A Fuzzy Genetic Algorithm Approach to an Adaptive Information Retrieval Agent

María J. Martín-Bautista* and María-Amparo Vila
Department of Computer Science and A. Intelligence, Granada University, Avenida Andalucía s/n, 18071 Granada, Spain. E-mail: {mbautis, vila}@decsai.ugr.es

Henrik Legind Larsen
Department of Computer Science, Roskilde University, P.O. Box 260, DK-4000 Roskilde. E-mail: hll@ruc.dk

We present an approach to a Genetic Information Retrieval Agent Filter (GIRAF) for documents from the Internet using a genetic algorithm (GA) with fuzzy set genes to learn the user's information needs. The population of chromosomes with fixed length represents such user's preferences. Each chromosome is associated with a fitness that may be considered the system's belief in the hypothesis that the chromosome, as a query, represents the user's information needs. In a chromosome, every gene characterizes documents by a keyword and an associated occurrence frequency, represented by a certain type of a fuzzy subset of the set of positive integers. Based on the user's evaluation of the documents retrieved by the chromosome, compared to the scores computed by the system, the fitness of the chromosomes is adjusted. A prototype of GIRAF has been developed and tested. The results of the test are discussed, and some directions for further works are pointed out.

Introduction

With the explosive growth of the amount of information resources available over the Internet, the information overload for the user has become overwhelming. With the corresponding dramatic increase of the number of users, the difficulty in assisting users in finding the best and newest information has increased exponentially. The absence of suitable alternatives in the functionality of most of current information systems can be framed as follows:

- *Lack of filtering:* A user looking for some topic on the Internet retrieves too much information.
- *Lack of ranking of retrieved documents:* The system provides no qualitative distinction between the documents.
- *Lack of support of relevance feedback:* The user can not tell his subjective evaluation of the relevance of the document.
- *Lack of personalization:* There is a need of personal systems that serve the specific interest of the users and build users' profiles (Maes, 1994).
- *Lack of adaptation:* The system should notice when the user changes his/her interests.

The agent approach (Etzioni & Weld, 1995; Maes, 1995) is now getting attention because it may help to solve the problem. We need personal search agents that do the dirty work, namely reading lots of documents to identify and display only those that are of interest to the user.

In this paper, we present an approach to such an agent, the Genetic Information Retrieval Agent Filter (GIRAF), which consists of a soft-intelligent agent that can work off-line to filter and rank the retrieved information according to the user's preferences by using Soft Computing techniques: (1) A Genetic Algorithm (GA) (Goldberg, 1989) keeps the knowledge about the user's preferences, adapts the changes in these preferences and gets the feedback from the user; and (2) Fuzzy Set theory (Zadeh, 1965) handles the imprecision of the user's preferences and the user's evaluation of the retrieved documents.

We describe the functionality and architecture of GIRAF, and explain the combination of GAs and Fuzzy Sets in the adaptive learning process. The empirical test of the model was made using case sets; the test measure and the results are presented. Finally, conclusions and some further research topics are pointed out in the last section.

Related Work

Recently, besides the traditional Internet search engines such as Yahoo, Lycos, AltaVista, etc., there have been

* To whom all correspondence should be addressed.

Received January 8, 1998; revised August 3, 1998; accepted December 16, 1998.

several systems developed to assist the user more closely to find information on the Internet. Regarding the user–system interaction, we can distinguish between two categories of systems. Those which are merely general services to help the user to gather information from the net, and those presenting an intelligent behavior, which can properly be considered as agents (Riecken, 1994). The difference between the two categories is mainly in the system's response of feedback from the user. The capabilities to learn and adapt to the user's needs are only provided by systems in the second category. These capabilities are, precisely, two of the main characteristics that make the system have intelligent behavior, as is proposed in the work (Maes, 1995).

The MetaCrawler (Selberg & Etzioni, 1995) is a service belonging to the first category that can access multiple databases and provide a larger number of potential higher quality references than any search service tied to a single database. It can be considered, just as the authors mentioned, as a meta-service over the traditional services, to gather the best references from them, and give the user a final ranked list. However, this system is a general service, and not a personal service since no user profile is built, and there is no feedback from the user to the system. Thus, we do not consider the behavior of MetaCrawler intelligent.

In the second category, the intelligent agents, we shall mention in particular two systems. The LIRA system (Balabanovic, Shoham, & Yun, 1997) uses a standard best-first search to find the best pages, based on a comparison between the user profile and the terms representing the documents. The user reads the retrieved documents and gives a feedback to the system. Another approach, the Smart Itsy Bitsy Spider (Chen, Chung, Ramsey, & Yang, 1998) also uses a best-first search for a local search, but the searching process is complemented by a genetic algorithm that develops a global stochastic search. A fitness function based on the Jaccard's similarity function (Rasmussen, 1992) is applied to determine the goodness of a given new homepage in the mutation process. The central differences between their approach and ours are the following. First, regarding the definition of the GA, they define individuals of the population are homepages, while in our system the individuals are keywords extracted from the documents. As for the functionality of the system, the main difference lays in the user evaluation in the "spider." The use of user-supplied starting pages as initialization of the population, and the lack of a real feedback to be considered as a part of the evolution process, make the modification of a query quite difficult, since the system can not adapt to the changes in the user preferences. Likewise, the personalization of the spider does not seem to imply the construction of a user profile.

The Functionality of the System

GIRAF is a personal search agent between the user and an Internet search engine. The agent can work off-line from the user and filter documents night and day. When the user interrupts, the documents are directly accessible from the hard disk of the user machine or a local server. There is no delay for connection. In this sense, the system functions as a support tool for information gathering.

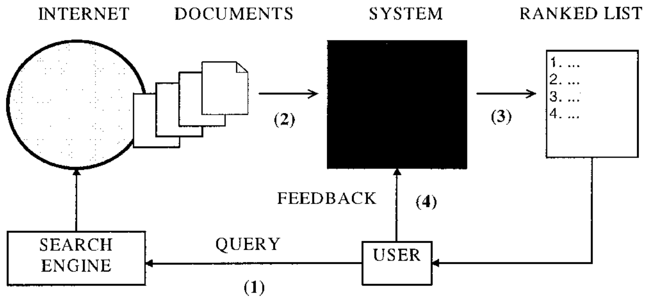


FIG. 1. Functionality of the system.

Looking at the system as a black box, the input consists of some documents from the Internet while the output consists of only the better documents, listed according to the user preferences (Figure 1).

