Query Optimization by Genetic Algorithms

Suhail S. J. Owais, Pavel Krömer, and Václav Snášel

Department of Computer Science, VŠB – Technical University of Ostrava
17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic
vaclav.snasel@vsb.cz

Abstract. This study investigated the use of Genetic algorithms in Information retrieval in the area of optimizing a Boolean query. A query with Boolean logical operators was used in information retrieval. For Genetic algorithms, encoding chromosomes was done from Boolean query; where it was represented in the form of tree prefix with indexing for all terms and all Boolean logical operators. Information retrieval effectiveness measures precision and recall used as a fitness function in our work. Other Genetic algorithms operators were used as single point crossover on Boolean logical operators, and mutation operator was used to exchange one of the Boolean operators and, or, and xor with any other one. The goal is to retrieve most relevant documents with less number of nonrelevant documents with respect to user query in Information retrieval system using genetic programming.

1 Introduction

Information retrieval system is used to retrieve documents that depend on or relevant to the user input query. The growth in the number of documents in the internet made it necessary to use the best knowledge or methods in retrieving the most relevant documents to the user query. For this, Information Retrieval has become a fact of life for most internet users.

Information retrieval systems deal with data bases which is composed of information items documents that may consist of textual, pictorial or vocal information. Such systems process user queries trying to allow the user to access the relevant information in an appropriate time interval. Where the art of searching will be in the databases or hypertext networked databases such as internet or intranet for text, sound, images or data, see [1]. Thus an information system has at its heart a collection of data about reality [4].

Most of the information retrieval systems are based on the Boolean queries where the query terms are joined by the logical operators AND and OR. The similarity between a query and documents is measure by different retrieval strategies that are based on the more frequent terms found in both the document and the query. The more relevant document is deemed to be the query request. The most frequently used measures of retrieval effectiveness are **precision** *the percentage of the retrieved documents that are relevant* and **recall** *the percentage of the relevant documents that are retrieved.*

K. Richta, V. Snášel, J. Pokorný (Eds.): Dateso 2005, pp. 125–137, ISBN 80-01-03204-3.

Information retrieval is concerned with collection and organization of texts, responding to the requests of internet users for the information seeking text, retrieving the most relevant documents from a collection of documents; and with retrieving some of non-relevant as possible. Information retrieval is involved in:

- Representation,
- Storage,
- Searching,
- Finding documents or texts or images those are relevant to some requirements for information desired by a user.

Genetic Algorithms (GA) represents one of the artificial intelligence algorithms that are attractive paradigm to improve performance in information retrieval systems. Retrieving necessary documents from a collection of such documents will be achieved using a query to select the most relevant documents. For implementation of genetic algorithms will be on queries. A form of genetic algorithm started to be applied in information retrieval systems in order to optimize the query by genetic algorithms

2 Genetic Algorithms (GA)

Genetic algorithms (GA) first described by John Holland in 1960s and further developed by Holland and his students and colleagues at the University of Michigan in the 1960s and 1970s [10]. GA used Darwinian Evolution to extract nature optimization strategies that uses them successfully and transform them for application in mathematical optimization theory to find the global optimum in defined phase space [5, 3].

GA is used to find approximate solutions to difficult problems through a set of methods or techniques *inheritance or crossover, mutation, natural selection, and fitness function.* Such methods are principles of evolutionary biology applied to computer science. GAs are useful for:

- Solving difficult problems.
- Modelling the natural system that inspired design.

Applying genetic algorithms over a population of individuals or chromosomes shows that several operators are utilized. They are as follows (see Figure 1):

- **Fitness operator**, metrics to measure scheduler performance for each chromosome in the problem, and calculate the values for each chromosome.
- **Selection operator**, select two chromosomes with the highest quality values from the population, that couple to produce two offspring.
- **Crossover operator**, exchanges two subparts of the selected chromosomes, the position of the subparts selected randomly.
- **Mutation operator**, randomly changes the allele value in some location.

