Genetic Approach to Query Space Exploration

M. Boughanem, C. Chrisment, L. Tamine
IRIT/SIG Campus Univ. Toulouse III,
118, Route de Narbonne
31062 Toulouse (France)
Fax : (33) 5 61 55 58 62
email: bougha@irit.fr

Abstract

This paper describes a genetic algorithm approach for intelligent information retrieval. The goal is to find an optimal set of documents which best matches the user's needs by exploring and exploiting the document space. More precisely, we propose a specific genetic algorithm for information retrieval based on knowledge based operators and guided by a heuristic for relevance multi-modality problem solving. Experiments with TREC-6 French data and queries show the effectiveness of our approach.

Keywords : Information retrieval, genetic algorithm, relevance feedback.

1 Introduction

The explosive growth of the Internet and other sources of information access have prompted the rapid proliferation of large information collections. Therefore, the effort required to retrieve relevant information has become significantly more difficult. Several investigations continue to deal with this problem [13]. These investigations are concerned with term weighting and the popular relevance feedback technique.

Relevance feedback is still the main technique for query modification and therefore a solution to search improving. This technique has been investigated for more than twenty years in various information retrieval models such, as the probabilistic model [22], [23],[9] and the vector space model [25]. In principle, relevance feedback is based on automatically changing the set of query terms as well as the weights associated to these terms according to some documents retrieved and judged during the initial search. However, even though relevance feedback improves significantly the performance of the search, no information retrieval process can be expected to recall all the possible relevant documents for all searches. Recently, there has been a growing interest in machine learning techniques, particularly the neural approach [19], [30], [31], [4] and the use of evolution algorithms that rely on analogies to natural processes. The genetic algorithm approach developed by Holland [16] and based on both the principles of natural evolution processes and Darwinian survival theory has proved its efficiency as a stochastic optimization methods. GA have been applied in information retrieval domain in the purpose of optimizing document descriptions [11] and improving query formulation [32] [6]. This is precisely the context of our work. Our goal is to build a GA that can find find an optimal set of documents which best match the user's need by exploring different regions of the document space simultaneously. We have been attracted by the use of GA to handle the information retrieval process for the following reasons :

- The document base represents a high dimensional space. As genetic algorithms have been shown to be powerful search mechanisms due to their robust nature and quick search capabilities, they seem to be suitable for information retrieval. Thanks to their ability to employ the inherent properties of implicit parallelism, GAs could perform the search in different regions of the document space simultaneously,

- Contrary to the classical retrieval models, the GA manipulates a population of queries rather than an individual query. Each query may retrieve a subset of relevant documents that can be merged. We believe that this is more efficient than using a hill-climbing search based on a single query,

- The classical methods of query expansion manipulate each term independently of each other. But several experiments have already shown that the terms occur in the documents by groups. The GA would contribute in this case to preserve useful information links representing a set of terms occurring in the relevant documents.

- The classical methods of relevance feedback are not efficient when no relevant documents are retrieved with the initial query. In contrast, the probabilistic exploration induced by the GA allows exploring new zones of the document space independently of the initial query.

As mentioned above some works applied the GA to information retrieval [11], [32], [17] and [6] either for document descriptors or query optimization. These are based on pure GA operators (called in this paper blind operators) and do not take advantage of the research in information retrieval. Whereas our model is characterized by its "knowledge enhanced operators" (knowledge only means using some known relevant information issued from the information retrieval domain) and the use of a niching technique for our GA.

The main aim of the work reported in this paper is to show the effectiveness of our GA model in improving search performance. This GA model is characterized by its knowledge-enhanced operators that explore several regions of the document space simultaneously. However, we must mention that we are not comparing our GA to other relevance feedback methods such as those based on Rocchio for example [24], [26]. Our goal is to show how we can do differently and how the GA could be used for this purpose. With this aim we first present in this paper a brief overview of genetic algorithms and the related work in IR. We then describe our GA model. Finally in the last section of this paper we present the experiments performed on the TREC-6 French database [13] and the results are discussed.

2 Genetic algorithm and information retrieval

Genetic algorithms are stochastic optimization methods developed by Holland [16] and based on both the principles of natural evolution processes and Darwinian survival theory. In such algorithms, a population of potential solutions is renewed at each generation by encouraging the reproduction of the fittest ones. After a number of generations, the program converges to the best individual representing the optimal solution.

2.1 Basic concepts of genetic algorithm

Individual and population : an individual, also called a chromosome,is the representation of a potential solution to the problem. It is implemented as a data structure where each component represents a *gene*. The value and position of a gene are respectively named *allele* and *locus*. All the individuals manipulated by the algorithm at one generation (one iteration of the GA) compose a population; this population evolves through the generations.

Adaptation function : this function evaluates the fitness of each individual

Genetic operators : some individuals of each generation undergo transformations by means of genetic operators to build new solutions that are hopefully more fit . Genetic algorithms apply three transformations: selection, crossover and the mutation.

- Selection: consists in the reproduction of a new population according to the probability distribution based on the fitness values. Individuals that are more fit have better chances to get more copies selected in the next generation.
- Crossover: is a recombination operator applied with a given probability P_c. Each pair of individuals selected for the crossover operation exchange part of their information delimited by a crossing point which is randomly selected along the chromosome.
- Mutation: it is an unary transformation operator applied with a given probability P_m. For each individual of the crossovered population and for each gene, we generate a random number r from the range of [0..1], if $(r < P_m)$ the gene is mutated.

