are considered, the *Pareto* set. These solutions are known as *non-dominated solutions* [4], while the remainder are known as *dominated solutions*. Since none of the Pareto set solutions is absolutely better than the other non-dominated solutions, all of them are equally acceptable as regards the satisfaction of all the objectives.

This way, thanks to the use of a population of solutions, EAs can search many Pareto-optimal solutions in the same run. Generally, multiobjective EAs only differ from the rest of EAs in the fitness function and/or in the selection mechanism. The evolutionary approaches in multiobjective optimization can be classified in three groups: *plain aggregating approaches*, *population-based non-Pareto approaches*, and *Pareto-based approaches* [14, 6].

The first group constitutes the extension of classical methods to EAs. The objectives are artificially combined, or aggregated, into a scalar function according to some understanding of the problem, and then the EA is applied in the usual way[4]. Optimizing a combination of the objectives has the advantage

[4] As said, this has been the approach usually followed in the application of EAs to IR.

of producing a single compromise solution but there are two problems: i) it can be difficult to define the combination weights in order to obtain acceptable solutions, and ii) if the optimal solution generated can not be finally accepted, new runs of the EA may be required until a suitable solution is found.

Population-based non-Pareto approaches allow to exploit the special characteristics of EAs. A non-dominated individual set is obtained instead of generating only one solution. In order to do so, the selection mechanism is changed. Generally, the best individuals according to each of the objectives are selected, and then these partial results are combined to obtain the new population. An example of a multiobjective GA of this group is Vector Evaluated Genetic Algorithm (VEGA) [31].

Finally, *Pareto-based approaches* seem to be the most active research area on multiobjective EAs nowadays. In fact, algorithms included within this family are divided in two different groups: first and second generation [6]. They all attempt to promote the generation of multiple non-dominated solutions, as the former group, but directly making use of the Pareto-optimality definition. To introduce this concept, let us consider, without loss of generality, a multiobjective minimization problem with m parameters (decision variables) and n objectives:

$$Min\ f(x) = (f_1(x), f_2(x), \ldots, f_n(x)), \quad with\ x = (x_1, x_2, \ldots, x_m) \in X$$

A decision vector $a \in X$ dominates another $b \in X$ if, and only if:

$$\forall i \in 1, 2, \ldots, n \mid f_i(a) \leq f_i(b) \quad \wedge \quad \exists j \in 1, 2, , \ldots, n \mid f_j(a) < f_j(b)$$

As said, any vector that is not dominated by any other is said to be *Pareto-optimal* or *non-dominated*.

This way, to calculate the probability of reproduction of each individual in this approach, the solutions are compared by means of the dominance relation. Different equivalence groups are defined depending on the dominance of their constituent individuals among the remainder and those individuals belonging to the "good" classes (those groups including individuals dominating a large number of the remainder) are assigned a higher selection probability than "bad" classes.

The difference between the first and the second generation of Pareto-based approaches arise on the use of elitism. Algorithms included within the first generation group, such as Niched Pareto Genetic Algorithm (NPGA), Non-dominated Sorting Genetic Algorithm (NSGA) and Multiple-Objective Genetic Algorithm (MOGA) (the one considered in this contribution), do not consider this characteristic. On the other hand, second generation Pareto-based multiobjective EAs are based on the consideration of an auxiliary population where the non-dominated solutions generated among the different iterations are stored. Examples of the latter family are Strenght Pareto EA (SPEA) and SPEA2, NSGA2 and NPGA2, among others. As can be seen, several of the latter algorithms are elitist versions of the corresponding first

generation ones. For the description of all of these algorithms, the interested reader can refer to [14, 6].

Finally, it is important to notice that, although the Pareto-based ranking correctly assigns all non-dominated individuals the same fitness, it does not guarantee that the Pareto set is uniformly sampled. When multiple equivalent optima exist, finite populations tend to converge to only one of them, due to stochastic errors in the selection process. This phenomenom is known as *genetic drift* [13]. Since preservation of diversity is crucial in the field of multiobjective optimization, several multiobjective EAs have incorporated the *niche* and *species* concepts [18] for the purpose of favouring such behaviour. We will also consider this aspect in our proposal.