This functionality may be characterized as follows:

- The user has needs of information in a specific topic area that he/she communicates to the search engine by a query.
- These needs must be satisfied by documents from the Internet.
- The system provides the user with a ranked list of the documents that best satisfy the needs.
- The user selects in the list the documents of interest for reading, and gives feedback on their relevance.

The goal of the system should be to give the user an update on new information about a specific topic area for a long period of time. The philosophy of the interaction between the user and the system builds on the observation that users are unable to express the exact criteria that are satisfied by, and only by, documents of interest. This is due to the unsettled nature of human preferences, the changes of the available information, and the human mental model that is highly complex, and therefore difficult to communicate.

To derive the answer to the user's information need, the systems first retrieve a set of potentially relevant documents from the Internet. This retrieval is efficient in terms of high recall from the Internet, and a fast response time at the cost of poor precision. Recall is the percentage of relevant documents that are retrieved, while precision is the percentage of documents retrieved that are considered relevant (Salton & McGill, 1983). When the relevance is treated as a fuzzy concept, as in our case, the recall and precision must be considered in terms of ranked documents, as is suggested in Kraft, Petry, Buckles, & Sadasivan (1995). In order to increase the precision, the retrieved high-recall set is filtered through ranking by the scores of its documents. These scores are given by the population of the GA, which represents the information needs of the user. The result to be presented to the user is a subset characterized by a high recall and precision for these information needs. This subset may then be stored as the answer that the user can browse and read as desired.

The capability of the system is to approach an exact representation of the user's information needs automatically. For this purpose, the user evaluates the relevance of the retrieved documents, and gives feedback to the system, which utilizes it to adapt to the evolving information needs. By means of the GA, the agent adjusts the representation of these needs when confronted with user feedback. Hence, the agent should be adaptive in the sense that it learns a representation of the user's information needs and maintains this representation to keep track of the evolution of these needs (Mitchell & Forrest, 1994).

In order to adjust the representation of the needs when confronted with the user feedback, we propose a specially designed GA. Every chromosome in the population of the genetic algorithm contains a set of genes. Each gene characterizes a fuzzy subset of the document set by means of a keyword (term) and its occurrence frequency in a document. The occurrence frequency is represented by a fuzzy subset of the set of non-negative integers. This process is based on the concept of relevance feedback (Salton & Buckley, 1990), where those genes with terms in documents that are evaluated as good are rewarded, while those ones that the user considers without relevance are penalized. Each chromosome is associated with a fitness value representing the system's belief in the hypothesis that the chromosome, as a query, represents the information needs. The user evaluates the relevance of the documents retrieved, and the system uses this relevance feedback to adjust the fitness of the chromosomes that contain these genes.

The Architecture of GIRAF

The GIRAF system contains four main modules, one for each of the central functionality maintained by the system: parsing, learning, evaluation, and man-machine interaction (user interface). An overview of the architecture of GIRAF is shown in Figure 2. To start the system, the user selects and evaluates one or more documents. This information is applied to the system to provide a starting point for the population adaptation to the information needs of the user. The modules are described in the following:

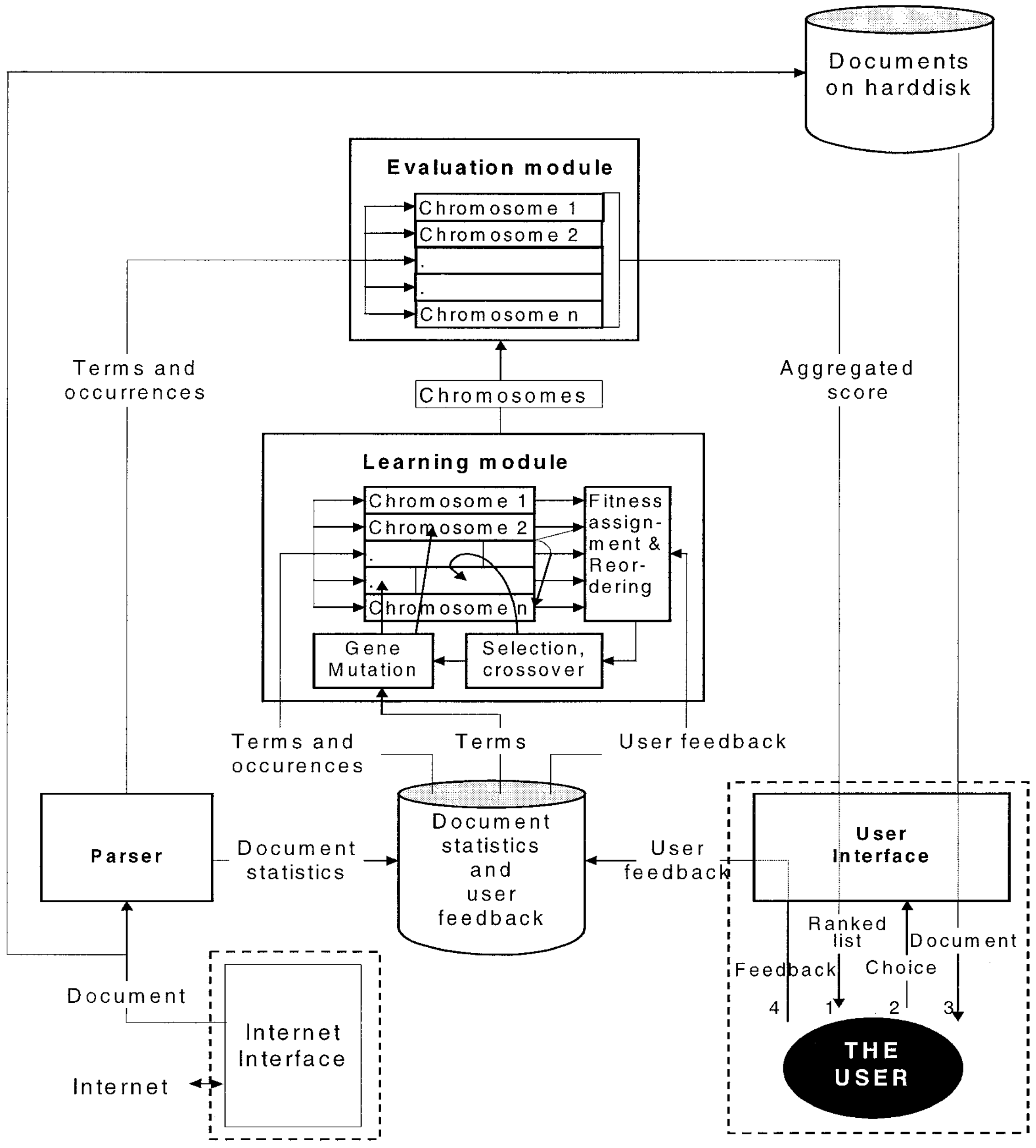
The *Parser Module* extracts the words (terms) from the documents and maintains statistics on word occurrences in the documents.

