Selection of Information Sources Using a Genetic Algorithm

Fatma Zohra Lebib[1,2(✉)], Habiba Drias[1], and Hakima Mellah[2]

[1] USTHB, LRIA, Algiers, Algeria
zmatouk@cerist.dz
[2] CERIST, Algiers, Algeria

Abstract. We address the problem of information sources selection in a context of a large number of distributed sources. We formulate the sources selection problem as a combinatorial optimization problem in order to yield the best set of relevant information sources for a given query. We define a solution as a combination of sources among a huge predefined set of sources. We propose a genetic algorithm to tackle the issue by maximizing the similarity between a selection and the query. Extensive experiments were performed on databases of scientific research documents covering different domains such as computer science and medicine. The results based on the precision measure are very encouraging.

Keywords: Information sources selection · Distributed information retrieval · Bio-inspired methods · Genetic algorithms

1 Introduction

Even though the web can be seen as a single network of distributed repositories, many of traditional information retrieval approaches became difficult to put into practice. One of the most important raisons is the variety, heterogeneity and distributivity of information sources. The pervasive sources are asked for a query at the same time. This operation will certainly, returns a huge of information and consumes a considerable time. Distributed Information Retrieval (DIR) [1,2] provides a solution to the problem of searching on several dispersed information sources. The DIR system consists of three phases, namely source description [3], source selection [4,5], and result merging [6]. In the first phase, representations of available remote sources are created, containing important information about the sources such as their contents and their sizes. In the second phase, the DIR system selects a subset of sources which are most useful for users' queries. The source description is used to estimate the relevance of each source, and to classify sources accordingly. The third phase combines documents retrieved from selected sources into a single ranked list which will be presented to the users.

We aim in this paper to address the source selection problem in a context of a large number of sources. Previous research [7] showed that the source selection phase is vital for the overall effectiveness of DIR system. We formalized the

Á. Rocha et al. (eds.), *Recent Advances in Information Systems and Technologies*,
Advances in Intelligent Systems and Computing 569, DOI 10.1007/978-3-319-56535-4_6

problem of sources selection as a combinatorial optimization problem, which consists in finding the optimal combination (a selection) in a prohibitive search space containing all possible solutions (combinations). We search the solution, which maximizes similarity between sources composing a selection and the user query. We address this problem by the use of intelligent methods, in particular the Genetic Algorithms (GAs) [8], which are considered robust and efficient [9] and outperform the analytical methods for the large scale data [10].

2 Related Work

2.1 Sources Selection Approaches in a Distributed Environment

The first generation of source selection approaches, known as big document approaches, represents each source as a concatenation of its documents. The big documents obtained are classified according to their lexical similarity with the query using standard information retrieval techniques based on tf (term frequency) and idf (inverse document frequency). In sources selection, df (document frequency) is used instead of tf and icf (inverse collection frequency) instead of idf. The most well-known approaches are CORI [1,4] and GlOSS [11]. The second generation or small document approaches use a centralized index of sampled documents from different sources. The sources are selected based on the ranking of their documents for a given query. The documents relevance is estimated to classify sources according to the number and position of their documents in a centralized ranking. Examples of these approaches are CRCS [12] and [5,13]. Finally, a classification-based approach combines the above approaches with a number of other query-based and corpus-based features in a machine learning framework [7,14].

2.2 Genetic Algorithm in Information Retrieval

In the last few years, there has been a growing interest of designing GAs in different areas of Information Retrieval (IR) [15]. GAs were used to modify document descriptors [16] or user queries [17–22] and are used to optimize the web crawling [23,24], to optimize parameters independently of retrieval models [15,25]. Others works used GAs to address the adaptive information retrieval problem that relies on evolutionary user-modeling [26] and to generate and adapt user's profile for filtering documents that match the user's interests [27]. To our knowledge, very few works that address the DIR problem using evolutionary methods, for example the work in [28] authors proposed an algorithm to select the appropriate search engine in meta-search engine for the user query using GA. The proposed search engine selection algorithm is based on the relevance between a search engine and the query introduced by the user, which represents the fitness function of the proposed algorithm. His experiments are based on a simulation in MATLAB.