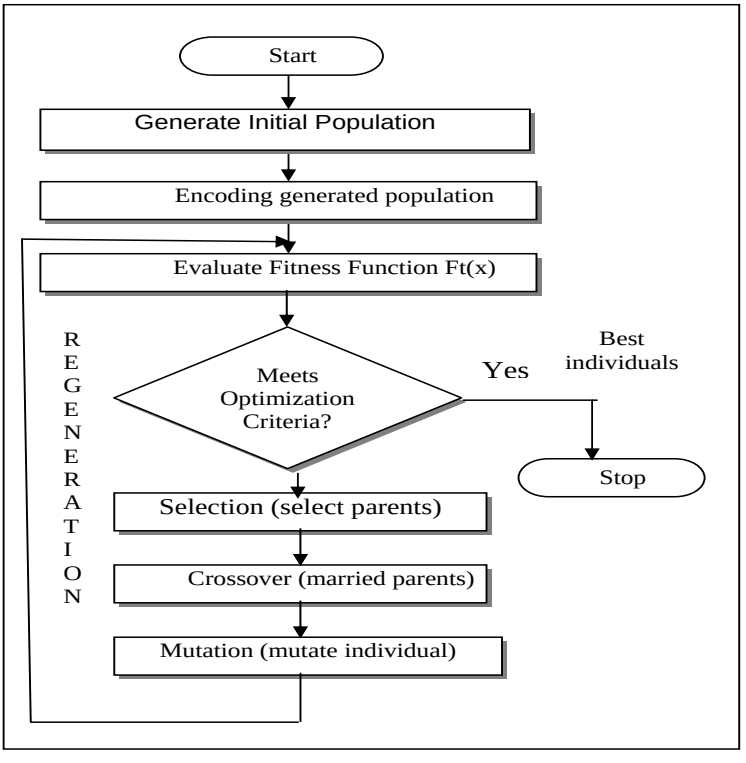

Fig. 1. Flowchart for Genetic Algorithm

3 Information Retrieval and Genetic Algorithm

This section will present the implementation of information retrieval using genetic algorithms (for SQL we can see [11, 8, 6, 2]). The GA is generally used to solve optimization problems [5]. GA starts on an initial population with fixed size of chromosomes "*P-chromosomes*". Each individual are coded according to chromosome length, where genes are allocated in each position in a chromosome with different data types, and each gene values called allele. In information retrieval, query for relevant documents are representing for each individual or chromosome, and each document described by set of terms. The description d_i for document D_i, where $i = 1 \ldots l$, the set of terms for D_i are T_j, where $j = 1 \ldots n$, thus $d_i = (w_{1_i}, w_{2_i}, \ldots, w_{n_i})$. The value for each term will be 1 if this term exists in the document or 0 if not (Note: about another weights for terms was mention in paper [9]), this indicate that the indexing function that is maps a given index term t and a given document d is

$$F : D \times T \rightarrow [0, 1].$$

Defining a query will be combination from set of terms and set of Boolean operators and, or, xor, not. The query set Q defined as set of queries for documents, define the query processing mechanism by which documents can be evaluated in terms of their relevance to a given query [7].

Ih this work, we develop genetic program for implementing GA with variable length of chromosomes and mixture symbolic of information, like real values and Boolean queries values.

4 Chromosome Encoding

In order to really understand the principle of chromosome encoding, consider a Boolean query with a series of terms $w_1, w_2, \ldots, w_n$, and set of Boolean operators from and, or, Xor , and Not. Examples of queries in infix form are:

$(w_2 \text{ or } w_6) \text{ and} (w_9 \text{ and } w_3)$
and
$(w_3 \text{ and } w_4) \text{ xor}((w_5 \text{ and } w_6) \text{ or } w_8)$

In the previous queries ordinary Boolean operators are used, and they are in infix form, they will be encoded to be chromosomes for genetic programming but in prefix form such as

$\text{and}(\text{or } w_2 w_6)(\text{and } w_9 w_3)$
and
$\text{xor}(\text{and } w_3 w_4)(\text{or}(\text{and } w_5 w_6) w_8)$

and also they will be represented in a tree form as the chromosomes shown in Figure 2.