The *Learning Module* is the central element of the architecture and is composed of a genetic algorithm for modeling adaptive and exploratory behavior. The main functionality of this module is to adjust the representation of the information needs of the user so it is consistent with the latest feedback values of the user, and yet retains the essential knowledge from the past. This knowledge is kept in a population of chromosomes, which is processed by the *genetic algorithm*. Each *chromosome* is a hypothesis on how to evaluate a document according to the information needs. All the chromosomes in the population are competing to predict the user's satisfaction from a document. The chromosome's evaluation of a document is called the *chromosome score* and the ability of a chromosome to classify a document is called the *fitness* of the chromosome. Each gene of a chromosome is specified by a triple (t, g, c), where t is a term (a word extracted from some document), g is the gene type, to be introduced in next section, and c is a positive real number. The pair (g, c) determines a fuzzy subset of the non-negative integers characterizing the term occurrence frequency in interesting documents. The same term may be applied in more than one gene in the same chromosome, as well as in different chromosomes. The *chromosome score* is calculated as the average of the gene evaluation of a document. In the following section, we give a more detailed description of the learning module.

The initialization of the population occurs as follows: For every new gene created in a chromosome, a random term t is selected from the pool of all the indexing terms extracted from a set of documents. These documents may be either the most relevant ones retrieved from the net, or an ideal document supplied by the user to the system. The type g of the gene with the selected term is determined randomly as well, but based on an initially fixed distribution of genes on gene types. In the present system, the value of c will be initially determined by the average of the occurrence frequency of the term t in all the documents analyzed. However, the normalization of the parameter c was not considered in the initial model, being calculated by the average of the number of occurrences of the term t in all the documents.

The *Evaluation Module* assigns a *score* to documents by using the present information in the population of chromosomes. The set of retrieved documents is assumed to represent an answer with high recall but low precision. The evaluation module then applies the genetic evolution to evaluate the documents in this answer to retain a subset that represents both high recall and high precision.

The User Interface Module presents documents found by the system, based on the current hypotheses. The user is asked to give relevance feedback on presented documents by rating, for each document read, how satisfying he finds it. For this purpose we may provide a set of feedback buttons, for instance, four buttons representing the four linguistic labels, respectively, "poor," "moderate," "good," and "very good" (Yager, 1996). The feedback is transformed into a numeric value for further processing. From the resulting best hypotheses (represented by the fittest chromosomes), the Internet Interface may construct queries, based on the terms of the best chromosome, to be submitted to an external search engine (somewhere on the Internet) to retrieve a number of potential relevant documents. Therefore, a query may be an AND aggregation of the terms of the chromosome. The terms provided by genes of type 2 must not appear in the document, thus, they will be represented as negated terms (NOT) in the query.



(parts not included in the present system are surrounded by dotted lines)

FIG. 2. Architecture of GIRAF.

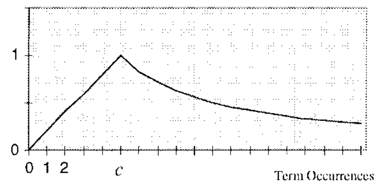
The Learning Module

The learning module incorporates the adaptive component of the system by means of a GA with fuzzy set genes. GAs are adaptive search and optimization algorithms that work by mimicking the principles of natural genetics (Deb, 1996). In our case, the function to be optimized is a hypothetical representation of the needs of a user looking for information in the Internet.

In the following, we present central elements of the GA model applied in GIRAF, namely: the fuzzy gene types, the GA operators and the fitness function.

The Fuzzy Set Genes

A gene *G (t, g, c)* characterizes documents by occurrence frequency of the term (word) *t* in the document. The parameters *g* and *c* determine a fuzzy set characterization of the number of occurrences, namely as a fuzzy subset of the set of non-negative integers. The parameter *g* identifies the gene type, that is the basic shape of the fuzzy set as described below for the four basic shapes applied in GIRAF. Finally, the parameter *c* is a non-negative real number that determines the fuzzy set in combination with the gene type.



$$\mu_{\approx}(x) = \frac{min(x,c)}{max(x,c)}$$
$$\forall x \in Z_0^+$$

FIG. 3. Membership function applied by gene type 1.

The normalization of the parameter c has not been considered in these preliminary tests. This problem may be studied in future experiments. In the following, we present the four gene types, where x is the occurrence frequency of the term t in the document.

Gene Type 1: G (t,≈,c). This gene type represents the occurrences of a term that the user likes. In this case c is the number of times that the term t must appear in the document to satisfy this gene completely. The shape and definition of the membership function applied by this gene type are shown in Figure 3.

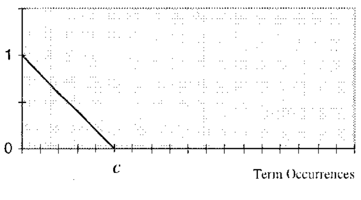
We notice that the membership function for this gene type is not symmetric. This is mainly due to the deviation sizes of the considered occurrences of a term, depending on whether they are right or left deviation. This asymmetry emphasizes right deviations, which are more desirable since they imply occurrences of the term in the document greater than the ideal number of times c. The effect of the shape of the function is that the term represented has to appear close to the ideal to influence the chromosome evaluation with the maximum degree.

Gene Type 2: G (t,<,c). This gene type is completely satisfied by documents that have no occurrences of the term t. The satisfaction is linearly decreasing from one to zero, as the number of occurrences increases from zero to c, as presented in Figure 4.

The function of this gene type is to eliminate documents dealing with topics in which the user is not interested.

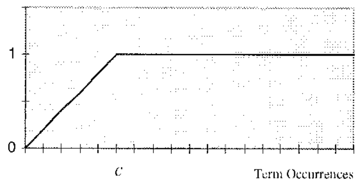
Gene Type 3: G (t,≥,c). Genes of this type are satisfied completely by documents with at least c occurrences of the term t. The gene type 3 is similar to gene type 1, except that the satisfaction does not decrease, as the number of occurrences increases above c, but remain "complete" (Figure 5).