Evolution : the resolution of an optimization problem using a genetic algorithm requires the modelisation of a generic solution and a fitness function. After the *initialization of the population*, follows *selection*, *crossover* and *mutation* operators in order to set up the next generation. The remaining evolutions are simply *cyclic repetitions of the above steps* until the program reaches a predetermined number of generations or converges. Several theoretic studies have already proved the effective convergence of genetic algorithms [16], [21]. Furthermore, these studies have suggested the use of heuristics to improve the control on the genetic exploration process. We note niching and speciation techniques [10] and knowledge based operators and adaptive control methods [10] [12] [27].

Main properties : the main properties of a genetic algorithm are the following:

- implicit parallelism: when manipulating an n size population, a genetic algorithm explores simultaneously a number of directions running to n^3. This expresses the fundamental properties of implicit parallelism demonstrated by Goldberg [10].
- resolution of exploration/exploitation dilemma: the genetic programming resolves efficiently the exploration/exploitation dilemma by allowing an exponentially increasing number of copies to the fitter individuals. Therefore, the exploration in good directions is encouraged.
- non-optimality: a genetic algorithm does not guarantee to reach the global optima. However, experiments showed that it generally returns near optimal solutions.

2.2 Related works in GA and IR

The development of scheme theory invented by Holland [15] and some theoretical studies in GA [1], have attracted scientists from several research areas. Some works and studies have been done in the IR area and we discuss few below:

Gordon [11] adopted GA to derive better descriptions to the documents. Each document is assigned N descriptions; each description is a set of indexing terms. Genetic operators and relevance judgement are applied to the descriptions in order to build the **best** document description. The author showed that the GA produces document descriptions better than the ones generated by the probabilistic model. Redescription improved the relative density of co-relevant documents by 39.74% after twenty generations and 56.61% after forty generations. Gordon exploited also these results and defined a classification method [11] based in clustering the relevant documents to a specific query.

Jing-Jye Yang and Robert R. Korfhage [32] proposed a GA to query optimization by reweighting the document term indexing without query expansion. They used a selection operator based on a stochastic samples, a blind crossover at two crossing points, and a classical mutation renews the population of queries. The experiments showed that the queries converge to their relevant documents after six generations.

Chen and Kim proposed a hybrid genetic and neural network based system called GANNET [6]. This system performs concept (keyword) optimization for user-selected documents using GA and uses the optimized concepts to perform concept exploration in a Hopfield net representing related concepts. The retrieving process is cyclic and is done in two stages. The first stage is the concept optimization, the GA manipulates input documents and their associated keywords to generate an initial set of optimized concepts. The second stage is the concept exploration, the set of optimized concepts is used by the hopfield net to produce other relevant concepts (new genes suggested by the nature) which are included in GA for the next concept optimization. This process is repeated until there is no further improvement.

Kraft and al. [17] apply a GA programming in order to improve the weighted boolean query formulation. The documents are viewed as a vector of index terms. A weighted boolean query is represented as chromosome in Koza's genetic model [18]. The goal of the GA is to modify the query in order to improve the search performance in term of recall and precision. Their first experiments showed that the GA programming is viable method of deriving good queries.

3 The GA model for document space exploration

The goal of our GA is to find an optimal set of documents which best matched the user's needs. The genetic algorithm attempts to involve, generation by generation, a population of queries towards those improving the outcomes of the system. The main characteristics of our model are its knowledge-based operators instead of the blind operators used on the previous cited works and its niching technique.

3.1 Query individual and query population

The genetic individual in our approach is a query. Each gene or chromosome represents a certain term or concept. The *locus* (the existence or the absence of certain gene) is represented by a real value and defines the importance of the term in the considered query. Each individual representing a query is under the form:

$$Q_u(q_{u1}, q_{u2}, ..., q_{uT})$$

Initially, a term weight can be computed by any query term weight scheme; it will then evolve through the generations. In our case, we used the following formula [28]

$$q_{ui} = \frac{(1 + log(tf_{ui})) * log(\frac{N}{n_i})}{\sqrt{\sum_{k=1}^{T}((1 + log(tf_{uk})) * log(\frac{N}{n_k}))^2)}}$$

Notations used in this article:

T : *total number of stemmed terms automatically extracted from the documents*
N : *total number of documents*
t_i : *ith term*
n_i : *number of documents containing term* t_i
d_j : *jth document*
tf_{ji} : *frequency of* t_i *in* d_j
d_{ji} : *term weight of* t_i *in* d_j
$Q_u^{(t)}$: *query individual u at the generation (t) of the GA*
q_{ui} : *weight (locus) of the* $term_i$ *(gene) in* $Q_u^{(t)}$
$pop^{(t)}$: *population of individuals at the generation (t) of the GA*
Pop_size : *size of the population*
$D_r^{(t)}$: *set of relevant documents retrieved by the* $pop^{(t)}$
$D_{nr}^{(t)}$: *set of nonrelevant documents retrieved by the* $pop^{(t)}$
D_r : *set of relevant documents retrieved through all generations*
D_{nr} : *set of nonrelevant documents retrieved through all generations*
Sim : *Jaccard measure*
$Eucl\_Distance$: *Euclidean distance*

A population of individuals is a set of queries. The initial population, ($Pop^{(0)}$), contains the initial query and a list of relevant documents retrieved by this initial query. However, if the population size is not reached some top nonrelevant documents are also listed to the initial population. Thus the initial population is not randomly constructed. Using this method, the genetic algorithm will begin the exploration of the document space "good" directions. This population is renewed after each iteration of the GA.