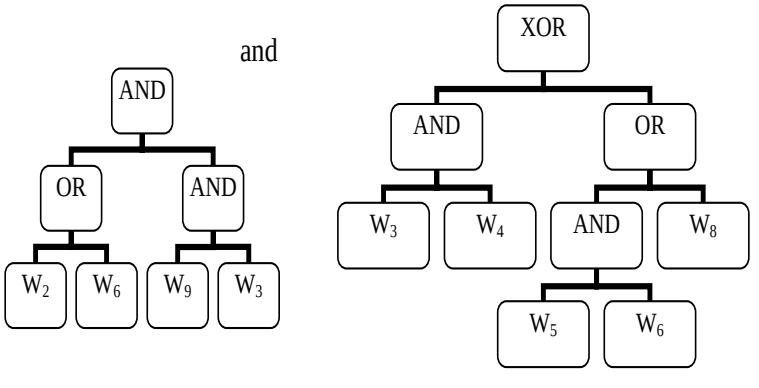

Fig. 2. Flowchart for Genetic Algorithm

5 Evaluation and Fitness Function

Evaluation of the information retrieval system is done by measuring its effectiveness. This is best measured by two statistics precision and recall, maximizing precision is subject to a constraint on the minimal recall accepted. Recall and precision are often perceived as being inversely related, i.e., complementary and competitive [7]. For any given set of documents called Collection of Documents, there is a subset of documents that are relevant to such query called Relevant documents, and a subset of documents that are retrieved called Retrieved Documents. Demonstration for precision and recall are shown in Figure 3 with respect to all documents collected. Where precision and recall are defined as:

$$Recall = \frac{RelevantRetrieved}{Relevant} \qquad Precision = \frac{RelevantRetrieved}{Retrieved}$$

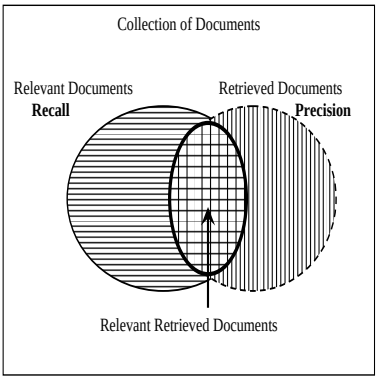

Fig. 3. Retrieved and relevant documents

Fitness function for our work will be considered as functions for precision and recall. One function will trade for the other function because both precision and recall are inversely related. And they will deal with the ranked documents. The shortcoming of using recall and precision as fitness functions is that if no relevant documents are retrieved by a chromosome, then its fitness is zero. This will lead to loss of all genes for this chromosome.

Computing recall and precision values for each query in our genetic programming as illustrated in equations 1 and 2, the first fitness function $RecallFitnessE_1$ measures recall (equ-1), and the second fitness function $PrecisionFitnessE_2$ measures Precision (equ-2). Where r_d is the relevance of document d (1 for relevant and 0 for nonrelevant), f_d is the retrieved document d (1 for retrieval and 0 for nonretrieval), and α and β are arbitrary weights. Where α and β are added specially to precision fitness function [7].

$$ReallFitnessE_1 = \frac{\sum_d [r_d \times f_d]}{\sum_d [r_d]}$$

$$PrecisionFitnessE_2 = \frac{\alpha \sum_d [r_d \times f_d]}{\sum_d [r_d]} + \frac{\beta \sum_d [r_d \times f_d]}{\sum_d [f_d]}$$

5.1 Selection operators

Processing genetic algorithm operators will be done in each generation over the best two chromosomes. From the population of chromosomes, the best two chromosomes depending on the highest fitness values for precision or recall measures will be selected. These two chromosomes will be called parent1 and parent2. These two parents will be used to produce two new offsprings.

5.2 Crossover or Recombination operators

In generating two new offsprings from the existing population, offsprings must have some inheritance from the two parents. Crossover will do that by exchanging subtree from parent1 with subtree from parent2. Positions for subtree1 and subtree2 will be selected randomly, and the position must be Boolean logical operator nodes in the tree; such as and, or, xor, and not. This will produce offspring1 and offspring2.

Single point crossover was used in our genetic programming. Index identifier was assigned for each term and each Boolean operator in a prefix form for each query that was encoded in a tree, see Figure 4.

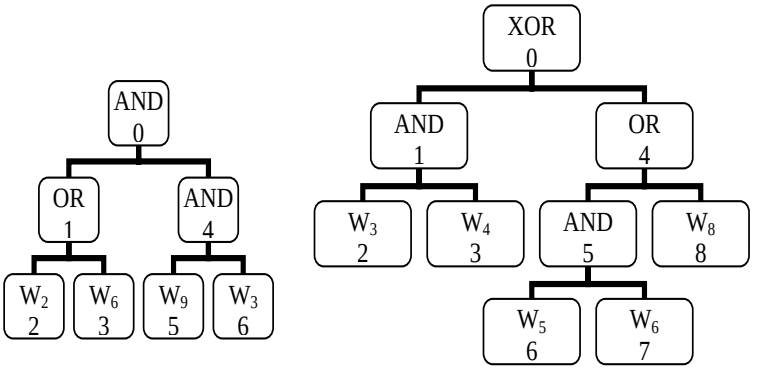

Fig. 4. Trees in Postfix form with indexing

For parent1, the selected subtree1 random number must not exceed the number of nodes of parent1. And same must be done for selecting subtree2 random number for parent2 must not exceed the number of nodes of parent2.