Gene Type 4: G (t,$\approx_{OR}$,c). By this gene type, we allow to take account for the situation that the significance of term



$$\mu_{<}(x) = \begin{cases} \frac{c-x}{c} & x < c \\ 0 & x \geq c \end{cases}$$
$$\forall x \in Z_0^+$$

FIG. 4. Membership function applied by gene type 2.



$$\mu_{\geq}(x) = \begin{cases} \frac{x}{c} & x < c \\ 1 & x \geq c \end{cases}$$
$$\forall x \in Z_0^+$$

FIG. 5. Membership function applied by gene type 3.

occurrences is different in the first part, the central part, and the last part of a document (Bordogna, Carrara, & Pasi, 1995). The division of the documents into three parts has been considered in a general way, without taking into account, at first, the amount of different information representations with which the system can deal. In future works, the structure of different types of documents may be considered, and a specific division for every one could be established. In the present system, for a general purpose, we define the first part, P_1, as the first 10% of words in the document, the central part, P_2, as the 80% of the words, and the last part, P_3, as the remaining 10% of them. The satisfaction of a gene *G (t,$\approx_{OR}$,c)* is defined as a weighted OR-aggregation of satisfaction of *G (t,≈,c)* by the three parts.

The GA Operators

The GA operators are selection, crossover, and mutation. Selection deals with the choice of chromosomes of the population that will reproduce. The crossover takes sequences of genes from each of two parent chromosomes selected and combines them to create an offspring chromosome. The mutation is the random alteration of a gene in the chromosome selected. Crossover and mutation are needed for exploitation, respectively, exploration of the search space (Davis, 1991; Goldberg, 1989; Spears, 1993). The three operators are described in the following.

Selection. For a crossover, we select two different chromosomes as parents, to produce a new chromosome as their offspring. For the selection, we first order the chromosomes in decreasing order of their fitness. For the first of the two parents, we select a chromosome in a random position in the ordered set. For the second parent, we select the chromosome in a random position between position of the fittest chromosome and the position of the first parent. By this selection strategy, we obtain that chromosomes with a higher fitness are more likely to become parents than chromosomes with a lower fitness, as in most of selection strategies (Goldberg & Deb, 1991).

For a mutation, we select one chromosome randomly from the whole population. The size of the population is maintained constant by deleting the chromosome with the lowest fitness each time a new chromosome is generated by crossover or by mutation.

Crossover. For a crossover, we first randomly choose the gene positions to be applied as crossover positions. In

GIRAF, each gene position in the chromosome is chosen with the probability 0.2. Thus, if the length of the chromosome is n, the expected number of positions chosen as crossover points is $0.2n$. The crossover is now completed as follows: starting at the first gene position, we copy all genes from the first parent to the same position in the offspring chromosome, until we have met and copied the gene at the first crossover position. We now repeat the procedure from the following position until the next crossover position, except that we now copy from the second of the two parent chromosomes, and so on.

The chromosome described above is a general case of the multi-point crossover operator in the sense that the number of cross points changes for every two chromosomes, and the number of cross points may be at most the length of the chromosome (DeJong & Spears, 1992).

Mutation. In GIRAF, the mutation is performed as follows: in the chromosome selected for mutation, we choose randomly a gene position for the mutation. The gene $G\ (t,g,c)$ in the chosen position is replaced by another gene $G\ (t',g',c')$ where t' is chosen randomly from the terms occurring in a document from the pool of all the terms considered in the parser tree, the gene type is unchanged ($g' = g$), and the parameter c' is set to number of occurrences of t' in the document from which it was chosen. In a future work, the most frequent term (different from the one to be replaced) from the considered document may be introduced as a new term in the mutation operator.

The Fitness Function

The fitness function measures the adaptation of every chromosome of the population in each iteration of the evolution process. Let us suppose a population of m chromosomes that evolves for k generations; and in the j'th generation, a new document (is retrieved from the net. The i'th chromosome, which is supposed to contain l genes with the terms closest to the user needs, evaluates the document in order to predict the user's satisfaction from that document. The value that the chromosome gives to a document is called *chromosome score*, considered as the result of the evolution of a query. As the chromosome is set of genes, which may be considered as the weighted terms of the query, this value must be calculated as some kind of aggregation of the fuzzy values from every gene. Specifically, we use the arithmetic average of the genes, as follows:

$$C_i^j\ (\omega) = \frac{1}{l} \cdot \sum_{h=1}^{l} \mu_g^h\ (x) \quad , \quad i = 1,\ldots,m;\ j = 1,\ldots,k$$

where:

- $C_i^j\ (\omega)$ is the i'th chromosome score of the document ω in the generation j.
- l represents the length of the chromosome, that is, the number of the genes that the chromosome contains.
- μ_g^h is the value of the membership function of the h'th gene of type g.

Initially, the chromosome score establishes the fitness value (f_i^j) of the chromosome i in the generation j. The result of the evolution process through a generation is a new generation where the chromosomes with higher fitness are expected to appear more frequently than those with lower fitness. However, in this way the population may contain several copies of the same chromosomes with high fitness, which rapidly push chromosomes with lower fitness out of the population. Therefore, the population loses diversity. This may provide an undesired effect when the fitness of the chromosomes is only based on documents and scores that are being updated when the user interacts with the system, because chromosomes with lower fitness tend to be pushed out immediately, although they may have the potential to increase their fitness in later generations.

In order to allow such chromosomes to survive through several generations, we keep the fitness of every chromosome through the generations, but modifying it by means of the addition of a *payoff (P)* to increase the accumulated fitness of every chromosome, and the subtraction of a certain *lifetax (L)* which prohibit to survive forever if the payoff continues to be low. Therefore, the general expression of the fitness function is as follows:

$$f_i^j = f_i^{j-1} + P_i^j - L_i^j \tag{1}$$

where f_i^j is the fitness of the i'th chromosome ($i = 1,2,\ldots,m$) in the j'th generation ($j = 1,2,\ldots,k$), P_i^j and L_i^j are the payoff and the lifetax of chromosome, respectively. For the initial generation, ($j = 0$) we set $f_i^j = f_i^0 = 0$, and for new chromosomes created in a generation shift (from mutation or crossover), we apply a special initialization: the fitness of the new chromosome is set to the half of the maximum fitness in the generation in which the chromosome was created, and the updating function (1) is first applied from the following generation.

The lifetax L_i^j is represented by the minimum fitness of the chromosomes in the population. As for the payoff P_i^j, several expressions have been set during the development of the system.

Initially, the payoff may be calculated by the relation between the chromosome score and the user feedback of the document, so the smaller the distance between these values is, the higher the payoff the chromosome will be. Therefore, the first payoff function may be defined by:

$$P_i^j = 1 - \left| C_i^j - U^j\ (\omega) \right| \tag{2}$$

where $C_i^j\ (\omega)$ is the i'th chromosome score, and $U^j(\omega)$ is the user's evaluation of the document ω evaluated, i.e., the feedback that the user gives to the system about the document ω.