3.2 Adaptation function

An importance value called a *fitness* is assigned to each query in the population. It represents the effectiveness of a query during the retrieving stage. Conventionally, the aim of the relevance feedback process is to move the query in the direction forwards relevant documents and away from nonrelevant documents. Thus, we proposed the following formula for the adaptation function:

$$QFitness(Q_u^{(t)}) = \frac{\frac{1}{\|D_r\|} * \sum_{d_j \in D_r} Sim(d_j, Q_u^{(t)})}{\frac{1}{\|D_{nr}\|} * \sum_{d_j \in D_{nr}} Sim(d_j, Q_u^{(t)})}$$

However, the "relevance" function is multi-modal in the sense that relevant documents corresponding to the same information need may be located at different regions of the document space and therefore have some different vector components. But, the classic genetic algorithm evolves moving the query population toward an optimal query, which retrieves documents of similar representation. For this reason, we adjust the adaptation function in order to encourage the construction of queries with comparable performances. Using the techniques of speciation in ecological theory [10], we define a **niche** as a set of queries that retrieve documents of similar genotype. We propose the following definition of the sharing function:

$$Niche(Q_u^{(t)}) = \{Q_v^{(t)} \quad Eucl\_Distance(Q_u^{(t)}, Q_v^{(t)}) \leq niching\_threshold\}$$

Here, $Eucl\_Distance(Q_u^{(t)}, Q_v^{(t)})$ is a normalized euclidian distance computed as follows $Eucl\_Distance(Q_u^{(t)}, Q_v^{(t)}) = \frac{sqrt(\sum(q_{ui}-q_{vi})^2)}{max(\|Q_u^{(t)}\|,\|Q_v^{(t)})\|}$. The normalisation takes into account the number of genes different from zero. The $niching\_threshold$ is determined experimentally. The adaptation function becomes:

$$Fitness(Q_u^{(t)}) = \frac{QFitness(Q_u^{(t)})}{\|Niche(Q_u^{(t)})\|}$$

As a result the query fitness will be proportional to the similarity measure with relevant documents and inversely proportional to both similarity measure with nonrelevant documents and number of queries in the same niche. This would encourage the selection and reproduction in less niche query populated. Therefore, new regions will be explored in agreement with the average fitness of the associated queries.

So, the query population is represented as follows:

$$Pop^{(t)} = \{(Q_u^{(t)}, Fitness(Q_u^{(t)})) \quad u = 1..Pop\_size\}$$

3.3 Genetic operators

A considerable number of techniques have been developed in order to increase the performances of a retrieval information process. In the precise context of query expansion, experiments showed that some parameters have a major effect on retrieval performance. These include the term selection method, the number of terms to add to the query, the term reweighting method [14]. We use these results in order to handle enhanced knowledge operators that improve the genetic exploration in the document space.

3.3.1 Selection

Our selection procedure is based on roulette wheel selection [10]. It consists essentially of assigning to every individual of the population a number of copies in the next generation, proportional to its relative fitness. More precisely, the roulette wheel slots are sized according to the fitness of the individuals. Each individual has a certain number of slots proportional to its fitness value and the ones spins the wheel and selects a single individual each time.

3.3.2 Crossover

The goal of the crossover operators we defined is to use the distribution of both grouped and independent indexing terms in the relevant documents. Furthermore, the genetic algorithm will apply the individual queries according to their niche in order to continue the exploration in different regions of the document space. We are thus aiming to reach relevant documents with different representations. We define two crossover operators: a crossover based on term weight in the queries which will be used for the queries in the same niche; and the crossover based on term co-occurrence in the relevant documents which will be used for the queries belonging to different niches.

Crossover based on term weight

This crossover does not use a crossing point, it allows us to modify the term weights according to their distribution in the relevant and nonrelevant documents. Let us consider $Q_u^{(t)}$ and $Q_v^{(t)}$ two individuals (queries) selected for crossover. The result is the new individual $Q_p^{(t)}$ defined as follows :

$$\left.\begin{array}{l} Q_u^{(t)}(q_{u1}, q_{u2}, ..., q_{uT}) \\ Q_v^{(t)}(q_{v1}, q_{v2}, ..., q_{vT}) \end{array}\right\} \longrightarrow Q_p^{(t+1)}(q_{p1}, q_{p2}, ..., q_{pT})$$

$$q_{pi} = \begin{cases} Max(q_{ui}, q_{vi}) \ if \ weight(t_i, D_r^{(t)}) \geq weight(t_i D_{nr}^{(t)}) \\ Min(q_{ui}, q_{vi}) \ otherwise \end{cases}$$

We defined the weight of term t_i in a set of documents D as follows :

$$weight(t_i, D) = \sum_{d_j \in D} d_{ji}$$

In words, if the weight of t_i term in the set of relevant documents is higher than its weight in the set of nonrelevant documents, this term is retained as significant and the highest weight among (q_{ui}, q_{vi}) is assigned to this term in the new query $Q_p^{(t+1)}$. Otherwise, the lowest weight is assigned to it in the new query. This operator will be applied on queries of the same niche. Indeed, the goal of this operator is to adjust the descriptions of the new queries in order to recall further relevant documents in the same region of the document space.