3 Single-objective IQBE Processes for Extended Boolean Queries

In this section, two different IQBE algorithms to learn extended Boolean queries are reviewed. First, the well known Kraft et al.'s GP-based process, which will be used for comparison purposes in this contribution, is presented in the next subsection. Then, a variant involving the use of the GA-P algorithm is analyzed in Section 3.2. This latter algorithm will the one extended in Section 4 to build the multiobjective proposal introduced in this paper.

3.1 The Kraft et al.'s Genetic Programming-based IQBE Algorithm for Fuzzy Information Retrieval Systems

In [24], Kraft et al. proposed an IQBE process to deal with extended Boolean queries in FIRSs. The algorithm is based on GP and its components are described next[5].

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, together with the elitist selection. The intermediate population is created from the current one by means of Tournament selection [25], which involves the random selection of a number t of individuals from the current population and the choice of the best adapted of them to take one place in the new population.

[5] 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 process have been maintained.

Genetic Operators

The usual GP crossover is considered [23], 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).
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 a random size and 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 [34]):

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

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.

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.

3.2 A GA-P-based Extension of Kraft et al.'s Method

Although the algorithm proposed by Kraft et al. analyzed in the previous section obtains good results, it suffers from one of the main limitations of the GP paradigm: while this EA 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. This way, good trees solving the problem can be discarded by the selection procedure as the parameters involved in them are not well adjusted.

Hence, in the problem of extended Boolean 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 a significant drawback due to their importance in the query.

Several solutions have been proposed for this GP problem. On the one hand, one can use a local search algorithm to learn the coefficients associated to each tree in the population [23]. On the other hand, the GA-P paradigm [21], 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. Thus, 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 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 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).

Taking the previous aspect into account, in [8, 9] we introduced a new IQBE technique for learning extended Boolean queries based on the GA-P technique. The different components of this algorithm are reviewed as follows.

Coding Scheme

When considering a GA-P to learn fuzzy queries, 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 2. In our case, a real coding scheme is considered for the GA part.

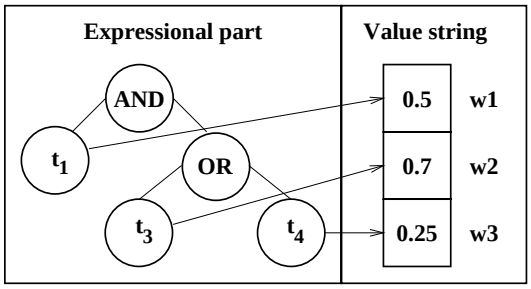

Fig. 2. GA-P individual representing the fuzzy query $0.5\, t_1\ AND\ (0.7\, t_3\ OR\ 0.25\, t_4)$

Selection Scheme

As in Kraft et al.'s algorithm, it is based on the classical generational scheme, together with the elitist selection. The intermediate population is created from the current one by means of Tournament selection.

Genetic Operators

A real-coded crossover operator – the BLX-α [15] – is considered for the GA parts. 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 with $I = max_i - min_i$ ($[min_i, max_i]$ is the interval where the $i - th$ gene is defined). In our case, $[min_i, max_i] = [0, 1]$ and the operator is always 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, 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

Both have the same definition as those in Kraft et al.'s proposal, introduced in the previous section.

4 A Pareto-based Multiobjective GA-P Algorithm to Derive Extended Boolean Queries

The Pareto-based multiobjective EA considered to be incorporated to the GA-P algorithm introduced in the previous section has been Fonseca and Fleming's MOGA [17], one of the classical, first generation multiobjective EAs. Up to our knowledge, the only previous contributions incorporating Pareto-based multiobjective techniques into a GP-based algorithm are those by Rodríguez-Vazquez et al. [27] and by Cordón et al. [10].