3 A Genetic Algorithm for Information Sources Selection

In this work, we propose an approach based on genetic algorithms to select with the optimal possible way, information sources to be interrogated.

3.1 Problem Definition

We define the problem of sources selection with a pair (instance, question) as follows:

Instance: S = $\{s_1, s_2, .., s_n\}$ a set of n information sources and the user query q.

Question: determine a subset S' of S such that the similarity between its elements and q is maximal, where $|S'| = k < |S| = n$; k is the number of selected sources for a query.

3.2 Search Space

The search space includes all possible solutions of the problem. As a solution is a combination of k sources from n total sources, the size of the search space is expressed as: $C_n^k = \binom{n}{k} = \frac{n!}{k!(n-k)!}$

When n is very large, the number of possible combinations is enormous and no complete method is able to yield a solution of good quality. One approach to cope with this issue is the use of artificial intelligence techniques such as a genetic algorithm.

3.3 Genetic Algorithm

We propose a Generic Algorithm for Sources Selection called GASS with the aim to find the optimal selection for a user query. GASS is initiated by a population of solutions represented by subsets of sources each, representing possible solutions to the problem. Each solution or chromosome is evaluated by the fitness function. Genetic operators (selection, crossing and mutation) are used to generate a new population from the current population. Once a new generation is created, the genetic process is repeated iteratively until an optimal solution or as default, a solution of good quality is found (Fig. 1).

Solution Encoding. A solution to the problem defined above is a set of k sources. The solution is represented by a vector of length k containing information sources. The latter will be encoded by integers to simplify their manipulation. Thus a source si is between 1 and n and a possible solution is a vector of k integers between 1 and n. For example, if the number of sources is equal to 5 and the number of sources to be selected is equal to 3, a solution can be: {1, 4,5} or {2, 3,5}, {3, 4,5} etc.

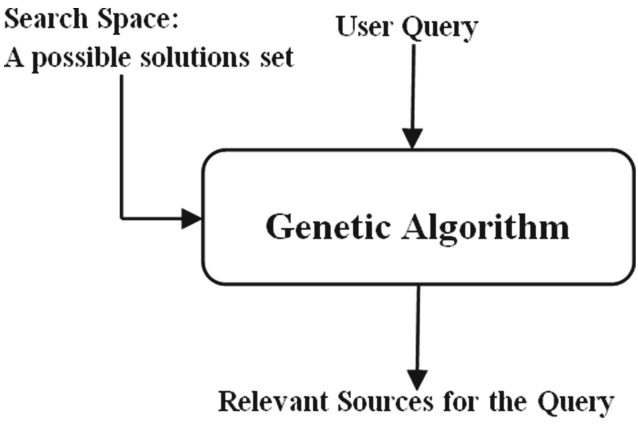

Fig. 1. Source selection approach

Fitness Function. The solutions evaluation function is a performance measure function that evaluates the quality of each solution. We evaluate a solution called *sol* by the average of the similarities between the sources of this solution and the user query. To calculate the similarity between a solution and the query q we consider *sol* as a collection of documents representing the sources. This similarity is calculated using the following formula:

$$Similarity(sol, q) = \frac{\sum_{h \in sol} Similarity(h, q)}{k} \tag{1}$$

Similarity (sol, q): similarity between a solution and the query q.
Similarity (h, q): similarity between source h in the solution and the query q.
k: number of sources in the solution.