Crossover based on term co-occurrence

Let us consider $Q_u^{(t)}$ and $Q_v^{(t)}$ two individuals selected for crossover with k as the crossing point and $Q_u^{(t+1)}$ and $Q_v^{(t+1)}$ as the new individuals resulting from the crossover. In the classical GA, the crossover operator, called a blind operator in our experiment, consists of exchanging a part of the genes between the selected individuals. Our second crossover operator uses the same idea, but based on the term co-occurrence in the relevant documents. Thus, the new individuals copy the first k terms from their parents and the remaining terms are built as follows: each term among the noncopied terms of one parent, is replaced by the most co-occurent term in the other parent. Formally, this operator is defined as :

$$\left.\begin{array}{l} Q_u^{(t)}(q_{u1}, q_{u2}, ..., q_{uT}) \\ Q_v^{(t)}(q_{v1}, q_{v2}, ..., q_{vT}) \end{array}\right\} \longrightarrow \begin{array}{l} Q_u^{(t+1)}(q'_{u1}, q'_{u2}, ..., q'_{uT}) \\ Q_v^{(t+1)}(q'_{v1}, q'_{v2}, ..., q'_{vT}) \end{array}$$

where

$$
\begin{array}{l}
q'_{ui} = q_{ui} \quad if \ (i \leq k) \\
othewise \ for \ i = k+1 \ to \ T \\
\qquad lookup \ t_l \ in \ Q_v^{(t)} \ / \ l > k \ \wedge \ max^D(C_{t_i,t_l}, Q_v^{(t)}) \\
\qquad q'_{ul} = q_{vl} \\
\qquad endfor
\end{array}
$$

$Q_v^{(t+1)}$ is built in same way, just replacing u by v.

Here, $max^D(C_{t_i,t_l}, Q_v^{(t)})$ returns term t_l of $Q_v^{(t)}$ that has the highest co-occurrence weight with t_i in the set of documents D.

The co-occurrence of two terms in a set of documents is defined as follows :

$$C_{t_i,t_l} = \frac{\sum_{d_j \in D_r} d_{ji} * d_{jl}}{\sum_{d_j \in D_r} d_{ji}^2 + \sum_{d_j \in D_r} d_{jl}^2 - \sum_{d_j \in D_r} d_{ji} * d_{jl}}$$

This operator will be applied to the queries belonging to different niches. It allows for the exchange of a part of information between two queries retrieving relevant documents located at different region of the document space. This would enable exploration of a new region in the document space. Notice that a variant of this operator have been defined in our works. One consists of copying all the genes different from zero from the parent to the new individual and then for each gene, from (k+1) to (T), lookup its most co-occurent term in the other parent and add it to the new individual [3].

blind crossover

The blind crossover is based on the classical GA crossover operator. Let us consider $Q_u^{(t)}$ and $Q_v^{(t)}$ two individuals selected for crossover with k as the crossing point and $Q_u^{(t+1)}$ and $Q_v^{(t+1)}$ as the new individuals resulting from the blind crossover. This operator is defined as follows :

$$\left.\begin{array}{l} Q_u^{(t)}(q_{u1}, q_{u2}, q_{uk}, q_{uk+1} \cdots, q_{uT}) \\ Q_v^{(t)}(q_{v1}, q_{v2}, q_{vk}, q_{vk+1} \cdots, q_{vT}) \end{array}\right\} \longrightarrow \begin{array}{l} Q_u^{(t+1)}(q_{u1}, q_{u2}, .., q_{uk}, q_{vk+1}, \cdots, q_{vT}) \\ Q_v^{(t+1)}(q_{v1}, q_{v2}, .., q_{vk}, q_{uk+1}, \cdots, q_{uT}) \end{array}$$

3.3.3 Mutation

Mutation based on term relevance

It consists essentially of exploring the terms occurring in the relevant documents in order to adjust the corresponding gene values in the query individual selected for the mutation. The mutation operator we defined is based on **term relevance**, this mutation is noted *Mutation1* .

Let us consider $Q_u^{(t)}$ as the selected individual query and $L^{(t)}$ as the set of terms from $D_r^{(t)}$, the relevant documents retrieved in the last generation of the GA. The mutation will alter the genes of the selected individual on the basis on the $L^{(t)}$ terms and on the probability P_m. The $L^{(t)}$ terms are sorted according to a score value calculated as follows:

$$Score(t_i) = \frac{\sum_{d_j \in D_r^{(t)}} d_{ji}}{\|D_r^{(t)}\|}$$

This score is used in case of limiting the number of terms that can be used for mutation process. Previous experiments have been performed using this limit [3]. The mutation operation is done as follows :

for each term t_i *in* $L^{(t)}$
if (random(p) $< P_m$*) then* /* mutate the gene t_i
$q_{ui} = average(Q_i^{(t)}) - \delta$ /* modify the weight of t_i in $Q^{(t)_u}$
endif
endfor

Here, $random(p)$ generates a random number p from the range of [0..1]. The average function is computed as follows : $average(Q_u^{(t)}) = \frac{\sum_j^T q_{ui}}{n_{q_{ui}}}$, where $n_{q_{ui}}$ is number of $q_{ui} \neq 0$ in $Q_u^{(t)}$. δ is a parameter used to control the average value, the value used yet is $\delta = 0$.

Blind mutation

In blind mutation the genes are mutated by modifying their weight arbitrary. Let us consider $Q_u^{(t)}$ as the selected individual for mutation. This operation is performed as follows :

for each t_i *in* $Q_u^{(t)}$
if (random(p) $< P_m$*) then* /* mutate
$q_{ui} = arbitrary(w)$ /* modify the weight of t_i in $Q^{(t)_u}$
endif
endfor

Here, $arbitrary(w)$ returns a real w in range of]0..1].

4 Experiments and results

4.1 Experiments

The experiments were carried out on TREC-6 French data and 21 queries. The documents are from the SDA News (Schweizerischen Depeschenagentur-Swiss News Agency- 141 646 documents). They were run using the Mercure information retrieval system [2], as used in different TREC conferences [13]. The main goal of these experiments is to evaluate the effectiveness of our GA for IR. More precisely, at first we measure the effects of P_c and P_m probabilities and the population size, and then we compare the knowledge based operators versus the blind operators. We also evaluate the effect of our niching technique.