The selection scheme of MOGA is based on dividing the population in several ranked blocks and assigning a higher probability of selection to the blocks with a lower rank, taking into account that individuals in the same block will be equally preferable and thus will receive the same selection probability. The rank of an individual in the population (and consequently of his belonging block) will depend on the number of individuals dominating it.

Therefore, the selection scheme of our multiobjective GA-P (MOGA-P) algorithm involves the following four steps:

1. Each individual is assigned a rank equal to the number of individuals dominating it plus one (chromosomes encoding non-dominated solutions receive rank 1).
2. The population is increasingly sorted according to that rank.
3. Each individual is assigned a fitness value which depends on its ranking in the population. In this contribution, we consider the following assignment: $f(C_i) = \frac{1}{rank(C_i)}$.
4. The fitness assignment of each block (group of individuals with the same rank, i.e., which are non dominated among them) is averaged among them, so that all of them finally receive the same fitness value.

Once the final fitness values have been computed, a usual selection mechanism is applied. In this contribution we consider the tournament selection introduced in Section 3.1 with an appropriate choice of the tournament size t to induce diversity.

A said in Section 2.4, it is well known that the MOGA selection scheme can cause a large selection pressure that might produce premature convergence. Fonseca and Fleming considered this issue and suggested to use a niching method to appropriately distribute the population in the Pareto [17].

This way, in this paper we apply niching in the objective space, in order to allow the algorithm to obtain a well-distributed set of queries with a different tradeoff between precision and recall, i.e., our initial aim. To do so, we make use of the usual Euclidean metric in order to measure the closeness between two different queries on the objective space.

Once a valid metric has been selected, it is easy to apply sharing by using the classical Goldberg and Richardson's sharing function [18]:

$$F(C_i) = \frac{f(C_i)}{\sum_{j=1}^{M} Sh(d(C_i, C_j))} \quad ; \quad Sh(d) = \begin{cases} 1 - (\frac{d}{\sigma_{share}})^{\gamma}, \text{ if } d < \sigma_{share} \\ 0, \qquad\qquad \text{otherwise} \end{cases}$$

with σ_{share} being the niche radius.

5 Experiments Developed and Analysis of Results

This section is devoted to test the performance of the MOGA-P IQBE algorithm for FIRSs introduced in this paper. To do so, both the Kraft et al.'s algorithm and our MOGA-P proposal have been run to generate extended Boolean queries for the well known *Cranfield* collection, composed of 1400 documents about Aeronautics.

These 1400 documents have been automatically indexed by first extracting the non-stop words, and then applying a stemming algorithm, thus obtaining a total number of 3857 different indexing terms, and then using the normalized IDF scheme (see Section 2.2) to generate the term weights in the document representations.

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 1.

Table 1. 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

As said, apart from our MOGA-P, Kraft et al.'s IQBE process has been run on the seven selected queries for comparison purposes. In order to make this comparison fair, both algorithms have been provided with the same parameter values (see Table 2) and have been run a different number of times with different initializations till the same fixed number of fitness function evaluations have been performed.

As seen, the expressional part has been limited to 20 nodes in every case and populations are composed of 1600 individuals (the high value for this parameter is because it is well known that GP requires large population sizes to achieve good performance). For the sake of simplicity, only experiments

Table 2. Common parameter values considered

Parameter	Decision
Fitness function	F_2
Population size	1600
Number of evaluations	100000
Tournament size	16
Kraft et al.'s Crossover and Mutation probability	0.8, 0.2
Kraft et al.'s α, β weighting coefficientes in F_2	(0.8,1.2), (1,1), (1.2,0.8)
MOGA-P GA and GP Crossover probability	0.8, 0.8
MOGA-P GA and GP Mutation probability	0.2, 0.2
Expression part limited to	20 nodes
Retrieval threshold σ	0.1

not considering the use of the NOT operator are reported (as done in [24]). MOGA-P has been run ten different times for each query in order to check the robustness of the algorithm. The sharing function parameter γ takes value 2 and the niche radius σ_{share} has been experimentally set to 0.1.

