ARTICLE IN PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Computers & Operations Research ▮▮▮ (▮▮▮) ▮▮▮–▮▮▮

computers & operations research

www.elsevier.com/locate/cor

Improving metaheuristics convergence properties in inductive query by example using two strategies for reducing the search space

Sancho Salcedo-Sanz[a,*], Jing Su[b]

[a] *Sancho Salcedo-Sanz, Departamento de Teoría de la Señal y Comunicaciones,Universidad de Alcalá, Campus Universitario Carretera Madrid-Barcelona, km. 33, 28871 Alcalá de Henares, Madrid, Spain*

[b] *School of Computer Science, The University of Birmingham, B15 2TT, Edgbaston, Birmingham, UK*

Abstract

In this paper we present two strategies for reducing the time of convergence of two metaheuristics (genetic algorithms and simulated annealing) in inductive query by example (IQBE), which is a process for assisting the users of a given information retrieval system in the formulation of queries. Both strategies are based on a reduction of the search space size of the metaheuristics. The first strategy that we introduce is a compression–expansion strategy, where the terms are arranged into sets of a given size. The second strategy we consider is the so-called restricted search, where the number of terms in the genetic algorithm and simulated annealing encodings is fixed to be a given number *m*. We describe the implementation of the strategies and analyze when they can be successful, and the main drawbacks associated with them.

Keywords: IQBE; Genetic algorithms; Simulated annealing; Convergence; Heuristics

1. Introduction

In recent years, the study of emergent and nature-based algorithms and their application to information retrieval problems has been massive [1–4]. The majority of the information retrieval applications tackled

* Corresponding author. Tel.: +34 91 885 6731; fax: +34 91 624 8749.
E-mail address: sancho.salcedo@uah.es (S. Salcedo-Sanz).

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.05.001

with evolutionary-type approaches are related to relevance feedback (RF) or to inductive query by example (IQBE) applications [3]. RF is a method for query optimization which consists of using the information provided by the user after he or she has seen a first retrieval by the system. In IQBE, on the contrary, no feedback is needed from the user, and the query optimization is entirely based on off-line machine learning algorithms. In this paper we focus on the IQBE approach.

IQBE is a process for assisting the users in the formulation of queries, based on the application of machine learning methods. IQBE, initially proposed by Chen et al. [2], applies an off-line learning algorithm in order to find a set of relevant documents among the initial set of documents provided by a user. IQBE has been the subject of an intensive research in the last few years, since it is a paradigm which might be very useful for fulfilling the user's needs in future information retrieval systems.

In the literature many different approaches to IQBE can be found, the majority of them based on metaheuristics algorithms, such as genetic algorithms (GA) [5], genetic programming (GP) [6], simulated annealing (SA) [7] or hybrid techniques involving several of these algorithms. The first work on IQBE was the work by Chen et al. [2], where a standard GA with binary encoding and fixed side was tested and compared with an SA approach to IQBE. In addition to this first work in IQBE, GAs have been profusely applied to improve query definition, for example in [8–11]. There are also approaches based on GP, for example [12,13], and on SA [14], where a hybrid-simulated annealing-GP has been presented.

In spite of the huge work carried out in improving metaheuristic algorithms for IQBE, there are still several unresolved problems related to algorithms performance (specifically in GA and SA), such as poor results obtained in large queries design or convergence issues, mainly the time needed for convergence. This last point has not been fully considered in the literature until now, since the majority of the works are focused on improving the performance of the algorithms in terms of the fitness function considered, whereas the computational cost and time consumption of the algorithms have been considered a secondary issue.

In this paper we propose two different techniques for reducing the convergence time of metaheuristics (GAs and SA) in boolean IQBE, by means of a reduction of the metaheuristics' search space. We consider the standard GA and SA algorithms, both using binary encoding, and the standard operators, selection, crossover and mutation in the GA [5] and bit mutation in the case of SA [7]. In this framework, we apply two techniques which allow a reduction of the convergence time of the algorithms. The idea is to test the two proposed techniques over simple, standard GA and SA algorithms applied to IQBE problems, by comparing the performance and convergence time of the algorithms with and without the proposed techniques.

First, we propose the use of a *compression–expansion* strategy. The objective of this strategy is evolving several bits at the same time, in such a way that, if the terms evolved at a time are significant enough, the algorithm is able to converge faster than the algorithms without compression scheme. In a second step, our algorithm refines the search in the expanded encoding, after removing the terms which have been eliminated in the compressed one. The compression–expansion encoding strategy proposed in this paper is inspired by the *building block* hypothesis in GAs [5]. Following the building block hypothesis, a GA constructs good solutions from small, high fitness sub-solutions. If, for a given problem, the building blocks can be identified in advance, evolving them together will considerably improve the performance of the algorithm used in this particular problem. The main difficulty is, however, that for a general problem it is nearly impossible to identify the building blocks in advance. However, IQBE is a special case, since the building blocks should be formed by those terms which are significant for the user.

The second technique we present in this paper is the so-called *restricted search*. The restricted search is an idea that has been successfully applied to the binary feature selection problem (FSP) in classification [15,16]. The FSP consists of removing the meaningless features of a given classification problem, which only introduce noise to the classifier, without giving any helpful information to perform the classification task. The FSP can be tackled by means of metaheuristics such as GAs or SA, by using a binary string encoding. Using this encoding, a 1 stands for a given feature to be considered for the classifier, whereas a 0 in the encoding means that the feature should be removed from the training set. The restricted search for the FSP then consists in fixing the number of features to be considered by the classifier, introducing operators for removing or adding 1s to the binary strings in the GA (see [16] for details on restricted search for the FSP). In this paper we consider the application of the restricted search to the IQBE problem. We will show that this technique is able to improve the performance of a GA and an SA for IQBE in terms of convergence time to the optimal solution.