Before explaining the experimentation, first we notice that the query evaluation, corresponding to do search in the algorithm, is done using the spreading activation technique developed in Mercure system [4]. This process enable to rank the retrieved documents. Second, concerning the number of judged documents, fifteen (15) in this experiment, this number has been chosen arbitrary knowing that a value in the range of [10..15] is commonly used in most works performing manual relevance feedback [23],[14], [26], [4]. However experiments with varying this number would be done in future work. Thus, the general experimentation is done as follows :

1- Submit the initial query and do the search

2- Judge the top fifteen documents

3- build the initial population as follows

3.1- some top judged relevant (or nonrelevant) documents join the initial query in the set of initial individuals

3.2- compute the fitness of this individuals

3.3- apply the genetic operators to these individuals as follows :

Repeat

- parent1 = wheel-selection($Pop^{(t)}$) ; parent2=wheel-selection($Pop^{(t)}$)

- Crossover(P_c, parent1, parent2, son1, son2)

- Mutation(P_m; son1, sonmut) ; Add population(sonmut,$Pop^{(t+1)}$)

- Mutation(P_m, son2, sonmut); Add population(sonmut,$Pop^{(t+1)}$)

until Size($Pop^{(t+1)}$)=$fixed$ population size. the population is then built

4- For each query in the population do a search

5- Build a merged list of documents

6- Judge the top 15 documents

7- Compute the fitness of each query

8- Apply the genetic operators; as in 3.3

9- Repeat 4, 5, 6 7, 8 steps until a fixed number of iteration

The steps 4, 5, 6, 7 and 8 are repeated until a fixed number of iterations. We have fixed this number to 5. At each iteration of the GA (corresponding to a new population and called a new generation of the GA), the system presents to the user a list of new documents, the user sees the top 15 and selects the relevant ones, the GA is then applied. 15 new more documents are retrieved and shown to the user and so on. The goal of our GA is to recall new relevant documents at each iteration of the search. Thus, in order to evaluate the different points outlined above, and with respect to the genetic approach we have defined three basic groups of operators. Each group is composed by a specific crossover operator and a mutation operator in Table 1.

Notice that the first experiments have been performed using the G1 and G2 operators

G1	Crossover1: based on term weight	Mutation1: based on term relevance
G2	Crossover2: based on term co-occurrence	Mutation1: based on term relevance
Blind	blind crossover	blind mutation.

Table 1: genetic operators

in order to measure the parameters of the GA (P_c, P_m, Pop_size). But the main operator we have defined is based on the niching technique and it will evaluated in the section 4.3.3.

4.2 Evaluation method

The main evaluations presented in this paper measure the ability of our GA to increase the precision in the top fifteen documents retrieved at each iteration of the GA. So, because of the multiple iteration aspect of the search and the use of relevance judgement, the results reported in the paper are based on a residual ranking evaluation [5]. This method is used to evaluate the effectiveness of relevance feedback methods. In this method all the documents previously judged are removed from the document rankings produced by both the initial query, which corresponds to the iteration 0 in our algorithm, and the feedback query, which corresponds to iteration 1 in our algorithm. Precision and recall are computed and then for both residual list of documents. In the case of multiple iteration the comparison is done is the same way between the residual document retrieved at iteration i to residual document retrieved at iteration (i+1). This tells us how much we gained by doing the next iteration of the GA for more details, readers can refer to [14].

In our context we compare the effect of using our GA to not using a GA. Let us consider that the iteration(i+1) of the GA is performed. In order to measure the effectiveness of the GA in this iteration, we compared the 15 top retrieved documents at that iteration to the no GA documents, which corresponds to the list resulting from just keeping the next 15 documents at iteration(i). We limited the number of documents to 15 because our main goal is to improve the precision at that number.

4.3 Results and discussions

4.3.1 Effects of the GA parameters : P_c, P_m, Pop_size

The effectiveness of the defined GA depends at first on a list of important genetic parameters such a crossover probability P_c, the mutation probability P_m and the population size.

Effect of the P_c and P_m probabilities

The first experiment has been performed by varying P_c and P_m, $P_c \in \{.1, .25, .5, .7\}$ and $P_m \in \{.01, .07, .1, .25, .5\}$ for a fixed $Pop\_size = 5$ (the extreme values 0 and 1 are not used in the GA, we respected this convention). The GA is performed by considering the G1 group only. The table 2 shows, for different P_c and P_m values, the number of relevant documents retrieved in a given iteration (totalled across all queries used at that iteration) and the cumulative total of relevant documents retrieved by that point. The columns numbered form 1 to 5 represent the iteration number, the iteration 0 corresponding to the initial search is not represented, because the results of the use of the GA begin at iteration 1.

	Pc=.1					Pc=.25				
	Nb. of relev. doc. in the 15 top retr. at given iteration (over 21 queries)									
Pm	1	2	3	4	5	1	2	3	4	5
.01	91	36	43	53	36	89	60	43	36	41
.07	91	36	43	53	37	89	60	43	36	40
.1	91	36	43	53	37	89	60	43	36	40
.25	91	36	44	53	35	89	60	43	36	40
.5	91	37	47	49	36	89	60	43	38	38
	Cumul. nb of relev. doc. in the 15 top retr. at given iteration (over 21 queries)									
.01	91	127	170	223	259	89	149	192	228	269
.07	91	127	170	223	260	89	149	192	228	268
.1	91	127	170	223	260	89	149	192	228	268
.25	91	127	171	224	259	89	149	192	228	268
.5	91	128	175	224	260	89	149	192	230	268
	Pc=.5					Pc=.7				
	Nb. of relev. doc. in the 15 top retr. at given iteration (over 21 queries)									
Pm	1	2	3	4	5	1	2	3	4	5
.01	92	49	49	48	37	94	54	69	39	36
.07	92	49	49	48	36	94	54	69	39	36
.1	92	49	49	48	36	94	54	69	39	36
.25	92	49	49	48	37	94	54	69	39	34
.5	92	49	47	46	34	94	54	68	40	34
	Cumul. nb. of relev. doc. in the 15 top retr. at given iteration (over 21 queries)									
.01	92	141	190	238	275	94	148	217	256	292
.07	92	141	190	238	274	94	148	217	256	292
.1	92	141	190	238	274	94	148	217	256	292
.25	92	141	190	238	275	94	148	217	256	290
.5	92	141	188	234	266	94	148	216	256	290