On the other hand, Kraft et al.'s technique has been run considering three different values for the parameters α and β weighting, respectively, the precision and recall measures in the F_2 fitness function, in order to check the performance of the single-objective algorithm when being guided to different zones of the Pareto front. Three different runs have been done for each combination of values, thus making a total of nine runs for each query. All the runs have been performed in a 350 MHz Pentium II computer with 64 MB of RAM[6].

The results obtained by Kraft et al.'s algorithm are showed in Tables 3 and 4 respectively, with the average results being showed on the former table and the best ones on the latter. In the first table, $\#q$ stands for the corresponding query number, (α, β) for the values associated to the fitness function weigthing parameters, Sz for the average of the generated queries size and σ_{Sz} for its standard deviation, and P and R for the average of the precision and recall values (respectively, σ_P and σ_R for their standard deviations). The columns of the other table stand for the same items as well as Run for the number of the run where the reported result was derived, $\#rt$ for the number of documents retrieved by the query, and $\#rr$ for the number of relevant documents retrieved.

Tables 5 and 6 show several statistics corresponding to our multiobjective proposal. The former table collects several data about the composition of the ten Pareto sets generated for each query, always showing the averaged value and its standard deviation. From left to right, the columns contain the number of non-dominated solutions obtained ($\#p$), the number of different objective vectors (i.e., precision-recall pairs) existing among them ($\#dp$), and the values

[6] Kraft et al.'s algorithm spends more or less 13 minutes whilst MOGA-P approximately takes 15 minutes.

Table 3. Average results obtained by the single-objective Kraft et al.'s IQBE algorithm

$\#q$	(α,β)	Sz	σ_{Sz}	P	σ_P	R	σ_R
1	1.2,0.8	19	0.0	1.0	0.0	0.3103	0.0912
	1.0,1.0	19	0.0	1.0	0.0	0.2873	0.0796
	0.8,1.2	15	2.0	0.0207	0.0	1.0	0.0
2	1.2,0.8	19	0.0	1.0	0.0	0.3866	0.0230
	1.0,1.0	19	0.0	1.0	0.0	0.3333	0.0461
	0.8,1.2	18.33	1.1547	0.01785	0.0	1.0	0.0
23	1.2,0.8	19	0.0	1.0	0.0	0.3232	0.1224
	1.0,1.0	19	0.0	1.0	0.0	0.2121	0.0909
	0.8,1.2	15	4.0	0.0235	0.0	1.0	0.0
73	1.2,0.8	19	0.0	1.0	0.0	0.5079	0.0727
	1.0,1.0	19	0.0	1.0	0.0	0.5714	0.0824
	0.8,1.2	18.33	1.1547	0.015	0.0	1.0	0.0
157	1.2,0.8	19	0.0	1.0	0.0	0.2583	0.0144
	1.0,1.0	19	0.0	1.0	0.0	0.175	0.05
	0.8,1.2	16.33	2.3094	0.0285	0.0	1.0	0.0
220	1.2,0.8	19	0.0	1.0	0.0	0.5166	0.0763
	1.0,1.0	19	0.0	1.0	0.0	0.5	0.05
	0.8,1.2	18.33	1.1547	0.0446	0.0525	1.0	0
225	1.2,0.8	19	0.0	1.0	0.0	0.44	0.04
	1.0,1.0	19	0.0	1.0	0.0	0.4266	0.0923
	0.8,1.2	16.33	3.0550	0.0178	0.0	1.0	0.0

of two of the usual multiobjective EA metrics $\mathcal{M}_2^*$ and $\mathcal{M}_3^*$ [36][7], all of them followed by their respective standard deviation values.