The rest of the paper is structured as follows: Section 2 presents the more important previous work related to IQBE and metaheuristics. Section 3 describes the compression–expansion strategy and the restricted search technique for improving the performance of GAs and SA in IQBE. In Section 4 we show by means of computational experiments the different possibilities that these techniques offer for improving the metaheuristics considered. Section 5 concludes the paper by giving some final remarks.

2. IQBE: previous approaches and model used

2.1. Previous approaches

IQBE is a process for assisting the users of an information retrieval system in the formulation of queries, based on the application of artificial intelligence methods. It was first proposed by Chen et al. [2]. In that paper, the authors presented a GA for learning the terms which better represent a relevant document provided by the user. In this approach an SA algorithm was also presented as another option for implementing an IQBE process. This first framework for IQBE was extended in order to include boolean operators (AND, OR, NOT) in addition to the query terms [14,17]. In addition to the first work by Chen et al. in IQBE, GAs have been profusely applied to improve query definition, for example in the works by López et al. [8,18] and Robertson and Willet [9,19], Tamine et al. [10], or Yang and Korfhage [11].

Also, several variations of the GP algorithm, [6], have been applied to IQBE. In [13], two evolutionary IQBE techniques for boolean query learning are proposed and analyzed. First, a GP approach is tested, formed by a roulette-wheel selection and usual GP crossover and mutation. The second approach proposed in this paper is a binary-encoded GP. Each binary string represents an expression tree, and the crossover and mutation of the algorithm can be done as a standard genetic algorithm, with one-point crossover and random flip-bit mutation. The authors test these two approaches in two databases, showing that the binary-encoded GP is superior to the classical GP in one of them, whereas in the other one the GP outperforms the binary-encoded one. There are more approaches to IQBE dealing with GP and its variants. One of the first works applying GP to an information retrieval problem was the paper by Kraft et al. [20]. This paper describes the application of a GP algorithm to fuzzy information retrieval systems using relevance feedback. Another interesting approach is the paper by Smith and Smith [12], where another GP-based approach to IQBE can be found. In [14], Cordón et al. presented a simulated annealing programming

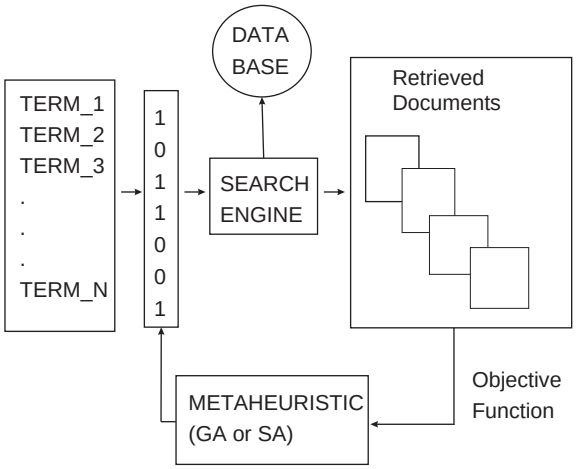

Fig. 1. IQBE example.

(SA-P) algorithm, which was compared with the approach by Kraft et al. The results reported in [14] show that the SA-P approach outperforms Kraft et al. GP in the IQBE problems tested.

Finally, there are some very interesting reviews in the literature which must be cited in this section of previous work. These papers cover the majority of the work published so far in the application of artificial intelligence techniques to information retrieval problems [3,4,17].

2.2. IQBE model used

In this paper, we have basically followed Chen's model for IQBE [2]. This model tries to learn the query terms that better represent a relevant document set provided by the user. In order to do this, Chen et al. proposed a GA and an SA which codify every possible set of terms as binary strings. Fig. 1 shows an example of this: each bit is associated with an existing term in the initial set of documents, in such a way that a 1 in the binary strings means that the corresponding term has to be included in the query, whereas a 0 in the strings means that the corresponding term has to be removed from the query. The objective function in both GA and SA is a measure of similarity between the set of documents in the database and the identified set of relevant documents obtained with the query defined by the binary string.

In spite of its simplicity, this model has important advantages for the comparison of the strategies proposed in this paper. First, it considers standard operators both in the GA and the SA. The majority of papers dealing with the metaheuristics algorithms on IQBE involves some specific operators or some tailoring in order to improve the algorithms' performance. Since our approach involves extra operators, maintaining the standard ones, the comparison of standard GA and SA with and without the implementation of the strategies presented in this paper is straight forward. This is also interesting in order to measure the average convergence time (in generations, or function evaluations) of the tested algorithms.

3. Improving metaheuristics convergence performance for IQBE

In this section we describe the two strategies presented for reducing the search space in IQBE problems. First we present the *compression–expansion* encoding strategy, and second, we introduce the *restricted search* strategy for IQBE.

3.1. The compression–expansion encoding strategy

The compression–expansion encoding strategy is basically a two-step strategy. In the first step (compression), several sets of κ bits each are defined, so that all bits in the same set evolve together. Note that the heuristic used in the search (GA or SA in this paper) will be run in a shorter encoding after the compression step. Specifically, the string size after the compression step will be $InitSize/\kappa$, where *InitSize* is the initial size of the binary encoding (total number of terms in the query). Note also that the compression only affects the specific operators of the heuristic applied (crossover and mutation in the case of an GA, and mutation in the case of an SA). For the calculation of the fitness function the string must be expanded. It is important to see that bits (terms) belonging to the same set will have the same status 1 or 0.

Using this strategy, it is expected that the heuristic algorithms for IQBE converge much faster in this compressed encoding. The number κ of bits which form each compressed set is a parameter of the strategy. Note that $\kappa = 1$ represents no compression in the algorithm. On the other hand, a very high parameter κ would lead to premature convergence and therefore to poor solutions after this first step.

The second step of this strategy (expansion) is a refinement step. The terms associated with bits 0 in the compressed string are removed (each bit equal to 0 means that a number κ of terms are removed[1]). Thus, the string size in this step will also be smaller than the initial one *InitSize*. After removing the terms associated with compressed bits equal to 0, the string is expanded again, and the algorithm is re-run in the new string for improving the quality of the final query. In this second step there is no expansion of the strings in the calculation of the fitness function.