Table 2: *Effect of the P_c and P_m parameters*

So we can easily notice as expected that P_c and P_m play an important role in the effectiveness of the GA. The table shows that the P_c plays more role than P_m, which is in agreement with the GA literature [8]. Indeed, it can be seen that the cumulative number of relevant retrieved documents at given iteration and P_c is nearly the same for varying values of P_m. The cumulative number of relevant retrieved documents at each iteration increases, and it goes from 259 for $P_c = .1$ to 292 for $P_c = .7$.

According to these results, some question can be asked. The main question concerns the real contribution of the mutation operator. It is well known in the GA literature ([8], [10]) that the mutation operator has a very small effect in the GA and it is often used with very small probability, around 0.005. But *is it worth using this operator in our algorithm?* We think that even though the effect of such as operator is very small, it is still worth of use. First, we want to be in agreement with the general GA approach. Second, even though the crossover operator efficiently recombine the existing genes (terms) it possible that the crossover operator loses some important terms. So, mutation will normally protect us from this problem because the mutation operator allows us to add some important terms extracted from the relevant judged documents.So for the remain experiments we have chosen $P_c = .7$ and $P_m = 0.07$.

Effect of the population size

The size of the population is a determinant factor, indeed a large population size induces noise and a high query run-time, whereas a small size induces the silence because the genetic transformations are done in the same region of the document space.

	G1				
	iter1	iter2	iter3	iter4	iter5
	Population size = 2				
	no GA				
rel. retr. at iteration	90	48	33	46	24
cum. rel. retr.	90	138	171	217	241
	with GA				
rel. retr. at iteration	72	52	36	46	24
cum. rel. retr.	72	124	160	206	230
	Population size = 4				
	no GA				
rel. retr. at iteration	90	57	50	42	42
cum. rel. retr.	90	147	197	239	281
	with GA				
rel. retr. at iteration	96	64	40	45	42
cum. rel. retr.	96	160	200	245	287
	Population size = 6				
	no GA				
rel. retr. at iteration	90	57	39	44	37
cum. rel. retr.	90	147	186	230	267
	with GA				
rel. retr. at iteration	96	64	55	51	30
cum. rel. retr.	96	160	215	266	296
	Population size = 8				
	no GA				
rel. retr. at iteration	90	64	42	37	42
cum. rel. retr.	90	154	196	233	275
	with GA				
rel. retr. at iteration	96	72	50	38	29
cum. rel. retr.	96	168	218	256	285

G2				
iter1	iter2	iter3	iter4	iter5
Population size = 2				
no GA				
90	59	36	40	21
90	149	185	195	216
with GA				
72	42	42	35	21
72	112	154	189	210
Population size = 4				
no GA				
90	54	46	40	41
90	144	190	230	273
with GA				
85	40	46	58	45
85	125	171	229	274
Population size = 6				
baseline, no GA				
90	64	40	31	31
90	154	194	225	256
with GA				
85	64	41	51	37
85	149	190	241	278
Population size = 8				
no GA				
90	64	36	40	30
90	154	190	230	260
with				
85	64	45	36	30
85	159	204	240	270

Table 3: *Results according the population size through the generations*

The table 3 compares the results using no GA and using our GA for various population size and by considering the $G1$ and $G2$ operators. The table lists the number of relevant document retrieved in a given (totalled across all queries used at that iteration) and the cumulative total number of relevant documents retrieved by that point.

We notice that when using our GA on both G1 and G2, the total number of relevant documents after 5 iterations is much higher than using no GA for all the population sizes greater than 2. It can be also seen that for the population size greater than 2 when using G1 group, the number of relevant document is much higher since the first iteration and still higher through the iterations than using no GA. Whereas for G2, this number begins to increase after the third iteration. Furthermore, the table shows that G1 is more effective and recall more relevant document than G2.

The main conclusion that can be drawn from this experiment is that the basic operators we defined improve the result of the search for a population size more than 2. In order to make a good compromise between run-query time and noise, the population of size 6 seems more significant in both G1 and G2. We retained this value for our algorithm for the remaining experiments.

4.3.2 Effect of the knowledge based operators versus blind operators

Table (4) compares the results of the GA using the blind operators and the knowledge based operators (population size = 6). The table gives the number of relevant document in the top 15 retrieved at each iteration of the GA, and the cumulative total number at that point.

	iter1	iter2	iter3	iter4	iter5
	G1				
rel. retr. at iteration	96	64	55	51	30
cum. rel. retr.	96	160	215	266	296
	G2				
rel. retr. at iteration	85	64	41	51	37
cum. rel. retr.	85	149	190	241	278
	blind				
	no GA				
rel. retr. at iteration	90	23	23	32	22
cum. rel. retr.	90	113	136	168	190
	with GA				
rel. retr. at iteration	27	28	41	28	24
cum. rel. retr.	27	55	96	126	150

Table 4: *Results for knowledge based operator vs. blind operators*

We clearly notice that the knowledge-based operators are more effective than the blind ones. Indeed the cumulative total number of relevant documents after 5 iterations is almost twice as high using G1 and G2.