In order to deal with the premature convergence of the GA to a local optimum, we may give extra credit to high payoff values. This may be done by modifying (2) to:

$$(P_i^j) = 1 - (C_i^j\,(\omega) - U^j\,(\omega))^2 \tag{3}$$

Furthermore, we would like to give a special reward for handling "problematic documents," characterized as documents in which global evaluation of the whole population (in the last generation, applied to rank the documents) is far from the user's evaluation. The value representing the global population's evaluation of a document is called *population score (S)*, and to calculate it, an arithmetic average of the best chromosomes is obtained. The percentage of chromosomes to be considered as the best may be variable, and initially, is fixed to the best 40 percent chromosomes of the population. The ability of the population to classify a document is estimated by the following measure of the ability in the *j-1′*th generation (i.e., in the previous iteration):

$$A^{j-1} = (S^{j-1}\,(\omega) - U^j\,(\omega))^2 \tag{4}$$

where $S^{j-1}(\omega)$ is the population' score of the document in the $j-1$'th generation, and $U^j(\omega)$ is the user's evaluation of the document in j'th (i.e., the current) generation.

As the final payoff function, we propose $(P_i^j)''$—a weighted combination of (3) and (4), namely:

$$(P_i^j)'' = ((P_i^j)')^{w_1}\,(A^{j-1})^{w_2} \tag{5}$$

where w_1 and w_2, both from the unit interval, are the importances of satisfying (P_i^j)' and A_i^{j-1} (Yager, 1978), and may be set by an expert.

The Test of the System

System Parameters Tested

The parameters of the system for testing are:

- size m of the population of chromosomes
- number n of genes in each chromosome
- probability of a gene to be a cross point
- probability of a chromosome to be selected to mutate
- probability of a gene to mutate in the chromosome
- the payoff function
- the distribution of genes over the gene types

We tested the effect of changing each of these parameters, as well as certain combinations of them, leaving the other parameters unchanged.

Performance Measures Considered

Two performance qualities of central importance for evaluation purposes are prediction precision and recapitulation ability:

The *Prediction Precision* measures the ability to, as accurately as possible, predict the feedback from the user. To test the system, we have to expose it to a number of situations that should reveal its ability to reason and learn in such a way that it satisfies this requirement. Thus, the system should be able to spot the similarities between documents despite their apparent differences, be able to spot the differences between documents despite their apparent

TABLE 1. Search history in the test example. Preferences: "agents-general information." (U = user evaluation).

Document title	Http address	U
1 IBM Intelligent Agent Strategy	http://activist.gpl.ibm.com/WhitePaper/ptc2.htm	0.9
2 Letizia: An Agent That Assists Web Browsing	http://lcs.www.media.mit.edu/people/lieber/Lieberary/Letizia/Letizia.html	0.9
3 The TkWWW Robot: Beyond Browsing	http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/Agents/spetka/spetka.html	0.9
4 Technical rationale	http://www.osf.org/ri/contracts/6.Rationale.frame.html	0.8
5 The @gency	http://www.info.unicaen.fr/~serge/sma.html	0.8
6 Free Agent 1.0 Tech Notes July 1995	http://phonebk.duke.edu/clients/tnfagent.html	0.7
7 WEBDOGGIE Personalized	http://webhound.www.media.mit.edu/projects/webhound/doc/Webhound.html	0.7
8 MMM—a WWW based tool for using remote	http://mmm.wiwi.hu-berlin.de/MMM/cebit engl.html	0.6
9 Adaptive Agents for Information	http://www.cs.umbc.edu/~cikm/iia/submitted/viewing/chen.html	0.6
10 Autonomous Agents	http://www.psychology.nottingham.ac.uk:80/aigr/research/agents/agents.html	0.6
11 Autonomous Agents	http://www.elet.polimi.it/section/compeng/air/agents/	0.6
12 International Society for Adaptive	http://netq.rowland.org/isab/isab.html	0.5
13 Web Links	http://www.cs.bham.ac.uk/~ämw/agents/links/	0.5
14 Julia's utility: simple examples	http://foner.www.media.mit.edu/people/foner/Julia/subsection3_2_2.html	0.4
15 Extended Abstract Pramod Jain	http://www.cs.umbc.edu/~cikm/1994/iia/papers/jain.html	0.4
16 Agents Info	http://www.cs.bham.ac.uk/~ämw/agents/index.html	0.4
17 Distributed Agent-Based approach	http://groucho.gsfc.nasa.gov/Code_520/Code_522/Projects/Agents/	0.4
18 Firefly	http://www.ffly.com/html/About1.html	0.2
19 Agent Gallery	http://www.hinet.com/realty/edge/gallery.html	0.1
20 Sales Agents Needed	http://maple.net/gbd/salagnts.html	0.1

TABLE 2. Test set 1. Preferences: "agents-general information."

Document title	Http address	U
21 Intelligent Software Agents	http://www.cs.umbc.edu/agents/	0.9
22 What's an Intelligent Agent?	http://www.yourcommand.com/ia int.htm	0.7
23 Agent theory—philosophy, formalisms, . . .	http://www.cs.umbc.edu/agents/theory/	0.6
24 Multi-Agent Systems	http://www.sd.monash.edu.au/~bdurnota/agents.html	0.3
25 Bus Pass Agent	http://ursu.uregina.ca/Services/SUServices.html	0.1

similarities, and be able to classify new types of documents close to the user's evaluation of the documents.

The *Recapitulation Ability* measures the degree to which the system adjusts itself with precision and speed to changes in the environment, that is, changes in user preferences (information needs) and available information. Of particular importance for this measure is how the system is able to improve itself when confronted with a user's surprising feedback value.

For the test of the prediction precision, we deal with sets of documents comprising two classes, namely "bad" and "good" documents (concerning the information needs). The bad documents have the user's evaluation of 0.1, while the good documents have the user evaluation of 0.9. The system should provide the same classification, approaching the user's evaluation as much as possible. This quality is measured by the *classification predictability (CP)*, and is defined as one minus the dispersion of the "error" $|S(\omega) - U(\omega)|$ where $S(\omega)$ and $U(\omega)$ are the system evaluation (the population' score) and the user's evaluation, respectively, of the i'th document in the evaluated set:

$$CP_p = 1 - \sqrt{\frac{1}{d-1}\sum_{i=1}^{d}(S_i(\omega) - U_i(\omega))^2} \qquad (6)$$

where the subscript p of CP refers to the set of parameters applied by the system, and d is the number of documents in the evaluated set.