3.1.1. Strategy implementation: sets formation

The key point of the compression–expansion strategy is the formation of the compression sets. As was stated before, the number of sets is fixed by the initial query length (*InitSize*) and the parameter κ. Once κ is chosen, the other important decision is how to put together groups of κ terms. The most straightforward way of doing this is grouping the terms by picking them randomly from the initial *InitSize* terms. However, it is expected that this way of grouping terms produces sets with the maximum of mixture between significant and nonsignificant terms. As we will show in the experiments section, this degrades the performance of the metaheuristics on the problem, so it does not seem to be a very good approach for obtaining the sets in our compression strategy.

Another approach that can be used for obtaining reasonably good sets in the compression step of the strategy is to put together terms which belong to the same topic. It is obvious that, even within the same topic, there will be significant terms and others not significant with respect to the documents provided by the user. However, this sets formation rule would discard quickly terms from topics not related at all to the

[1] This is completely true only if the initial size of the binary string *InitSize* is divisible by κ. In the case that it is not, the last set of terms in the compressed string will have less than κ bits.

documents required, and also would avoid the mixture of very different terms in the sets, which should be good for the compression strategy to work. Another advantage of this method of forming the sets is that some specific user's profiles might be used for putting the terms together, so that the classification of terms into different topics is not an added problem.

In this paper we will discuss the performance of the compression–expansion strategy once the sets are formed, the empirical analysis of the sets formation summarized in this subsection remaining for a future work.

3.1.2. Strategy implementation: outline of the heuristics tested

Compression–expansion in a genetic algorithm for IQBE: GAs [5] are a class of robust problem-solving techniques based on a population of solutions, which evolve through successive generations by means of the application of three genetic operators: selection, crossover and mutation [5]. GAs are useful to perform search in huge search spaces, where other methods (local or gradient searches) cannot provide good results. IQBE encoded with a binary representation is one of these cases. The standard GA modified for including the compression–expansion strategy can be outlined as follows:

Genetic algorithm

Initialize GA population at random
while(max. number of generations not reached) **do**
  for(every individual **x**)
    $(\mathbf{x} \rightarrow \mathbf{x}_{\text{expanded}})$
    evaluate$(f(\mathbf{x}_{\text{expanded}}))$;
  endfor
    selection
    crossover
    mutation
end while

where **x** stands for the current configuration in its compacted form (binary string of length $InitSize/\kappa$), and $\mathbf{x}_{\text{expanded}}$ for the current configuration in its expanded form (binary string of length *InitSize*). $f(\cdot)$ represents the objective function.

Compression–expansion in a simulated annealing for IQBE: SA is a heuristic which has been successfully applied to solve combinatorial optimization problems [7]. It is inspired by the physical process of heating a substance and then cooling it slowly, until a strong crystalline structure is obtained. This process is simulated by lowering an initial temperature by slow stages until the system reaches an equilibrium point, and no more changes occur. Each stage of the process consists in changing the configuration several times, until a thermal equilibrium is reached, and a new stage starts, with a lower temperature. The solution of the problem is the configuration obtained in the last stage.

The most important parts in an SA algorithm are: the chosen representation for solutions, the objective function to be minimized during the process and the mutation or configuration change operator. In IQBE, the representation and objective function have been given in Section 2.2, and the mutation operator consists of flipping a number of bits randomly chosen. The outline of this algorithm modified for including the

compression strategy is as follows:

Simulated annealing

$k = 0$;
$T = T_0$;
Initialize the current configuration $\mathbf{x}$ at random;
$(\mathbf{x} \rightarrow \mathbf{x}_{\text{expanded}})$: **evaluate** $\left(f(\mathbf{x}_{\text{expanded}})\right)$;
repeat
  for $j = 0$ **to** ξ
    $\mathbf{x}^{\text{mut}} = \textbf{mutate}(\mathbf{x})$;
    $\left(\mathbf{x}^{\text{mut}} \rightarrow \mathbf{x}^{\text{mut}}_{\text{expanded}}\right)$: **evaluate** $\left(f(\mathbf{x}^{\text{mut}}_{\text{expanded}})\right)$;
      if$\left(\left(f\left(\mathbf{x}^{\text{mut}}_{\text{expanded}}\right) > f(\mathbf{x}_{\text{expanded}})\right)$ OR $\left(random(0,1) < \mathrm{e}^{(-a/T)}\right)\right)$ then
      $\mathbf{x} = \mathbf{x}^{\text{mut}}$;
      endif
  endfor
$T = f_T(T_0, k)$;
$k = k + 1$;
until$(T < T_{\min})$;

where k counts the number of iterations performed; T maintains the current temperature; T_0 is the initial temperature; $T_{\min}$ is the minimum temperature to be reached; $\mathbf{x}$ stands for the current configuration (compacted form, length $InitSize/\kappa$), $\mathbf{x}^{\text{mut}}$ stands for the new configuration after a mutation operator is applied (compacted form), $\mathbf{x}_{\text{expanded}}$ is the current configuration in its expanded form, and $\mathbf{x}^{\text{mut}}_{\text{expanded}}$ stands for the new configuration after mutation, in its expanded form (binary string of length *InitSize*). $f(\cdot)$ represents the objective function; ξ is the number of changes performed with a given temperature T; f_T is the freezer function; and a is a previously fixed constant. Parameter a and the initial temperature T_0 are calculated in order to obtain the initial acceptance probability to be 0.8, which is the value commonly used. In this paper we assume that the probability of accepting worse solutions than the current one $\left(\mathrm{e}^{(-a/T)}\right)$ cannot be less than 0.001, so when $\mathrm{e}^{(-a/T)} < 0.001$ we use the value 0.001 as this probability.