The similarity between a source and the query can be calculated by the cosine measure of the vector search model. The source is considered as a set of terms. We represent the query and the source by vectors of terms weights in an m-dimensional space corresponding to the terms present in the search space. Thus the similarity between a source h and a query q is given as follows:

$$Similarity(s_h, q) = \frac{\sum_{j=1,m}(t_{hj} * t_{qj})}{\sqrt{\sum_{j=1,m}(t_{hj})^2 * \sum_{j=1,m}(t_{qj})^2}} \tag{2}$$

where t_{hj} and t_{qj} are the weights of the term j in the source h and query q respectively, calculated using the tf-idf approach [29] by replacing tf (the term frequency) with df and idf (inverted document frequency) with icf. It is defined as follows:

$$Weight(q/c) = df * icf \tag{3}$$

where df: document frequency, icf: inverse collection frequency, calculated as follows: log(N/cf). Where N is the number of all collections and cf is the number of collections that contain the term t.

Algorithm 1 outlines the proposed genetic algorithm for the selection of information sources.

Algorithm 1. GASS: a Genetic Algorithm for Sources Selection

Input: a set of n sources and a user query q,
Output: the optimal selection of sources ($sol_{optimal}$) for the query q
Process:
1: Generate randomly an initial population of *PopSize* size from the possible combinations of sources
2: Evaluate each solution in the initial population using the fitness function given by formula (1)
3: Create a new population
a. Select the appropriate chromosomes for reproduction (parents)
b. Apply the crossover operator to the parents according to crossover probability to produce new chromosomes (offspring)
c. Apply the mutation operator on the offspring chromosomes according to a mutation probability. Add the new chromosomes to the new population
d. While the new population size is smaller than current population size return to (a)
e. Replace the current population with the new population
4: Evaluate the current population using fitness function (1)
5: Check the termination criterion (maximum number of iteration is reached); If the criterion is not met go to (3)

In the following, we describe the different components of the algorithm.

Initial Population. The evolution process starts with an initial population of size *PopSize* generated randomly from the set of possible combinations. It consists of a set of chromosomes; each denotes a solution to the problem and is represented by a vector of k sources encoded by integer numbers from 1 to n. During the population generation, the same sources (duplicate genes) are avoided in the same chromosome. For example in the chromosome {2,5,3,2}, the number 2 is repeated, such construction of chromosomes must be avoided. We should also avoid to repeat the same chromosome in the population (duplicate chromosomes), for example {3,4,5,8} is the same as {8,3,5,4} because the order is not important.

Genetic Operators (a) Selection. The selection operator simulates the "survival-of-the-fittest". There are various mechanisms to implement this operator, and the idea is to give preference to the better chromosomes. We used natural selection that takes the best chromosomes in the next generation. The best chromosomes are identified by evaluating their fitness value.

(b) Crossover. It is a genetic operator that combines two chromosomes together to form new offspring. It occurs only with crossover probability Pc. Chromosomes that are not subjected to crossover remain unmodified. The intuition behind

crossover is the exploration of new solutions and exploitation of old solutions. We use single-point crossover without "duplicate". Thus, the gene values in the generated chromosome must not be repeated (see Algorithm 2).

Example. Let consider n = 9 and k = 6. S = {1, 2, 3, 4, 5, 6, 7, 8, 9}, S' is subset of S of length 6. Figure 2 shows an example of crossover.

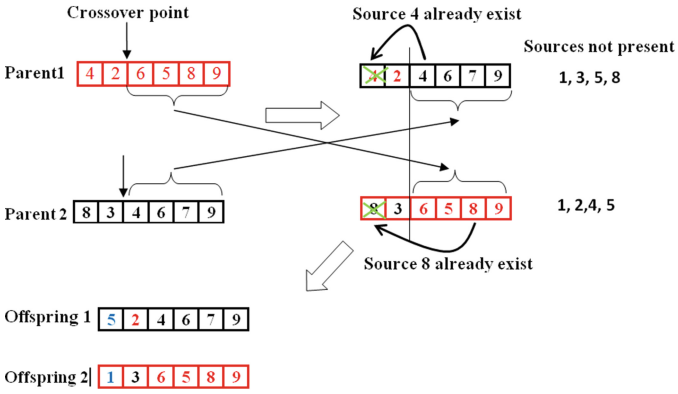

Fig. 2. Single-point crossover

Algorithm 2. Crossover Algorithm

Let Y = $(y_1, y_2, \ldots y_k)$ and X = $(x_1, x_2, \ldots, x_k)$ two chromosomes to be crossed