Test Example

By a systematic test, we emulated a period with a user interacting with the system. Some of the documents were carefully selected in the topic area of "Intelligent Internet Information Systems"; later the system was supplemented by other documents. The total case sets comprised 20 documents, to which we assigned the score 0.1 or 0.9 manually, modeling the user's evaluation of the document. As the initial information needs, we assumed the topic "software agents." As in a realistic situation, the needs were refined and modified as we received more information.

The Virtual User Search History

The test should not begin before the system has developed a set of chromosomes of a certain quality. This will minimize the risk of initial incidents to have a great impact on the test results. The less developed the structure of the GA, the greater is the possibility that small events have greater impact on the evolution.

In order to avoid initial disturbances, the test module is programmed to feed a virtual history into the learning module, so that the system can build up a reliable mapping of the virtual user preferences before the system is tested.

The search history (Table 1) consists of 20 documents with an assigned score. The scores have been assigned between 0.1 and 0.9, and the documents have been carefully chosen to make the most complex and realistic test.

Hence, the virtual user therefore can be specified as a person that wants information on the topic of software agents. As a real person, the user preferences change as the user receives information. In other words, the user does not initially know his specific preferences on the topic; his preferences evolve as the user receives more and more information.

The parameter combination undergoes a first test; a population that adapts to the documents and their assigned scores is created. An ability threshold and an iteration threshold are fixed; when any of them is reached for all the documents, the search history is complete and the population is ready to test.

TABLE 3. Test set 5. Preferences: "adaptive agents."

Document title	Http address	U
41 An Endogenous Fitness Paradigm for Adaptive	http://www-cse.ucsd.edu/users/fil/agents/info-spiders.html	0.9
42 The SodaBot Homepage	http://www.ai.mit.edu/people/sodabot/sodabot.html	0.6
43 Agents of Change	http://www.byte.com/art/9503/sec10/art1.htm	0.6
44 The Development of Intelligent Autonomous	http://www.ai.univie.ac.at/oefai/agents.html	0.3
45 Software agents are on-line pseudo-people	http://www.ai.mit.edu/people/mhcoen/agents/subsection2 1 1 1.html#SECTION0011100000000000000	0.3

TABLE 4. Test set 9. Preferences: “mobile agents and agent communication.”

Document title	Http address	U
61 Mobile Unstructured Business Object (MuBot)	http://www.crystaliz.com/logicware/mubot.html	0.9
62 INTELLIGENT SOFTWARE AGENTS	http://www.csd.abdn.ac.uk/research/intelligent_agents.html	0.7
63 Learning Agents for Information Filtering	http://www.csd.abdn.ac.uk/~pedwards/res/filter.html	0.6
64 Cooperative Research	http://mizo01.ia.noda.sut.ac.jp/Research/Coop/Coop_index_e.html	0.4
65 Autonomous system project	http://www-iiia.unine.ch/IA.GRP/autonomous.html	0.4

Test Procedure

In this example, let us suppose that the system is fed with 13 test sets, each one with five documents. Once the search history is formed, a set of five new documents comes from the Internet (see Table 2), and the evaluation module has then to assess them without any information about the virtual user feedback value. The closer the evaluations are to the assigned scores, the better will be the Prediction Precision performance of the parameter combination will be. When the virtual user receives these documents, its gives the feedback to the system, and the population evolve to reach again the ability or the iteration threshold again.

The first four test sets are chosen so that they emulate that the user preferences have not changed since the search history. However, in order to test the Recapitulation Ability, we emulate a change in the user preferences (see Table 3). Let’s suppose that the user has received enough general information about “software agents,” and the utility from such information is decreasing because a new document on more specific “adaptation” techniques for “software agents” appears in test set 1, and triggers the user curiosity, so he now feels that he would like to delve deeply into this new topic. So the virtual user assigns scores to the documents, giving max value (0.9) on good information about “adaptive agents,” and less value to more general information about software agents. The population should now be able to modify it’s chromosomes quickly. The higher the diversity of the population is, the higher the Recapitulation Ability will be, and the less information it already contains about “adaptation,” the slower it will recapitulate.

In test set 9 (Table 4), the virtual user’s preferences are changed towards agent mobility and communication because a document (number 61) announces new possibilities in this area.

When the last five documents are received (see Table 5), the virtual user feels updated in most topics concerning agents. He decides to change his preferences to get more information about machine learning techniques usable in agents. The user finds a document concerning GAs and agents, and gives it the highest score. This topic has been touched before in a more general manner in test set 5–8 (which have not been included in this paper), so the system should be able to recapitulate quickly and evolves towards these new preferences.

Experimentation

Test Case Formulation

Preliminary tests have been made due to the several possible combinations of parameters to find a reasonable starting point for the systematic test, which comprised 220 runs with different parameter sets. The starting parameter combination is:

- number of iterations: 200
- size of the population (N): 80
- number of genes in the chromosome (L): 40
- probability of crossover (P_c): 0.01
- probability to select a chromosome for mutation (P_m): 0.01
- probability of a gene to mutate: 1/chromosome length
- proportion of genes for every type ($\%g$): 25%.
- ability threshold to stop: 0.9
- iteration threshold to stop: 200

Summary of Test Results

In the following, we present some interesting results and observations from the prediction precision and the recapitulation ability tests.

Prediction Precision Test. To test the prediction precision on all parameter combinations we use the measure called *Classification Predictability* described in the previous

TABLE 5. Test set 13. Preferences: “agent learning.”

Document title	Http address	U
81 The Genetic Algorithms Group	http://www.cs.gmu.edu/gag/index.html	0.9
82 Online Workshop on EVOLUTIONARY	http://www.bioele.nuee.nagoya-u.ac.jp/wec/papers/index.html	0.7
83 MAPS—Multi-Agent Problem Solver	http://expasy.hcuge.ch/sgaico/html/olb/Sources/Maps/Maps.html	0.4
84 Expersys paper abstract	http://expasy.hcuge.ch/sgaico/html/olb/Sources/Publications/Expersys.html	0.1
85 COALA—COoperative Agent Language	http://expasy.hcuge.ch/sgaico/html/olb/Sources/Coala/Coala.html	0.1