The freezer function is defined as

$$f_T = \frac{T_0}{1 + k}. \tag{1}$$

The minimum temperature $T_{\min}$ is calculated on the basis of the desired number of iterations as

$$T_{\min} = f_T(T_0, numIt). \tag{2}$$

3.2. Restricted search in metaheuristics for IQBE

The restricted search strategy consists in fixing the number of desired words (1s in the binary string), by means of the so-called restricted search operator. This operator works in the following way: after the application of the crossover and mutation operators in the GA, or the mutation in the SA, the individual $\mathbf{x}$ will have p 1s that, in general, will be different from the desired number of desired words m. If $p < m$

the restricted search operator adds $(m - p)$ 1s randomly and if $p > m$, the restricted search operator randomly selects $(p - m)$ 1s and removes them from the binary string. This operator can be described in pseudo-code, as follows:

The restricted search operator

```
Select m (number of features) before running the GA or SA algorithms.
for every generation of the GA or SA:
    for every individual of the GA population or state of the SA:
        check the number of 1s p.
            if(p < m)
                Add_ones(m − p);
            else
                Remove_ones(p − m)
            end(if)
        end(individual)
end(generation)
```

Note that this operator forces the GA or the SA to search in a reduced search space. In fact, the standard GA and SA search in a space of size $2^{InitSize}$, whereas introducing the restricted search operator, the size of the search space is reduced to $\binom{InitSize}{m} \ll 2^{InitSize}$. The application of the restricted search operator implies consequently a faster convergence of the metaheuristics in IQBE, however, just like in the case of the compression–expansion strategy, there are solutions which are not being considered. The restricted search operator makes the GA or SA to look for the best IQBE solution containing *m* words, which can be different from the optimal solution, but it is expected that this difference would be small if an appropriate value of *m* is chosen.

4. Computational tests and analysis

In order to test our proposed techniques, we have used the well-known *Cranfield* documentary base. We have chosen this text collection because it has been used before in the majority of works dealing with metaheuristics for IQBE [3,8,18] for testing their performance, so it allows the comparison of results easily. The Cranfield documentary base has a total of 1398 documents related to all aspects of aeronautical engineering, with 225 associated queries. In order to test our techniques, we have chosen query number 294. First we have removed all the terms which do not retrieve any documents, retaining a total of 15 significant terms. We have then selected another 36 terms from the Cranfield vocabulary and added them to complete a new query containing a total of 51 terms, 15 significant and 36 not significant (only noise). The fitness function used (objective function for the SA) is the one given in [14], formed by combining the classical precision and recall measures:

$$F = \left(\alpha \frac{\sum_k r_k f_k}{\sum_k f_k} + \beta \frac{\sum_k r_k f_k}{\sum_k r_k} \right), \tag{3}$$

where $r_k \in \{0, 1\}$ is the relevance of document k for the user, and $f_k \in \{0, 1\}$ is the retrieval of document k in the processing of the query.

Note that the main objective of the computational tests carried out is to show how the metaheuristics considered perform better in convergence properties with the strategies for reducing the search space than without them. However, it is not easy to evaluate the convergence of metaheuristics, neither from a theoretical nor a practical point of view. We have considered the following framework for evaluating the convergence of metaheuristics:

First, we consider that the computational resources (in terms of objective function evaluations) are limited; let us assume that this limit is η function evaluations. If this number is low enough, the performance of the metaheuristic algorithms will not be optimal. The idea is then to study whether the strategies presented in this paper improve the performance of the algorithms within the same number of function evaluations. This would mean that the strategies improve the time that the algorithms need to achieve optimal solutions, and then their convergence is improved. In the first computational test carried onto we have run the GA and SA with the compression–expansion strategy. In a second test, we have used the restricted search in order to improve the convergence performance of the metaheuristics considered.

4.1. Computational tests using the compression–expansion strategy

In order to analyze the compression–expansion strategy proposed in this paper, we consider a division of the 51 terms of our query into 17 groups, by using groups of $\kappa = 3$ terms each[2] . The analysis of the strategy is carried on by varying the composition of the groups. The best possible scenario is to have 5 groups which include the significant terms, and 12 groups including the noisy words (recall that there are a maximum of 15 significant terms, and each group is formed by 3 terms). The worst one is to have significant and noisy terms mixed up all along the groups. Between these two extreme possibilities, there are other intermediate group structures: 1 group contains only significant words and the rest are mixed along the 16 remaining, 2 groups with significant words and the rest mixed in the 15 remaining, 3 groups and 14, and 4 groups and 13 are the last two possibilities. The GA and SA with the compression–expansion strategy are tested in the group structure stated before. In these experiments the parameters of the GAs were: a population of 50 individuals, maximum number of generations 200 (100 compression, 100 expansion), $P_c = 0.6$ and $P_m = 0.01$. And the parameters of the SA: $numIt = 200$ (100 compression, 100 expansion) with $\xi = 50$ iterations with each temperature. Note that with these parameters both algorithms perform the same number of function evaluations ($\eta = 10\,000$). All the figures shown are the average of 10 runs of the metaheuristics.

First we run the metaheuristics without implementing any strategy for reducing the search space. Figs. 2(a) and (b) show the performance of the standard GA and SA in the IQBE instance considered. Note that none of them is able to reach the optimal instance solution, and they need more generations (function evaluations) to achieve it. The next step for measuring the effect of the compression–expansion strategy is running some experiments with different groups structure, as explained in the previous paragraph. Fig. 3 shows the performance of the GA with the compression–expansion strategy in the worst possible case, when the significant and noisy terms are mixed up all along the groups. Fig. 4 shows it for the SA. Note that the performance of both metaheuristics considered is much worse using the compression–expansion strategy if the significant terms are mixed with the noisy ones in the groups. Figs. 5 and 6 show the

[2] With this value of κ we have an encoding of length $InitSize/\kappa = 17$ after the compression part of the strategy.