1: Choose a random number r on the set {0, 1, 2 ... k-1}, two new chromosomes X' and Y' are created according to the following rule:

$$x'_i = \begin{cases} x_i & \text{if } i < r \\ y_i & \text{otherwise} \end{cases} \qquad y'_i = \begin{cases} y_i & \text{if } i < r \\ x_i & \text{otherwise} \end{cases}$$

2: Remove, before the cutting point (r), the sources which are already placed after the cutting point

3: Identify sources that do not appear in each of the two chromosomes

4: Randomly fill the holes in each chromosome

(c) Mutation. Mutation is the process of randomly altering the genes in a particular chromosome. Mutation involves the modification of the gene values of a solution with some probability Pm. The objective of mutation is restoring lost and exploring variety of data. We used a single-point mutation. A gene is changed with a certain probability by a random number generated in the interval [1, n], while avoiding genes duplication (see Algorithm 3). Figure 3 shows example of mutation.

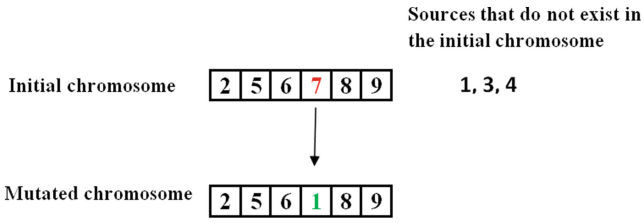

Fig. 3. A single mutation (on the 4th gene)

Algorithm 3. Mutation Algorithm

1: for each chromosome in the current population **do**
2: Generate a random number r on the interval [0 1] // *to select the chromosome //to be mutated*
3: if $(r < p_m)$ **then** // *apply the mutation operator to this chromosome*
4: begin
5: Select the gene to be modified (generate a random number i between 0 and k-1)
6: Choose the new value to be placed (generate a random number v between 1 and n; v must be different from the values already existing in the chromosome if not repeat the generation)
7: Change the value of gene i by the value v
8: Insert the new chromosome into the new population
9: end if
10: else insert the chromosome into the new population // *the chromosome is //inserted into the new population without change*
11: end while

There is no simple way to configure the GA parameters. We defined these parameters (population size, crossover and mutation probabilities) during the experiments.

Termination Criterion. The generation process is repeated until a termination criterion is reached. The termination criterion is the maximum number of generations to reach the convergence of the algorithm.

4 Experiments

We have implemented the proposed genetic algorithm in a java environment using JGAP[1]. In this section we describe the data and measures used in the experiments.

[1] Java Genetic Algorithms and Genetic Programming (http://jgap.sourceforge.net/).

Table 1. Sources test

Source number	Source	Domain
1	ACM Digital Library	Computer science
2	ClinicalKey	Medicine
3	Edward Elgar Products	Economics, finance, business and management, law and public policy
4	IEEE, Institute of Electrical and Electronics Engineers	Computer science, Electronics, Telecommunications
5	IOP science Extra of IOP Publishing	Physics, Materials Science, Applied Mathematics
6	JSTOR	Multidisciplinary
7	Royal Society of Chemistry	Chemistry, Materials Science, Environment, Biology
8	ScienceDirect of Elsevier	Multidisciplinary
9	SpringerLink	Multidisciplinary
10	SpringerProtocols	Science and technology, Life and earth sciences

4.1 Test Sources

We used databases of scientific research documents covering different domains (computer science, medicine, law...). The access to these libraries is ensured through a user account by a platform called SNDL[2]. The databases used are described in Table 1.

The Query-Based Sampling method (QBS) [3] is used to construct the sources description. Probe queries composed of a single term are sent to each of the sources. Queries are chosen according to the domains to which the sources belong. For each query (in a set of 15 queries) the top 4 documents are downloaded from the source. These documents are used to represent the source. We used Indri[3] to index these sources and search for documents. We set the following genetic algorithm parameters: Crossover rate: 60% Mutation rate: 10% and Generation number: 1500. A set of 20 test queries is selected manually. We choose general queries that return results and we avoid queries that do not return any response. We have varied the number of sources to select (k = 2, 4, 6, 8, 9).