It can be also seen that the blind operators have a negative effect in the first iteration of the search. However, we notice that the initial search (the search with the initial query without any iteration for the GA) gives a good result. although the table shows that after the first iteration the number of relevant retrieved documents increases through the generation of the GA. So, this is in agreement with the GA literature. Here it is mentioned that the convergence of the algorithm is observed after many iterations of the GA, at least more than 20. But in our context, a large number of the iterations is not conceivable except if this algorithm is used to build a profile in a filtering process, for example [13].

4.3.3 Effect of the niching technique : *Niching or not niching?*

We recall that the main idea of our GA is the use of the niching technique. Thus, the crossover operators have been defined in order to take into account this notion. In order to evaluate the niching technique we build a combination of genetic operators by considering specific crossover according to niching. Table 5 shows this combination.

niching operator	same niche	different niche
G3	Crossover1, Mutation1	crossover2, Mutation1

Table 5: niching operator

This operator is applied as follows: when two queries are selected for transformations, if they belong to the same niche, crossover1 is applied to these queries and then mutation1 to the new query (result of the crossover1). If they are in different niches, crossover2 is applied to these queries and then mutation1 to each new query resulting from the crossover2.

To be effective the niching technique needs suitable a $niching\_threshold$, to decide whether two queries belong to the same niche or not. The first experiments is performed by varying the $niching\_threshold \in \{0, .1, .25, .5, 1\}$.

Table 6 shows that the threshold plays an important role in the niching operator. It can be seen that a good $niching\_threshold$ is in around .1-.2, knowing that a $niching\_threshold =$ 0 corresponds to the use of G2 only with all the queries belong to different niches, while $niching\_threshold = 1$ corresponds to the use of G1 with all the queries are in the same niche.

Concerning the effectiveness of the niching technique, Table 7 shows the comparison between the G1 and G2 (operators without niching) and G3 (with the niching technique using $niching\_threshold = .1$).

	iter1	iter2	iter3	iter4	iter5	
Niching threshold	nb. relev. doc. in the 15 top retrieved					total
0	85	64	41	51	37	278
.1	105	56	51	51	41	304
.25	94	60	51	50	37	300
.5	94	60	48	50	42	294
1	96	64	55	51	30	296

Table 6: Results of the niching according to thresholds

	iter1	iter2	iter3	iter4	iter5
	G1				
rel. retr. at iteration	96	64	55	51	32
cum. rel. retr.	96	160	215	266	298
	G2				
rel. retr. at iteration	85	64	41	51	37
cum. rel. retr.	85	149	190	241	278
	G3				
rel. retr. at iteration	105	56	51	51	41
cum. rel. retr.	105	161	212	263	304

Table 7: *Effect of the niching comparing to the basic operator*

It can be easily seen that *G3* is more efficient than its components taken separately (i.e., *G1* and *G2*). We notice that beyond the first iteration, the number of relevant documents in G3 is much higher than G1 and G2.

	iter1	iter2	iter3	iter4	iter5
	G3				
	no GA				
rel. retr. at iteration	90	48	46	44	39
cum. rel. retr.	90	138	184	228	267
	with GA				
rel. retr. at iteration	105	56	51	51	41
cum. rel. retr.	105	161	212	263	304

Table 8: *Effectiveness of the niching*

Table 8 compares the results using GA (G3) and using no GA. We notice that when using G3, the number of relevant documents retrieved at each iteration is much higher than using no GA. Therefore the cumulative total number of relevant documents using G3 is still higher through all the iterations than not using G3.

To summarize, it is clear from these tables that all the genetic operators we defined improve the performance of the search whether the niching is used or not. It seems that even though the differences between the results are not very great, the niching technique is effective in our GA. But, these results might be improved by using more suitable combinations between the operators, probably by revising the niching formula.

5 Conclusion

The experiments reported here constitute our first investigation in GA and IR. The results presented here demonstrate the effectiveness of our GA approach to improve the performance of an information retrieval system. We mainly showed that:

- manipulate a population of queries,
- use the probabilistic exploration of different zones of the document space,
- use acknowledge based operator instead of blind operators used in the classical GA,
- integrate the niching technique to recall relevant documents with different descriptions.

However, the results produced by the GA depend on the values of many parameters, such as the different probabilities, the population size and the different thresholds. This first experiment in TREC French data, allowed us to tune these parameters, and particularly to show another way of adapting the genetic approach to IR. We have shown that our GA for IR improves the results of the search in the interactive environment, even though the random aspect of the GA is still a big question. We also have to say that we are not comparing our method to other relevance feedback method. Our goals are to show how we can do differently and how the GA could be used for IR. However, this first investigation allowed us to be optimistic for future work in this direction, particularly when the documents are heterogeneous. In this case an IRS needs to use several queries, each one with specific representation to be efficient.

Our next goal is to use our approach on the English TREC data [13], in some TREC tasks (ad hoc and the interactive track) with more data and more queries. Then a global comparison for example between the size of the data and the GA parameters could be performed.

References

[1] Ankenbrandt, C., "An Extension to the Theory of Convergence and a Proof of the Time Complexity of Genetic Algorithms," FOGA90, pp 53-58, 1990.

[2] Boughanem, M., Soule-Dupuy, C., "Mercure at trec6," 6th International Conference on Text REtrieval TREC6, In Harman D.K. (Ed.), NIST SP, November 21-23, 1997.