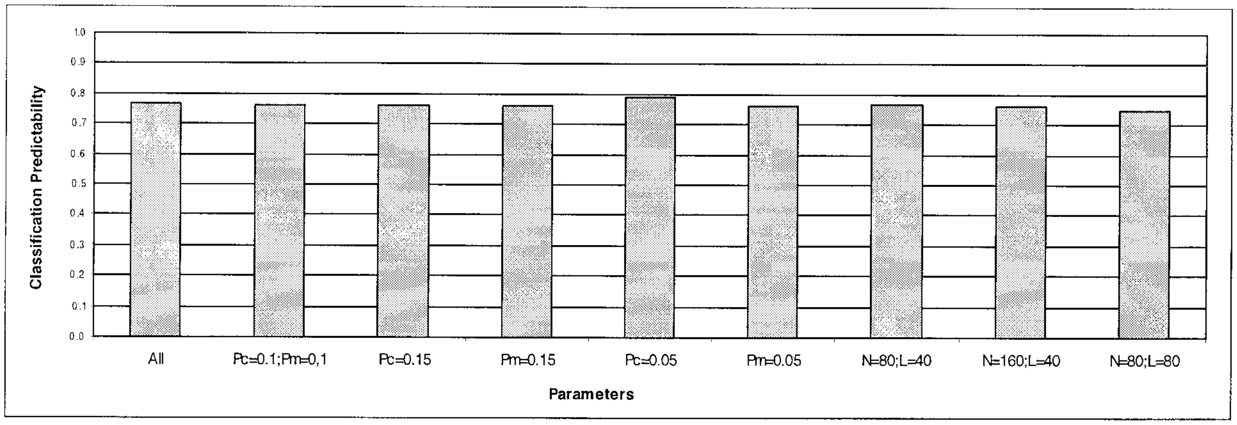


FIG. 6. Search history in the test example. Preferences: "Agents-General Information." (U=User Evaluation).

section. The results of the test—referring to the genetic parameters such as the probability of mutation (P_m), probability of crossover (P_c), the size of the population (N) and the length of the chromosomes (L)—can be observed in Figure 6, while those parameters referred to the payoff function and the percentage of gene of every type can be observed in Figure 7. Findings to be mentioned are:

- Changing the crossover and mutation probabilities showed no significant changes in the classification predictability.
- An increase in the occurrence of gene type 2 showed the poorest performance.
- Gene type 3 showed the best performance.
- We expected payoff function P'' to perform better than P' and payoff function P to be the worst. However, payoff function P presents the best performance , while P' and P'' performed equally well.
- Specially, a lowering of the crossover probability, an increasing of the occurrence of gene types 3 and 4 and the use the fitness function with payoff function P increase the classification predictability.

Recapitulation Ability Test. To test the recapitulation ability we look only at the performance after the parameter combinations has been through several changes of preferences. Since in the Prediction Precision Test, the results concerning the genetic parameters show no significant changes in the performance of the system, we have focused this part of the test in the payoff function and the type of the genes. In general, as the number of documents increases, the system improves its performance with a lowering of the probability of the crossover, and with an increasing of the probability of the mutation.

Other new results appear now (see Fig. 8):

- Gene type 1 now has the best performance.
- Payoff function P'' has better recapitulation ability than payoff function P' and this one is better than P. This is the reverse of the prediction precision, and it is probably due to the combined effect obtained by formula (5).

Concluding Remarks

Important conclusions can be drawn from the experiments. The first is that the presented approach to an adaptive information retrieval agent is a potential viable approach to a system that can learn the information needs of the user,

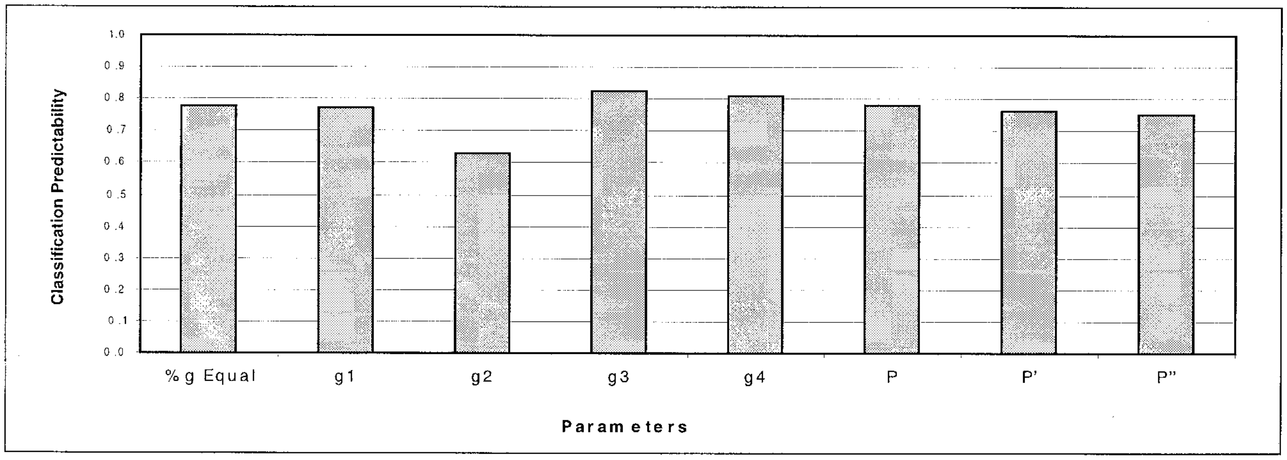


FIG. 7. Test set 1. Preferences: "Agents-General Information."

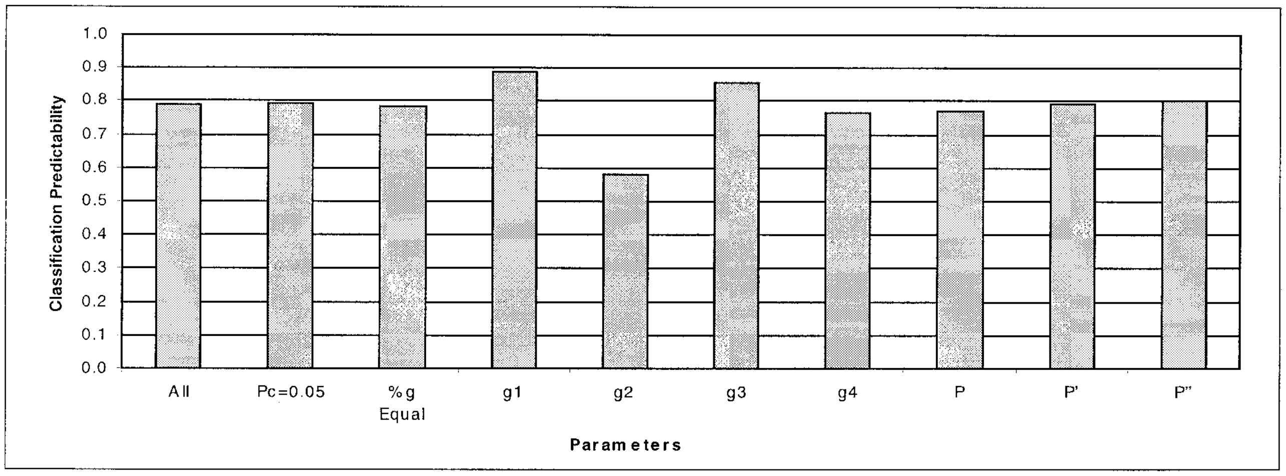


FIG. 8. Test set 5. Preferences: "Adaptive Agents."

and keep up with the evolution of these needs, utilizing only relevance feedback from the user.