For example, if we have two random numbers **1** and **4** for subtree1 and subtree2 respectively, and we implement single point crossover process on parent1 and parent2 for trees shown in Figure 3, the new generated offsprings shown in see Figure 5, after exchange sub trees.

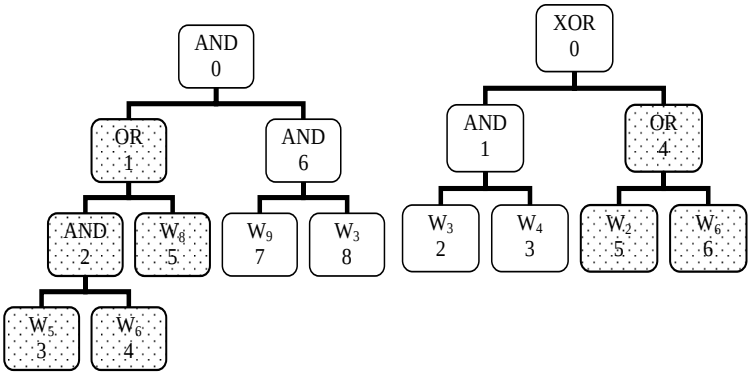

Fig. 5. New offsprings after single point crossover.

5.3 Mutation operators

Mutation, random perturbation in the chromosome representation, is necessary to assure that the current generation is connected to the entire search space, and it is necessary to introduce new genetic material into a population that has stabilized level [7]. In our genetic programming, mutation operator may change one Boolean logical operator by other one. That is for mutation will be for and, or and *Xor* Boolean operators.

For each new offspring, the selected random number is less than one. If the selected random number is less than mutation value, another selected random number will be drown and checking if it is on Boolean operator in the offspring, then exchange will process on it, (i.e. and will be or or xor). Example is shown in Figure 5, where mutation was not process on offspring1 but done on offspring2, where randomly if we select 1 to denote for the subtree position on offspring2, and from offspring2 we see that it is a Boolean logical operator and, and randomly it was changed to or, the new two offsprings are shown in Figure 5.

5.4 Updating Population

Fitness values will be computed for the two generated new offsprings, and after that the best chromosomes will survive for the next generation. Offsprings will be replaced by the worst chromosomes in the population, for this the new population

size will not be changed, replacing two worst chromosomes by two offsprings if the new offsprings fitness values exceed any of the chromosomes in the population.

The new population was ready to start next generation. And this will be repeated until number of generations or until we get the best solution for our problem.

5.5 Experiments

We developed our genetic program over a testing environment to see the influence of genetic algorithms on the information retrieval. The environment had a population from set of Boolean queries. For our program an input data used Boolean model of a collection of documents and set of Boolean queries as an initial population, and for execution the genetic program we used:

- Different Collections of documents with variant number of words and documents.
- Two sets of queries that represent as tree prefix form used as two different initial populations with fixed size in number of chromosomes (8 individuals).
- The results will be for a one requesting user query
- An initial values was fixed in all experiments are
  - Mutation value is 0.2
  - α is 0.25
  - β is 1.0
- Fixed number of generations are 50 generations

5.6 Limitations of current version

- At this time genetic operators are applied only for Boolean logical operators and, or and xor. This causes following: if queries in initial population do not include all the terms we have in the user query, they cannot come into existence.
- In our implementation, precision fitness value can be greater than 1.0, where the maximal value of precision is 1.25 that is coming from (α and β), thus we cannot interpret it as probability.

5.7 Experiment Tests

Our tests was done using user query

w_8 or w_2,

and we used two different initial populations Q_1 and Q_2; and they are shown in table 1, and table 2 respectively.

Table 1. Initial population Q_1.