4.2 Evaluation Function

Since all the relevant documents for a query is difficult to know for our data set we used the precision to evaluate our algorithm which is given by the following formula:

$$Precision = \frac{|Selected\ Relevant\ Sources|}{|Selected\ Sources|} \tag{4}$$

[2] https://www.sndl.cerist.dz/.
[3] Indexing and information searching system, http://www.lemurproject.org/indri.

Table 2. Average precision of GASS and CORI algorithms

Sources (k) algorithm	2	4	6	8	9
GASS	0.875	0.875	0.79165	0.733	0.68345
CORI	0.95	0.8375	0.76665	0.7125	0.68345

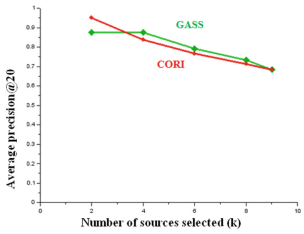

Fig. 4. Precision values for GASS and CORI algorithms on SNDL sources

To identify relevant sources, a user query is sent to each selected source and we only count the returned documents that are relevant. We analyze the first 20 documents returned by each source. We asked users to judge the relevance of the returned documents. A source is marked relevant if it returns at least 3 documents relevant to the query. The average precision is calculated over 20 test queries. We compared the proposed Algorithm (GASS) with the CORI algorithm [4]. The default parameters of the CORI algorithm are used.

4.3 Experiment Results

Table 2 shows the average precision reached by each algorithm over 20 queries. Figure 4 shows that the proposed algorithm is better than CORI algorithm in terms of precision. It can be concluded that the proposed algorithm provides a solution to sources selection problem in distributed environment being better or at least as efficient as other state-of-the-art source selection algorithms (CORI algorithm).

5 Conclusion

We have shown in this work how bio-inspired methods and more precisely genetic algorithms can provide solutions to the problem of source selection in a multi-source environment. First, we designed a genetic algorithm to find the best sources for the user query. Experiments showed a performance improvement in terms of precision of our algorithm in comparison with the CORI algorithm. This asserts that this approach can be efficient in distributed information retrieval. In future work, we plan to improve the experimental parameter values that allow to achieve better results. Further experiments are needed to demonstrate our proposal by increasing the number of sources.