As regards the later metrics, $\mathcal{M}_2^* \in [0, \#p]$ measures the distribution of the objective vectors of the $\#p$ non-dominated solutions in the derived Pareto set Y' (i.e., the diversity of the solutions found) by means of the following expression:

$$\mathcal{M}_2^*(Y') = \frac{1}{|Y'-1|} \sum_{p' \in Y'} |\{q' \in Y';\, ||p'-q'||^* > \sigma^*\}|$$

with $\sigma^* > 0$ being a neighborhood parameter, and $||\cdot||$ being a distance metric. In this contribution, σ^* is set to σ_{share}, the niche radius considered, and $||\cdot||$ is the Euclidean distance. Of course, the higher the value of the measure, the better the distribution of the solutions within the Pareto front in the objective space.

On the other hand, $\mathcal{M}_3^*$ estimates the range to which the Pareto front spreads out in the objective values as follows:

[7] We should note that a third metric is proposed in that paper, $\mathcal{M}_1^*$, that can not be used in this contribution as it needs from the real Pareto front in order to be computed, which is not known in this case.

Table 4. Best results obtained by the single-objective Kraft et al.'s IQBE algorithm

$\#q$	(α, β)	Run	Sz	P	R	$\#rr/\#rt$
1	(1.2,0.8)	2	19	1.0	0.4137	12/12
	(1.0,1.0)	3	19	1.0	0.3793	11/11
	(0.8,1.2)	1,2,3	13	0.0207	1.0	29/1400
2	(1.2,0.8)	1,2	19	1.0	0.4	10/10
	(1.0,1.0)	1,3	19	1.0	0.36	9/9
	(0.8,1.2)	1,2,3	17	0.0178	1.0	25/1400
23	(1.2,0.8)	1	19	1.0	0.4545	15/15
	(1.0,1.0)	3	19	1.0	0.3030	10/10
	(0.8,1.2)	1,2,3	11	0.0235	1.0	33/1400
73	(1.2,0.8)	3	19	1.0	0.5714	12/12
	(1.0,1.0)	1,2	19	1.0	0.6190	13/13
	(0.8,1.2)	1,2,3	17	0.015	1.0	21/1400
157	(1.2,0.8)	1	19	1.0	0.275	11/11
	(1.0,1.0)	2	19	1.0	0.225	9/9
	(0.8,1.2)	1,2,3	15	0.0285	1.0	40/1400
220	(1.2,0.8)	1	19	1.0	0.6	12/12
	(1.0,1.0)	2	19	1.0	0.55	11/11
	(0.8,1.2)	3	19	0.1052	1.0	20/190
225	(1.2,0.8)	1	19	1.0	0.48	12/12
	(1.0,1.0)	1,2	19	1.0	0.48	12/12
	(0.8,1.2)	1,2,3	13	0.0178	1.0	25/1400

$$\mathcal{M}_3^*(Y') = \sqrt{\sum_{i=1}^{n} \max\{||p' - q'||^*;\, p', q' \in Y'\}}$$

Since our problem is composed of just two objectives, it is equal to the distance among the objective vectors of the two outer solutions (hence, the maximum possible value is $\sqrt{2} = 1.4142$). Again, the higher the value, the larger the extent of the Pareto being covered.

Besides, two queries are selected from each Pareto set, the ones with maximum precision and maximum recall, respectively, and their averaged results are collected in Table 6.

In view of these results, the performance of our proposal is very significant. On the one hand, it overcomes the single-objective Kraft et al.'s algorithm in all cases but one (the best precision value obtained for query 23) as the results obtained by the latter when considering typical values for the weighted combination are dominated by the solutions in the Pareto front of the former, both in precision and recall. It seems that the diversity induced by the Pareto-based selection and the niching scheme make MOGA-P converge to better space zones. Notice the bad results obtained by the single-objective Kraft et al.'s process when giving more importance to the recall objective ($(\alpha, \beta) = (0.8, 1.2)$ combination), as in every case but one (query 220), a query retrieving