No.	Query
1.	$(w_1 3 \text{ and } w_8) \text{ and}(w_{10} \text{ or } w_4)$
2.	$(w_1 \text{ and}(w_8 \text{ and } w_2)) \text{ or}(w_4 \text{ or } w_2)$
3.	$(w_1 \text{ or } w_2) \text{ and}((w_5 \text{ or } w_4) \text{ and}(w_3 \text{ and } w_6))$
4.	$(w_9 \text{ and } w_{14})$
5.	$(w_{14} \text{ and } w_1)$
6.	$(w_2 \text{ or } w_6) \text{ or}(w_8 \text{ and } w_{13})$
7.	$(w_3 \text{ and } w_4) \text{ or}((w_{12} \text{ xor } w_{15}) \text{ and } w_8)$
8.	$(w_1 \text{ or } w_5)$

Table 2. Initial population Q_2.

No.	Query
1.	$(w_{13} \text{ and } w_8) \text{ and}(w_{10} \text{ or } w_4)$
2.	$(w_1 \text{ and}(w_8 \text{ and } w_2)) \text{ or}(w_4 \text{ or } w_2)$
3.	$(w_1 \text{ or } w_2) \text{ and}((w_5 \text{ or } w_4) \text{ and}(w_3 \text{ and } w_6))$
4.	$(w_9 \text{ and } w_{14})$
5.	$(w_{14} \text{ and } w_1)$
6.	$(w_2 \text{ xor } w_8) \text{ or}(w_8 \text{ and } w_{13})$
7.	$(w_3 \text{ and } w_4) \text{ or}((w_2 \text{ or } w_8) \text{ and } w_8)$
8.	$(w_1 \text{ or } w_5)$

We mention that in initial population Q_1 there is in one query contain a sub expression

$w_8 \text{ and } w_2$,

and in initial population Q_2 there are in three different queries contains three different sub expressions

$w_8 \text{ and } w_2$, $w_8 \text{ or } w_2$, $w_8 \text{ xor } w_2$.

Our testing of execution for genetic programming was done independently from other execution results. Three cases were studied. The results of our tests with different collections (different number of documents and different number of words/terms). The three cases descriptions are:

- Test case 1: collection of 10 documents, 30 words in collection, maximal number of different words in a document is 10. Results are shown in table 3.
- Test case 2: collection of 200 documents, 50 words in collection, maximal number of different words in a document is 50. Results are shown in table 4.
- Test case 3: collection of 5000 documents, 2000 words in collection, maximal number of different words in a document is 500. Results are shown in table 5.

Table 3. Results of test case 1

No.	Selection for Parents depends on	Initial population	Result	Fitness value for Precision Recall
1	Precision	Q_1	(((w_8 or w_2) or (w_{13} and w_8)) or ((w_8 xor w_2) or (w_{13} and w_8)))	1.250000 1.000000
			(((w_8 or w_2) or (w_8 or w_2)) or (w_8 and w_2))	1.250000 1.000000
	Recall		((w_{13} or w_8) or (w_6 or w_2))	1.083333 1.000000
			((w_{13} and w_8) or ((w_6 or w_2) or ((w_{13} or w_8) or (w_6 or w_2))))	1.083333 1.000000
2	Precision	Q_2	((w_8 xor w_2) or (((w_{13} and w_8) or (w_8 xor w_2)) or (w_8 xor w_2)) or (w_8 xor w_2)))	1.250000 1.000000
			((w_8 or w_2) or ((w_{13} and w_8) and (w_8 or w_2)))	1.250000 1.000000
	Recall		(((w_8 xor w_2) or (w_8 xor w_2)) or (w_8 xor w_2))	1.250000 1.000000
			((w_{13} and w_8) or (w_8 xor w_2))	1.250000 1.000000

Table 4. Results of test case 2

No.	Selection for Parents depends on	Initial population	Result	Fitness value for Precision Recall
3	Precision	Q_1	((w_8 or w_2) or ((w_2 and w_4) or ((w_8 or w_2) and w_1)))	1.250000 1.000000
			((w_2 and w_4) or (((w_2 and w_4) xor (((w_2 and w_4) or (((w_4 or w_{10}) and (w_8 and w_{13})) or (w_8 and w_{13}))) and (w_8 and w_{13})) or (w_8 and w_{13}))) and (w_8 and w_{13})))	1.169580 0.678322
	Recall		(((w_2 or w_4) or (((w_2 or w_4) or (w_6 or w_2)) or (w_6 and w_2))) or (w_6 and w_2))	1.074290 0.930070
			(((w_2 or w_4) or (((w_2 or w_4) or (((w_{13} and w_8) or (w_6 or w_2)) and (w_8 xor w_2))) or (w_6 or w_2))) or (((w_{13} and w_8) or (w_6 or w_2)) or (w_8 xor w_2)))	1.101190 1.000000
4	Precision	Q_2	((w_{13} and w_8) or (w_8 or w_2))	1.250000 1.000000
			((w_8 or (w_8 or w_2)) or ((w_{13} and w_8) or (w_8 xor w_2)))	1.250000 1.000000
	Recall		(((w_{13} xor w_8) or (w_8 or w_2)) or (w_4 and w_3))	1.138199 1.000000
			((w_8 or w_2) or (w_8 or w_2))	1.250000 1.000000