[3] Boughanem, M., Tamine, L., "Reformulation de requêtes dans les SRI : une approche basée sur la génétique," Research report MSI-98-01, Jan. 1998.

[4] Boughanem, M., Soule-Dupuy, C., "Query modification based on relevance backpropagation," Proceedings of the 5th International Conference on computer-assisted information searching on Internet (RIAO'97), Montreal, June 1997.

[5] Chang, Y. K., Cirillo, G. C., Razon, J. "Evaluation of feedback retrieval using modified freezing, residual collection and test and control groups," chapter 17, pp. 355-370. Prentice-Hall Inc., 1971. In The Smart Retrieval System : Experiments in Automatic Document Processing.

[6] Chen, H., "Machine Learning for Information Retrieval: Neural Networks, Symbolic Learning and Genetic Algorithms," JASIS 46(3):194-216, April 1995.

[7] Daniels, P.,J., "The user modelling function of an intelligent interface for document retrieval systems," In B.C Brookes (Ed.), Intelligent information systems for the information society, Elsevier science publishers B.V.; North Holland, Amsterdam, 1986.

[8] Davis, L., "Handbook of genetic algorithms," Van Nostram Reinhold, New York, 1991.

[9] Haines, D., & Croft, W., B., "Relevance Feedback and Inference Networks," ACM SIGIR International Conference on Research and Development in Information Retrieval, 1993.

[10] Goldberg, D. E., "Algorithmes génétiques. Exploration, optimisation et apprentissage automatique," Edition Addison-Wesley France Juin 1994.

[11] Gordon, M., "Probabilistic and genetic algorithms for document retrieval," Communications of the ACM pp 1208-1218, October 1988.

[12] Grefenstette, J. J., "Virtual genetic algorithms: First results," Technical report AIC-95-013 , Navy Center for Applied Research in Artificial Intelligence, February 1995.

[13] Harman, D., "TREC overview," 6th International Conference on Text REtrieval TREC6, In Harman D.K. (Ed.), NIST SP, November 21-23, 1997.

[14] Harman, D., "Relevance feedback revisited," 15 th Annual ACM SIGIR Conference On Research and Development in Information Retrieval Copenhague, Denmark 1992.

[15] Holland, J., "Les algorithmes gënëtiques," Revue POUR LA SCIENCE 179 pp 44-51, Septembre 1992.

[16] Holland, J. , "Adaptation in natural and artificial systems," University of Michigan press, Ann Arbor, 1975.

[17] Kraft, D., H., Petry, F. E., Buckles, B. P., and Sadisavan, T., " Applying Genetic Algorithms to Information Retrieval Systems Via Relevance Feedback," In Bosc, and Kacprzyk, J. (eds.), _Fuzziness in Database Managment Systems_, Studies in Fuzziness Series, Heidelberg, Germany, Physica-Verlag, 1995, pp. 330-344.

[18] Koza, J. R., "A Hierarchical Approch to Learning the Boolean Multiplexer Function," In Rawlins, G.(ed), Foundations of Genetic Algorithms, San Mateo, CA:Morgan Kaufman, 1991, pp.171-192

[19] Kwok, K. L., " A neural network for probabilistic information retrieval," In proceedings of the twelfth annual international ACM/SIGIR Conference on Research and Development in Information Retrieval, pages 21-30, Cambridge, MA, June 25-28, 1989.

[20] Monarch, I., Carboll, J.,G., "CoalSORT: acknowledge-based interface," IEEE EXPERT, pages 39-53, Spring 1987.

[21] Radcliffe, N. J., "Equivalence class analysis of genetic algorithms,' Complex systems, 5:183-20, 1991.

[22] Robertson, S., & Sparck Jones, K., " Relevance weighting of search terms," Journal of the American Society for Information Science, 27:129-146 1976.

[23] Robertson, S., & Walker, S., "On relevance weights with little relevance information,", ACM/SIGIR International Conference on Research and development in Information Retrieval,1997.

[24] Rocchio, J. J., "Relevance feedback in information retrieval," In The smart system experiments in automatic document processing, G.Salton, editor, Prentice-Hall, Inc., Englewood Cliffs, NJ, 313-23.

[25] Salton, G., "The SMART Retrieval System," Englewood Cliffs, N. J. : Prentice-Hall, Inc, 1970.

[26] Salton, G., Buckley, C., "Improving retrieval performance by relevance feedbak," Journal of the American society for Information science 41(4): 288-297, 1990.

[27] Sebag, M., Schoenauer, M., "Contrôle d'un algorithme genetique," Revue d'intelligence artificielle, Volume n- 2-3, pages 389-428, 1996.

[28] Singhal, A., Buckley, C., Mitra M. "Pivoted document length normalisation," Conference on Research and Development in Information Retrieval (SIGIR) pp 21-29, 1996.

[29] Swartout, W., "Explanation and the role of the user model: how much will it help ?," In User modelling panel, Proceedings of the Ninth International Joint Conference on Artificial Intelligence, pages 1298- 1302, Los Angeles, CA, August 1985.

[30] Wilkinson, R., Hingston, P., "Using the Cosine Measure in a Neural Network for Document Retrieval," ACM/SIGIR International Conference on Research and development in Information Retrieval, 1991.

[31] Wong, S. K. M., Cai, Y. J., Yao, Y. Y., "Computation of term associations by a neural network," Conference on Research and Development in Information Retrieval (SIGIR) pp 107-115, 1993.

[32] Yang, J., J., Korfhage, R., "Query optimization in information retrieval using genetic algorithms", ICGA ' 93.