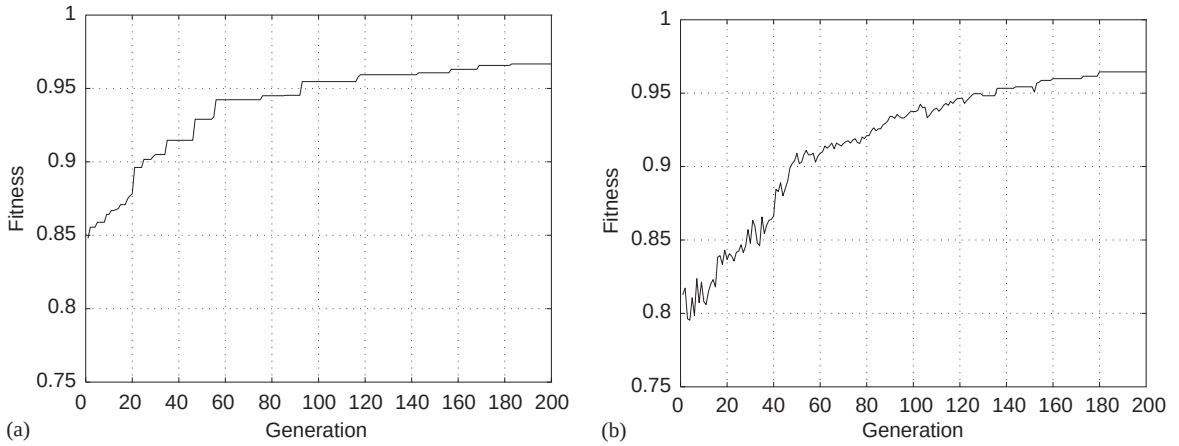

Fig. 2. (a) Performance of a standard GA on the IQBE instance tackled. (b) Performance of a standard SA on the IQBE instance tackled.

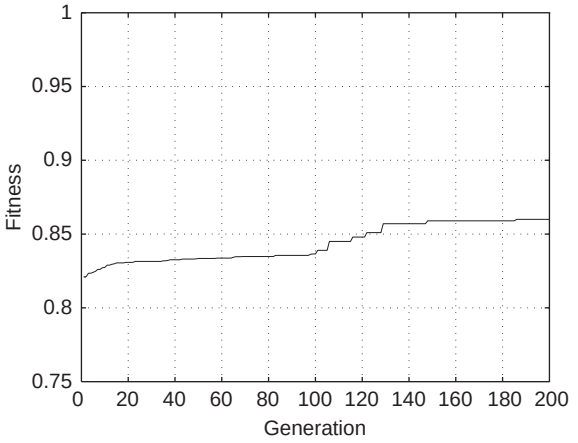

Fig. 3. Performance of the genetic algorithm with the compression–expansion strategy, in the case when the significant and nonsignificant terms are completely mixed up in the groups.

performance of the GA when some significant terms are put together in some of the groups. Note the improvement of the GA performance with respect to the case in which all the terms are mixed up in the groups. The same behavior is obtained using the SA, as can be seen in Figs. 7 and 8. Finally, if all the significant terms are arranged together in different groups from noisy terms, the performance of the metaheuristics using the compression–expansion strategy is much better than in the previous cases, obtaining the fast convergence of the metaheuristics to the optimal solution, as can be verified in Figs. 9 and 10 for the GA and SA, respectively.

Note that the compression–expansion strategy presented above has a main objective: speed up the convergence of the algorithms used in the IQBE process by removing as many terms as possible in the first step, whereas the refinement of the search is done in the second step, after expanding the string. It is

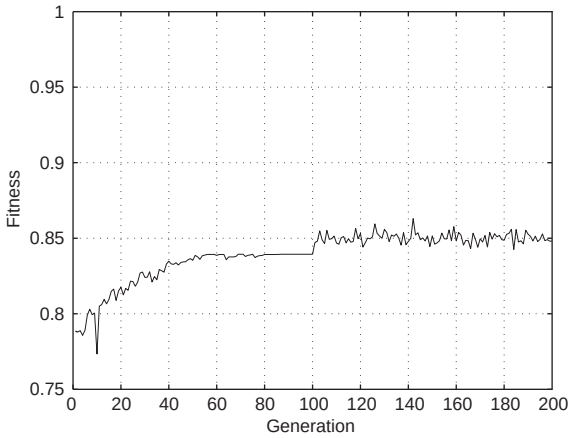

Fig. 4. Performance of the simulated annealing with the compression–expansion strategy, in the case when the significant and nonsignificant terms are completely mixed up in the groups.

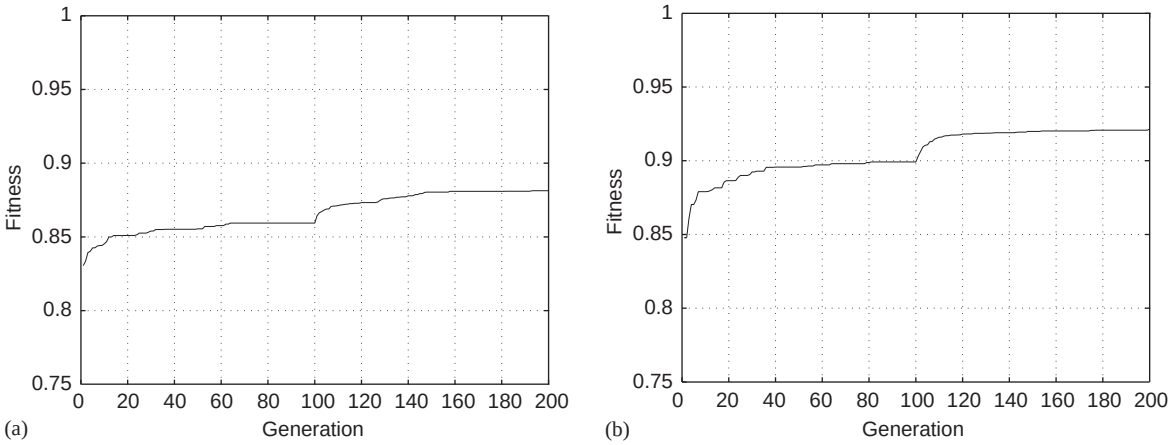

Fig. 5. Performance of the genetic algorithm when some significant terms are put together in some of the groups. (a) 1 group contains significant terms only, and the rest of the significant terms are mixed up in the remaining 16 groups; (b) 2 groups contain only significant terms, and the rest of the significant terms are mixed up in the remaining 15 groups.