Table 5. Results of test case 3

No.	Selection for Parents depends on	Initial population	Result	Fitness value for Precision Recall
5	Precision	Q_1	((w_{13} and w_8) and (w_{13} and w_8))	1.045644 0.182575
			(((((w_6 and w_2) or ((w_6 and w_2) or (w_8 and w_{13}))) and (((w_6 and w_2) or (w_8 and w_{13})) or (w_8 and w_{13}))) or ((w_6 or w_2) or (w_8 and w_{13}))) and (((w_6 and w_2) or (w_8 and w_{13})) or (w_8 and w_{13})))	1.084435 0.337739
	Recall		(w_8 or w_2)	1.250000 1.000000
			((w_{13} or w_8) or (w_6 or w_2))	0.913958 1.000000
6	Precision	Q_2	((w_8 and w_2) or (w_8 xor w_2))	1.250000 1.000000
			(w_8 or (w_8 or (w_8 or w_2)))	1.250000 1.000000
	Recall		(((w_{13} and w_8) or (w_8 or (w_8 or w_2))) or (w_8 or (w_8 or w_2))) or (w_8 or w_2))	1.250000 1.000000
			(((w_{13} or w_8) and (w_8 xor w_2)) or ((w_{13} or w_8) or (w_8 xor w_2)))	1.025921 1.000000

6 Conclusions

The results of this study suggests that the final population composed of individuals having the same strength (quality) will have the same precision and recall values. The best individual result was randomly chosen as best. We conclude that the quality of initial population was important to have best result of genetic programming process, and the less quality of initial population caused worse results. This could be seen when we chose parents based on precision, and therefore recall was very small for the large collection.

We concluded that in order to get good results, we choose parents depending on the recall fitness values than the precision fitness values, but that will increase the number of Boolean logical operators for the final best results.

For the future work we suggest to use less number of prefix tree by identifying the Boolean logical operators only without identify index for terms. For selecting the best individual with less number of Boolean operators and less number of terms instead of random selection if we got the final population with same strength. Modifying our system to work with different Boolean operators (of, adj, ... see in [4]), and extend this for fuzzy logic.

References

1. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison Wesley, New York, 1999.
2. Freytag, Johann Christoph: A Rule-Based View of Query Optimization. Proceedings of ACM-SIGMOD, 1987, pp. 173-180.
3. Goldberg, David E.: Genetic Algorithms in Search, Optimization and Machine Learning. Reading, Massachusetts: Addison-Wesley, 1989.
4. Korfhage Robert R.: Information Storage and Retrieval. John Wiley & Sons, Inc. 1997.
5. Melanie M.: An Introduction to Genetic Algorithms. A Bradford Book The MIT Press 1999.
6. Kim, Won: On Optimizing an SQL-like Nested Query. ACM Transactions on Database Systems 7, 3 (September 1982), pp. 443-469.
7. Kraft, D.H., Bordogna, G., and Pasi, G.: Fuzzy Set Techniques in Information Retrieval, in Bezdek, J.C., Didier, D. and Prade, H. (eds.), Fuzzy Sets in Approximate Reasoning and Information Systems, vol. 3, The Handbook of Fuzzy Sets Series, Norwell, MA: Kluwer Academic Publishers, 1999.
8. McGoveran, David: "Evaluating Optimizers." Database Programming and Design. January 1990, pp. 38-49.
9. Salton, G. and Buckley, C.: Terms-Weighting approach in automatic text retrieval. Information Processing and management, 1988 24(5):513-523.
10. Suhail S. J. Owais.: Timetabling of Lectures in the Information Technology College at Al al-Bayt University Using Genetic Algorithms. Master thesis, Al al-Bayt University 2003, Jordan. (in Arabic).
11. Yao, S. Bing: "Optimization of Query Algorithms." ACM Transactions on Database Systems 4, 2 (June 1979), pp. 133-155.