Table 5. Statistics of the Pareto sets obtained by the proposed MOGA-P IQBE algorithm

$\#q$	$\#p$	$\sigma_{\#p}$	$\#dp$	$\sigma_{\#dp}$	M_2^*	$\sigma_{M_2^*}$	M_3^*	$\sigma_{M_3^*}$
1	350.2	27.155	15.5	1.125	136.125	10.178	1.035	0.022
2	333.2	20.039	11.2	0.881	129.594	7.913	0.964	0.026
23	406.4	20.794	20.6	1.151	149.522	4.956	1.067	0.026
73	277.0	17.474	8.5	0.652	104.339	8.843	0.901	0.025
157	421.1	19.433	22.0	1.086	161.545	5.145	1.137	0.018
220	269.7	15.190	7.1	0.263	103.616	6.930	0.729	0.026
225	312.8	18.280	12.8	0.903	123.589	6.887	1.030	0.013

Table 6. Extreme solutions in the Pareto sets obtained by the proposed MOGA-P IQBE algorithm

	Best Precision						Best Recall					
$\#q$	Sz	σ_{Sz}	P	σ_P	R	σ_R	Sz	σ_{Sz}	P	σ_P	R	σ_R
1	19.0	0.0	1.0	0.0	0.452	0.017	18.8	0.190	0.123	0.018	1.0	0.0
2	19.0	0.0	1.0	0.0	0.464	0.014	18.2	0.580	0.200	0.025	1.0	0.0
23	19.0	0.0	1.0	0.0	0.415	0.017	17.0	0.800	0.110	0.028	1.0	0.0
73	18.8	0.190	1.0	0.0	0.676	0.018	18.8	0.190	0.161	0.023	1.0	0.0
157	18.6	0.379	1.0	0.0	0.333	0.022	18.0	0.949	0.083	0.022	1.0	0.0
220	19.0	0.0	1.0	0.0	0.680	0.016	19.0	0.0	0.346	0.025	1.0	0.0
225	19.0	0.0	1.0	0.0	0.484	0.012	19.0	0.0	0.109	0.010	1.0	0.0

the whole documentary base is obtained, thus showing the bad convergence of the algorithm when considering these weighting factor values.

On the other hand, the main aim of this paper has been clearly fulfilled since the Pareto fronts obtained are very well distributed, as demonstrated by the high number of solutions included in them and the high values in the M_2 and M_3^* metrics. Maybe the only problem found is that the number of solutions presenting different precision-recall values (different objective value arrays) can be a little bit low with respect to the large number of solutions in the Pareto set. We think that this can be solved by considering a second generation Pareto-based approach, making use of a elitist population of non-dominated solutions.

As an example, Figures 3 and 4 show the Pareto fronts obtained for queries 1 and 157, representing the precision values in the X axis and the recall ones on the Y axis. As done in [36], the Pareto sets obtained in the ten runs performed for each query were put together, and the dominated solutions where removed from the unified set before plotting the curves.