References

1. Callan, J.: Distributed information retrieval. In: Croft, W.B. (eds.): Advances in Information Retrieval, pp. 127–150. Kluwer Academic Publishers (2000)
2. Shokouhi, M., Si, L.: Federated search. J. Found. Tren. Inf. Ret. **5**(1), 1–102 (2011)
3. Callan, J., Connell, M.: Query-based sampling of text databases. ACM Trans. Inform. Syst. **19**(2), 97–130 (2001)
4. Callan, J.P., Lu, Z., Bruce Croft, W.: Searching distributed collections with inference networks. In: 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 21–28. ACM, New York (1995)
5. Thomas, P., Shokouhi, M.: SUSHI: scoring scaled samples for server selection, pp. 419–426. ACM SIGIR, Singapore, Singapore (2009)
6. Shokouhi, M., Zobel, J.: Robust result merging using sample-based score estimates. ACM Trans. Inform. Syst. **27**(3), 1–29 (2009)
7. Cetintas, S., Si, L., Yuan, H.: Learning from past queries for resource selection. In: 18th ACM Conference on Information and Knowledge Management, pp. 1867–1870. ACM, New York (2009)
8. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley Publishing Company, Boston (1989)
9. Eiben, A.E., Smith, J.E. (eds.): Introduction to Evolutionary Computing. Springer, Heidelberg (2007). ISBN 978-3-540-40184-1
10. Drias, H., Khennak, I., Boukhedra, A.: Hybrid genetic algorithm for large scale information retrieval, pp. 842–846. IEEE (2009)
11. Gravano, L., Ipeirotis, P., Sahami, M.: GlOSS: text-Source discovery over the Internet. ACM Trans. Inf. Syst. **24**(2), 229–264 (1999)
12. Shokouhi, M.: Central-rank-based collection selection in uncooperative distributed information retrieval. In: 29th European Conference on Information Retrieval, Rome, Italy, pp. 160–172 (2007)
13. Markov, I., Azzopardi, L., Crestani, F.: Reducing the uncertainty in resource selection. In: Serdyukov, P., Braslavski, P., Kuznetsov, S.O., Kamps, J., Rüger, S., Agichtein, E., Segalovich, I., Yilmaz, E. (eds.) ECIR 2013. LNCS, vol. 7814, pp. 507–519. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36973-5_43
14. Hong, D., Si, L., Bracke, P., Witt, M., Juchcinski, T.: A joint probabilistic classification model for resource selection. In: 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval SIGIR, pp. 98–105 (2010)
15. Fujita, S.: Retrieval parameter optimization using genetic algorithms. Inf. Process. Manage. **45**(6), 664–682 (2009)
16. Gordon, M.: Probabilistic and genetic algorithms for document retrieval. Commun. ACM **31**(10), 1208–1218 (1988)
17. Ravi, S., Neeraja, G., Raju, V.: Search engine using evolutionary algorithms. Int. J. Com. Net. Sec. (IJCNS) **1**(4), 39–44 (2012)
18. Al Mashagba, E., Al Mashagba, F., Nassar, M.O.: Query optimization using genetic algorithm in the vector space model. Int. J. Comp. Sci. **8**(3), 450–457 (2011)
19. Sathya, A.S.S., Simon, B.P.: A document retrieval system with combination terms using genetic algorithm. J. Comp. Elect. Eng. **2**(1), 1–6 (2010)
20. Ibrahim, N.A., Selamat, A., Selamat, M.H.: Query optimization in relevance feedback using hybrid GA-PSO for effective web information retrieval, pp. 91–96. IEEE (2009)
21. Araujo, L., Pérez–Iglesias, J.: Training a classifier for the selection of good query expansion terms with a genetic algorithm. In: IEEE Congress on Evolutionary Computation, Barcelona, pp. 1–8 (2010)

22. Araujo, L., Zaragoza, H., Pérez-Aguera, J.R., Pérez-Iglesias-Iglesias, J.: Structure of morphologically expanded queries: a genetic algorithm approach. Data Knowl. Eng. **69**, 279–289 (2010)
23. Nhan, N.D., Son, V.T., Binh, H.T.T., Khanh, T.D.: Crawl topical vietnamese web pages using genetic algorithm. In: 2nd International on Knowledge and System Engineering, pp. 217–223 (2010)
24. Fan, H., Zeng, G., Li, X.: Crawling strategy of focused crawler based on niche genetic algorithm. In: 8th IEEE DASC, pp. 591–594 (2009)
25. Bhatnagar, P., Pareek, N.K.: A combined matching function based evolutionary approach for development of adaptive information retrieval system. J. Emerg. Tech. Adv. Eng. **2**(6), 249–256 (2012)
26. Maleki-Dizaji, S., Siddiqi, J.I.A., Soltan-Zadeh, Y., Rahman, F.: Adaptive information retrieval system via modelling user behaviour. J. Ambient Intell. Humanized Comput. **5**, 105–110 (2014)
27. Bouchachia, A., Lena, A., Vanaret, C.: Online and interactive self-adaptive learning of user profile using incremental evolutionary algorithms. Evolving Syst. **5**(3), 143–157 (2014)
28. Kumar, R., Singh, S.K., Kumar, V.: A heuristic approach for search engine selection in meta-search engine. In: International Conference on Computing, Communication and Automation (ICCCA), Noida, pp. 865–869 (2015)
29. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Inf. Process. Manage. **24**(5), 513–523 (1988)