easy to see that, if it is possible to identify which terms should be put together in order to evolve them at the same time, the strategy is going to be successful: the algorithm will eliminate quickly the noise terms and it will converge much faster than the same algorithm without this strategy. Moreover, if we would be able to put together all significant terms, and in separate sets, all the noise terms for the query the algorithm are likely to converge to the optimal solution of the problem in the first (compression) step. This would be the best possible case, where the compression strategy is optimal (Figs. 9 and 10). This is better explained if we compare the performance of the metaheuristics with and without the compression–expansion strategy. It is easy to verify that this strategy improves the convergence of the standard algorithms only if several significant terms are evolved together. In fact, it is possible to see

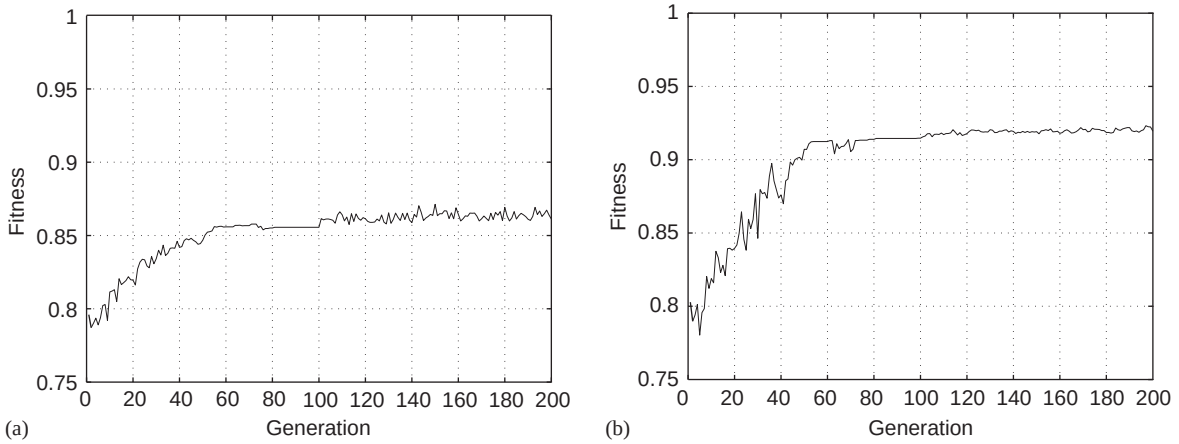

Fig. 6. Performance of the simulated annealing when some significant terms are put together in some of the groups. (a) 1 group contains significant terms only, and the rest of the significant terms are mixed up in the remaining 16 groups; (b) 2 groups contain only significant terms, and the rest of the significant terms are mixed up in the remaining 15 groups.

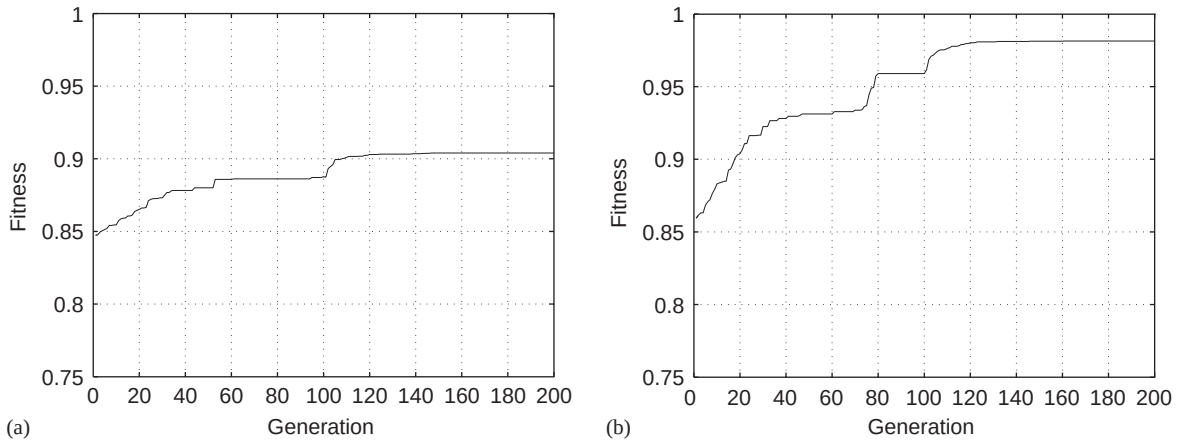

Fig. 7. Performance of the genetic algorithm when some significant terms are put together in some of the groups. (a) 3 groups contain significant terms only, and the rest of the significant terms are mixed up in the remaining 14 groups; (b) 4 groups contain significant terms only, and the rest of the significant terms are mixed up in the remaining 13 groups.

that when 4 out of 5 sets contain significant terms, or all significant terms are put together in groups, the metaheuristic with the compression–expansion strategy performs better than the standard metaheuristics within the same number of function evaluations, but, in the rest of the cases, this strategy does not achieve good results.