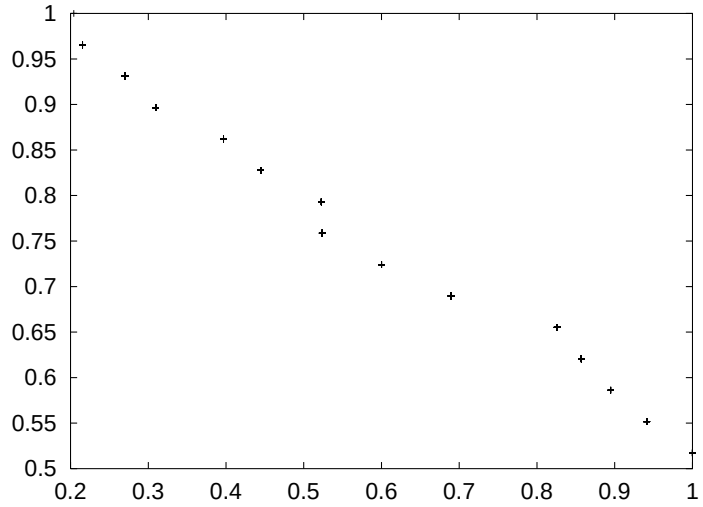

Fig. 3. Pareto front obtained for query 1

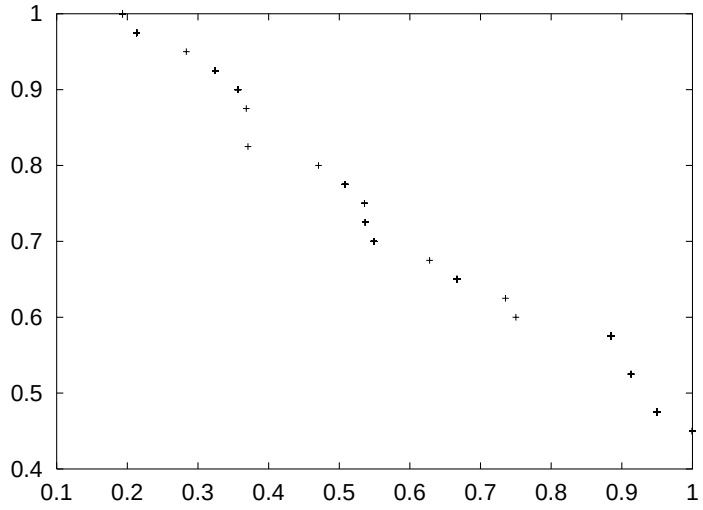

Fig. 4. Pareto front obtained for query 157

6 Concluding Remarks

The automatic derivation of extended Boolean queries has been considered by incorporating the MOGA Pareto-based multiobjective evolutionary approach to an existing GA-P-based IQBE proposal. The proposed approach has performed appropriately in seven queries of the well known Cranfield collection

in terms of absolute retrieval performance and of the quality of the obtained Pareto sets.

In our opinion, many different future works arise from this preliminary study. On the one hand, as it has been mentioned before, more advanced Pareto-based multiobjective EA schemes, such as those second generation elitist ones considering an auxiliary population to better cover the Pareto front (see in Section 2.4), can be incorporated to the basic GA-P algorithm in order to improve the performance of the multiobjective EA proposed. On the other hand, preference information of the user on the kind of queries to be derived can be included in the Pareto-based selection scheme in the form of a goal vector whose values are adapted during the evolutionary process [17]. Moreover, a training-test validation procedure can be considered to test the real-world applicability of the proposed IQBE algorithm. Finally, and more generically, Pareto-based multiobjective evolutionary optimization can be applied either to the automatic derivation of queries for other kinds of IR models or to other IR problems being solved by EAs [7], thus benefiting from the potential of these techniques in the problem solving.

References

1. Bäck T (1996) Evolutionary algorithms in theory and practice. Oxford University Press.
2. Baeza-Yates R, Ribeiro-Neto, B (1999) Modern information retrieval. Addison-Wesley.
3. 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. Physica-Verlag, pp. 231–274.
4. Chankong V, Haimes Y Y (1983) Multiobjective decision making theory and methodology. North-Holland.
5. Chen H, Shankarananrayanan G, She L, Iyer A (1998) Journal of the American Society for Information Science 49(8):693–705.
6. Coello C A, Van Veldhuizen D A, Lamant G B (2002) Evolutionary algorithms for solving multi-objective problems. Kluwer Academic Publishers.
7. Cordón O, Moya F, Zarco C (April, 1999) A brief study on the application of genetic algorithms to information retrieval (in spanish). In: Proc. Fourth International Society for Knowledge Organization (ISKO) Conference (EOCONSID'99), Granada, Spain, pp. 179–186.
8. Cordón O, Moya F, Zarco C (September, 1999) Learning queries for a fuzzy information retrieval system by means of GA-P techniques. In: Proc. EUSFLAT-ESTYLF Joint Conference, Palma de Mallorca, Spain, pp. 335–338.
9. Cordón O, Moya F, Zarco C (2000) Mathware & Soft Computing 7(2-3):309–322.
10. Cordón O, Moya F, Zarco C (2002) Soft Computing 6(5):308-319.
11. Cordón O, Herrera-Viedma E, Luque M (September, 2002) Evolutionary learning of Boolean queries by multiobjective genetic programming. In: Proc. Seventh Parallel Problem Solving from Nature (PPSN-VII) International Conference, Granada, Spain, LNCS 2439. Springer, pp. 710-719.