The second conclusion is that, in general, gene type 3 is the best gene to predict the users' preferences on documents, but considering the recapitulation ability test, we also can conclude that gene type 1 has some advances.

Preliminary tests showed that, when the appearance of gene types in chromosomes were not fixed, genes of type 2 would drive out all other gene types. The problem only increases with the very bad performance of gene type 2 in the systematic tests. The gene type appears to utilize "loop holes" by assigning medium evaluation to all documents and thereby surviving by "not putting anything at stake." Future developments must try to punish this kind of behavior because it leads to poor system answers. As mentioned, the easy way to avoid the spreading of gene type 2 was to fix the number of genes of each type in the chromosomes.

Another conclusion is that the changes from fewer documents in our preliminary tests to the larger number in our systematic tests in connection with a performance increase when the crossover probability is lowered, may indicate some connection between this probability and the size of the set of documents processed by the system. The test indicates that a higher probability mutation than traditional one (Holland, 1992) is needed as the number of documents increases. This shows a higher need for exploration than exploitation in our search space system (Spears, 1993).

Finally, we can conclude that by an appropriate parameter setting, it is possible to satisfy both quality requirements, prediction precision, and recapitulation ability, to a higher degree. This includes satisfying the central issue of prediction precision, namely that the system's ranking of the documents is close to the ranking determined by the user's evaluation of these documents.

Future Work

Some directions for further research and future work may be:

- To reduce the size of the chromosome, by giving higher priority to terms with a high degree of discrimination, that is, ability to differentiate the documents, in selection of terms for the new genes.
- To control the large amount of system parameters applied by the agent system.
- To develop pattern recognition genes, in order to extend the capability of the agent from search in text to search for patterns in graphics, video, and voice.
- To associate a set of terms with a certain user, and build a profile of the user with different populations, each related to a general topic with the options to start a new agent or to run an old one.

References

Balabanović, M., Shoham, Y., & Yun, Y. (1997). An adaptive agent for automated web browsing. Stanford University Technical Report CS-TN-97-52.

Bordogna, G., Carrara, P., & Pasi, G. (1995). Fuzzy approaches to extend Boolean information retrieval. In P. Bosc, & J. Kacprzyck (Eds.), Fuzziness in database management systems (pp. 231–274), Germany: Physica-Verlag.

Chen, H., Chung, Y., Ramsey, M., & Yang, C.C. (1998). A smart Itsy Bitsy Spider for the Web. Journal of the American Society for Information Science, 49, 604–618.

Davis, L. (Ed.). (1991). Handbook of genetic algorithms. New York: Van Nostrand Reinhold.

Deb, K. (1996). Genetic algorithms for function optimization. In F. Herrera & J.L. Verdegay (Eds.), Genetic algorithms and soft computing (pp. 3–29), Germany: Physica-Verlag.

DeJong, K.A., & Spears, W.M. (1992). A formal analysis of the role of multi-point crossover in genetic algorithms. Annals of Mathematics and Artificial Intelligence Journal, 5, 1–26.

Etzioni, O & Weld, D.S. (1995). Intelligent agents on the internet: Fact, fiction, and forecast. IEEE Expert, August 1995, 44–49.

Goldberg, D.E. (1989). Genetic algorithms in search, optimization and machine learning. Addison-Wesley.

Goldberg, D.E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In G.J.E. Rawlins (Ed.), Foundations of Genetic Algorithms (pp. 69–93). California: Morgan Kaufmann.

Holland, J.H. (1992). Adaption in natural and artificial systems. Massachusetts: MIT Press.

Kraft, D.H., Petry, F.E., Buckles, B.P., & Sadasivan, T. (1995). Applying genetic algorithms to information retrieval systems via relevance feedback. In P. Bosc & J. Kacprzyk (Eds.), Fuzziness in database management systems (pp. 330–344), Germany: Physica-Verlag.

Kraft, D.H., Petry, F.E., Buckles, B.P., & Sadasivan, T. (1997). Genetic algorithms for query optimization in information retrieval: Relevance feedback. In E. Sanchez, T. Shibata, & L. Zadeh (Eds.), Advances in fuzzy systems: Applications and theory, vol.7 (pp. 155–173), Singapore: World Scientific.

Maes, P. (1994). Agents that reduce work and information overload. Communications of the ACM, 37, 30–40.

Maes, P. (1995). Modeling adaptive autonomous agents, In C.G. Langton (Ed.), Artificial Life (pp. 135–162), Massachusetts: MIT Press.

Mitchell, M., & Forrest, S. (1994). Genetic algorithms and artificial life. Artificial Life, 1, 267–289.

Rasmussen, E. (1992). Clustering algorithms. In W.B. Frakes & R. Baeza-Yates (Eds.), Information retrieval: Data structures and algorithms, Englewood Cliffs, NJ: Prentice Hall.

Riecken, D. Intelligent agents. Communications of the ACM, 37, 18–21.

Salton, G., & Buckley, C. (1990). Improving retrieval performance by relevance feedback. Journal of the American Society for Information Science, 41, 288–297.

Salton, G. & McGill, M.J. (1983). Introduction to modern information retrieval. New York: McGraw-Hill.

Selberg, E. & Etzioni, O. (1995). Multi-engine search and comparison using the MetaCrawler. Proceedings of the 4th World Wide Web Conference (pp. 195–208).

Spears, W.M. (1993). Crossover or Mutation? In L.D. Whitley (Ed.), Foundations of Genetic Algorithms 2 (pp. 221–237), California: Morgan Kaufmann.

Yager, R.R. (1978). Fuzzy decision making including unequal objectives. Fuzzy Sets and Systems, 1, 87–95.

Yager, R.R. (1996). Intelligent agents on the World Wide Web. Proceedings of the 1996 Workshop on Flexible Query-Answering Systems (FQAS'96) (pp. 289–306). Denmark: Roskilde University.

Zadeh, L.A. (1965). Fuzzy sets. Information and Control, 83, 338–353.