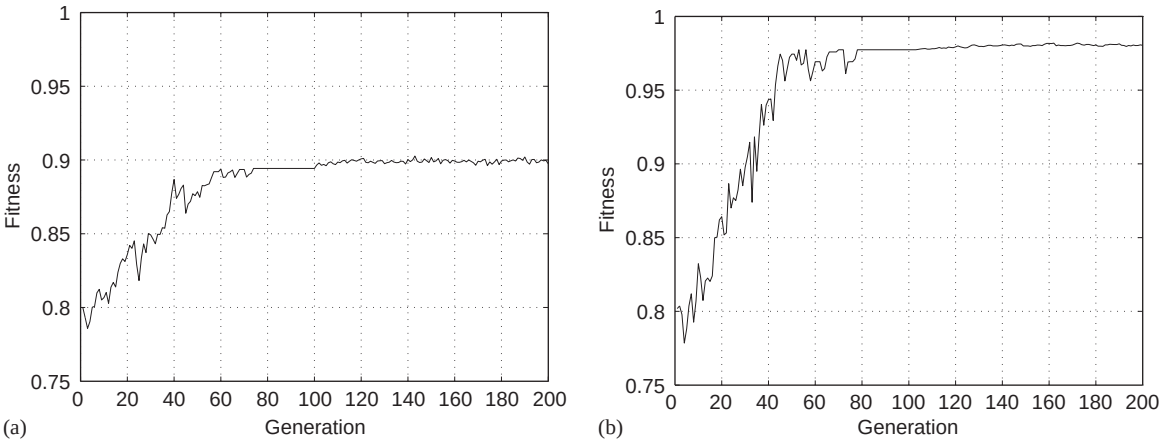

Fig. 8. Performance of the simulated annealing when some significant terms are put together in some of the groups. (a) 3 groups contain significant terms only, and the rest of the significant terms are mixed up in the 14 remaining groups; (b) 4 groups contain significant terms only, and the rest of the significant terms are mixed up in the 13 remaining groups.

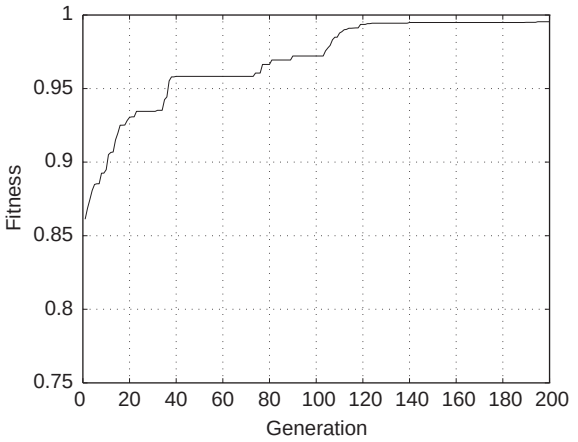

Fig. 9. Performance of the genetic algorithm when some significant terms are put together in some of the groups. In this case all the significant terms are put together, in 5 different group; the rest of the terms (12 groups) are nonsignificant.

4.2. Computational tests using the restricted search

In order to test the restricted search approach for IQBE, we have run the GA and SA fixing m to 5, 10, 15, 20, 25 and 30, and we have compared these results with the ones obtained by the metaheuristics without using the restricted search. In these experiments we have limited the number of function evaluations even more than in the experiments for the compression–expansion strategy. The parameters of the GAs were: a population of 50 individuals, maximum number of generations 50, $P_c = 0.6$ and $P_m = 0.01$. The parameters of the SA: $numIt = 50$ with $\xi = 50$ iterations with each temperature. Note that with these parameters both algorithms perform the same number of function evaluations ($\eta = 2500$).

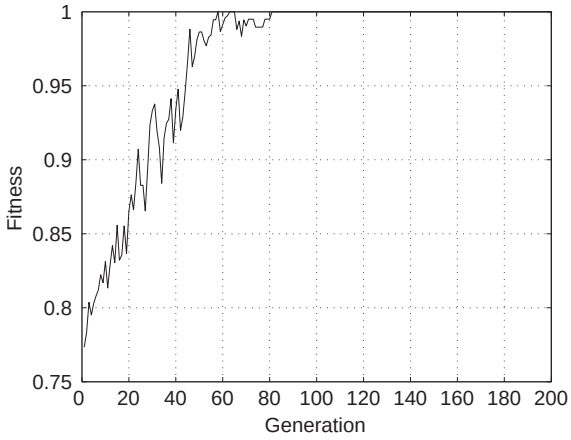

Fig. 10. Performance of the simulated annealing when some significant terms are put together in some of the groups. In this case all the significant terms are put together, in 5 different groups; the rest of the terms (12 groups) are nonsignificant.

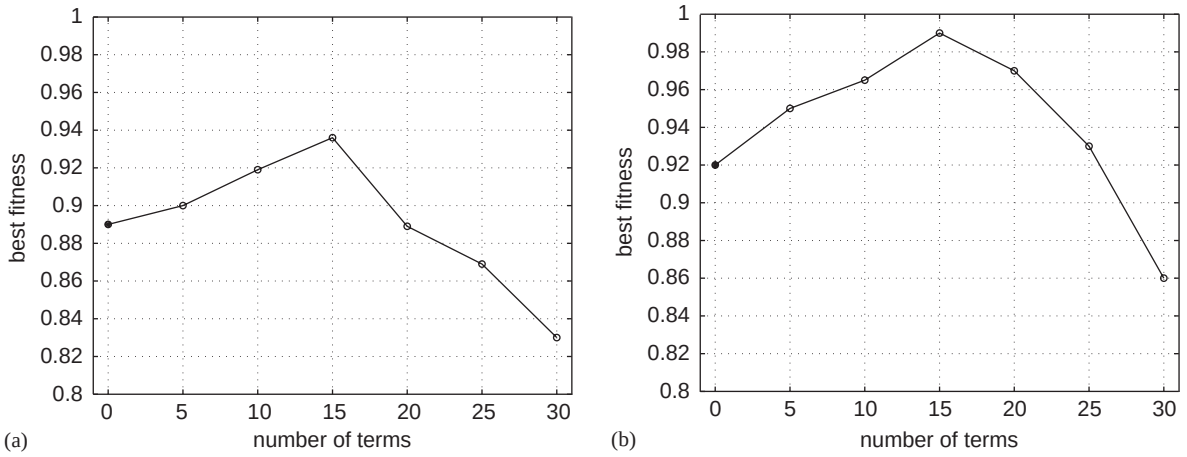

Fig. 11. (a) Performance of the genetic algorithm applying the restricted search for different values of m. The GA performance is optimum in $m = 15$ because there are 15 significant terms in the problem; (b) Performance of the simulated annealing applying the restricted search for different values of m. The SA performance is optimum in $m = 15$ because there are 15 significant terms in the problem.