12. Cross V (1994) Journal of Intelligent Information Systems 3:29–56.
13. Deb K, Goldberg D E (1989) An investigation of niche and species formation in genetic function optimization. In: Proc. Third International Conference on Genetic Algorithms (ICGA'89), Hillsdale, USA, pp. 42-50.
14. Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley.
15. Eshelman L J, Schaffer J D (1993) Real-coded genetic algorithms and interval-schemata. In: Whitley L D (ed) Foundations of Genetic Algorithms 2, Morgan Kaufmann, pp. 187–202.
16. Fogel D B (1991) System identification trough simulated evolution. A machine learning approach. Ginn Press, USA.
17. Fonseca C M, Fleming P J (July, 1993) Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Proc. Fifth International Conference on Genetic Algorithms (ICGA'93), San Mateo, CA, pp. 416–423.
18. Goldberg D E, Richardson J (1987) Genetic algorithms with sharing for multimodal function optimization. In: Proc. Second International Conference on Genetic Algorithms (ICGA'87), Hillsdale, USA, pp. 41–49.
19. Gordon M, Pathak P (1999) Information Processing and Management 35(2):141–180.
20. Herrera-Viedma E (2001) Journal of the American Society for Information Science 52(6):460–475.
21. Howard L, D'Angelo D (1995) IEEE Expert: 11–15.
22. Ide E (1971) New experiments in relevance feedback. In: Salton G. (ed) The SMART Retrieval System. Prentice Hall, pp. 337–354.
23. Koza J (1992) Genetic programming. On the programming of computers by means of natural selection. The MIT Press.
24. Kraft D H, Petry F E, Buckles B P, Sadasivan T (1997) Genetic algorithms for query optimization in information retrieval: relevance feedback. In: Sanchez E, Shibata T, Zadeh L A (eds) Genetic algorithms and fuzzy logic systems. World Scientific, pp. 155–173.
25. Michalewicz Z (1996) Genetic algorithms + data structures = evolution programs. Springer-Verlag.
26. Mitchel T M (1997) Machine learning. McGraw-Hill.
27. Rodríguez-Vazquez K, Fonseca C M, Fleming P J (July, 1997) Multiobjective genetic programming: A nonlinear system identification application. In: Late Breaking Papers at the Genetic Programming 1997 Conference, Stanford, USA, pp. 207–212.
28. Salton G, McGill M J (1989) Introduction to modern information retrieval. McGraw-Hill.
29. Sanchez E (1989) Information Systems 14(6):455–464.
30. Sánchez L, Couso I, Corrales J A (2001) Information Sciences 136(1-4):175-191.
31. Schaffer J D (1985) Multiple objective optimization with vector evaluated genetic algorithms. In: Genetic algorithms and their applications. Proc. of the First International Conference on Genetic Algorithms, pp. 93-100.
32. Schwefel H-P (1995) Evolution and optimum seeking. Sixth-Generation Computer Technology Series. John Wiley and Sons.
33. Smith M P, Smith M (1997) Journal of Information Science 23(6):423–431.
34. van Rijsbergen C J (1979) Information retrieval (2nd edition). Butterworth.

35. Zadeh L A (1965) Information and Control 8:338–353.
36. Zitzler E, Deb K, Thiele L (2000) Evolutionary Computation 8(2):173–195.