Figs. 11(a) and (b) show the performance of the GA and SA, respectively, using the restricted search strategy. The value of 0 fixed terms (highlighted with a star inside the circle in the figures) is the performance of the corresponding metaheuristic without fixing any number of terms, that is, without implementing the restricted search strategy. Note that both GA and SA perform better when the number of terms is fixed to be the number of significant terms (faster convergence than the metaheuristic without restricted search strategy). Both algorithms without the restricted strategy would need much more generations (function evaluations) to obtain competitive results. The GA and the SA with the restricted search strategy achieve good results even with the small number of generations run. This means that

the restricted search considerably improves the convergence time of metaheuristics algorithms to IQBE. Another interesting point is that the SA seems to perform better than the GA in our IQBE tests, obtaining better results with the same number of terms fixed.

5. Conclusions

In this paper we have presented and analyzed two strategies for reducing the time of convergence of two metaheuristics (genetic algorithms and simulated annealing) in IQBE. The first strategy that we have presented is a compression–expansion strategy, where the terms are arranged into sets of a given size. We have shown that if the significant terms of the query can be put together in the sets, the performance of the metaheuristics is considerably improved. However, if the significant terms are mixed up in the sets with other non-significant words, the performance of the metaheuristics is degraded. The second strategy we have presented is the so-called restricted search, where the number of terms in the GA and SA encodings is fixed to be a given number m. In this case we have shown that the election of m is of great importance in the performance of the metaheuristics, the best election of m being equal to the number of significant terms in the query.

The two strategies presented in this paper can be used for speeding up the performance of GAs and SA in IQBE, or improving their performance in situations of limited computational resources. Both of them are based on a reduction of the search space size. We have shown some cases where they can be used successfully, and also the main drawbacks associated with them.

Acknowledgements

The authors really appreciate the comments on the paper by the anonymous reviewers and Professor G. Laporte, which have helped to improve the quality of this article.

References

[1] Caramia M, Felici G, Pezzoli A. Improving search results with data mining in a thematic search engine. Computers & Operations Research 2004;31:2387–404.
[2] Chen H, Shankaranarayanan G, She L. A machine learning approach to inductive query by examples: an experiment using relevance feedback genetic algorithms and simulated annealing. Journal of the American Society for Information Science 1998;49(8):693–705.
[3] Cordón O, Herrera-Viedma E, López-Pujalte C, Luque M, Zarco C. A review on the application of evolutionary computation to information retrieval. International Journal of Approximate Reasoning 2003;34:241–64.
[4] Mitra S, Pal K, Mitra P. Data mining in soft computing framework: a survey. IEEE Transactions on Neural Networks 2002;13(1):3–14.
[5] Goldberg D. Genetic algorithms in search, optimization and machine learning. Reading: Addison-Wesley, MA; 1989.
[6] Koza J. Genetic programming. Cambridge, MA: MIT press; 1992.
[7] Kirpatrick S, Gerlatt CD, Vecchi MP. Optimization by simulated annealing. Science 1983;220:671–80.
[8] López-Pujalte C, Guerrero-Bote VP, de Moya-Anegón F. A test of genetic algorithms in relevance feedback. Information Processing & Management 2002;38:793–805.
[9] Robertson AM, Willet P. An upperbound to the performance for ranked-output searching: optimal weighting of query terms using a genetic algorithm. Journal of Documentation 1996;52(4):213–32.

[10] Tamine L, Chrisment C, Boughanem M. Multiple query evaluation based on an enhanced genetic algorithm. Information Processing & Management 2003;39:215–31.
[11] Yang J, Korfhage R. Query modifications using genetic algorithms in vector space models. International Journal of Expert Systems 1994;7(2):165–91.
[12] Smith MP, Smith M. The use of genetic programming to build boolean queries for text retrieval through relevance feedback. Journal of Information Science 1997;23(6):423–31.
[13] Fernandez-Villacañas JL, Shackleton M. Investigation of the importance of the genotype-phenotype mapping in information retrieval. Future Generation Computer Systems 2002;19:55–68.
[14] Cordón O, Herrera-Viedma E, Luque M, Moya-Anegón F, Zarco C. An inductive query by example technique for extended boolean queries based on simulated-annealing programming. In: Proceedings of the 7th International ISKO Conference Granada. Spain, 2002. p. 429–36.
[15] Salcedo-Sanz S, dePrado-Cumplido M, Pérez-Cruz F, Bousoño-Calzón C. Feature selection via genetic optimization. Lecture Notes in Computer Science 2002;2415:547–52.
[16] Salcedo-Sanz S, Camps-Valls G, Pérez-Cruz F, Sepúlveda-Sanchis J, Bousoño-Calzón C. Enhancing genetic feature selection through restricted search and Walsh analysis. IEEE Transactions on Systems, Man and Cybernetics, Part C 2004;34(4):398–406.
[17] Pal K, Talwar V, Mitra P. Web mining in soft computing framework: relevance, state of the art and future directions. IEEE Transactions on Neural Networks 2002;13(5):1163–77.
[18] López-Pujalte C, Guerrero-Bote VP, de Moya-Anegón F. Genetic algorithms in relevance feedback: a second test and new contributions. Information Processing & Management 2003;39:669–87.
[19] Robertson AM, Willet P. Generation of equifrequent groups of words using a genetic algorithm. Journal of Documentation 1994;50(3):405–20.
[20] Kraft DH, Petry FE, Buckes BP, Sadasivan T. Genetic algorithm for query optimization in information retrieval: relevance feedback. In: Genetic algorithms and fuzzy logic systems. 1997. p. 155–73.