Scientific Research and Essays Vol. 5(13), pp. 1576-1582, 4 July, 2010
Available online at http://www.academicjournals.org/SRE
ISSN 1992-2248 ©2010 Academic Journals

Review

Optimizing N relations join queries by genetic algorithm

Najmeh Danesh[1], Hossein Shirgahi[2*] and Homayun Motameni[1]

[1]Department of Computer, Islamic Azad University, Sari Branch, Sari, Iran.
[2]Department of Computer, Islamic Azad University, Jouybar Branch, Jouybar, Iran.

Accepted 4 June, 2010

An important subject in optimizing queries at N relations join is the huge cost time. In web-based systems, the most important problem is obtaining answer at a minimum time. At the rank aware queries, suppliant does not need all possible answers rather having top K answers is enough. Although obtaining top K answers is time consuming in huge databases, in this paper we introduce a new concept at obtaining suitable K answers for rank aware queries. These suitable K answers are not top K answers necessarily, but they are much close to those answers. The operation is obtaining M (primal population) with length-N from the value of K that is determined in the query, N relations that are used in the query and by using genetic algorithm (GA). Then we apply GA on this primal set and repeat this algorithm until the average of different ranking value between sets elements in two sequential steps becomes less than a threshold level. So we can answer N relations join queries for obtaining suitable top K answers efficiently. We implement the proposed method and from the results obtained it was concluded that the query process time is reduced from 35 - 55% compared to traditional methods. Also, in the proposed method, by comparing the obtained suitable K answers with top K answers in traditional methods, they coincided approximately to 80%.

Key words: Relations join, query optimization, relational database, rank aware query, top K, suitable K, genetic algorithm.

INTRODUCTION

Emerging applications which are dependent on rank aware queries need an efficient support of rank aware queries in database management systems in real world. Supporting rank aware queries can give an efficient reply to information retrieval queries of database systems. During recent years, the combination of information retrieval systems and database systems is the main goal for many researchers. Database systems can supply data management with powerful integrity and compatibility assurance. An important issue in this field is that if we want to achieve joint operation between relation join in a traditional way, we need $O(N^L)$ time complexity for joining these relations with L relation records in average, whereas, we do not need most of these answers. For solving big time complexity problems, we can use evolutionary methods to reduce time complexity. The point is that these evolutionary algorithms do not supply best answers. But they try to select suitable answers based on user's query and defined parameters.

To answer a query with top K answers, the traditional method, N relation joins first. Then it sorts the answers based on ranking function and selecting top K answers.

The implementation of this method is very easy but there is no need to have a lot of tuples that have no role in final answer. By using a suitable strategy, the additional tuples should have been pruned. In firmware environments, there are some efficient algorithms to answer these queries (Fagin, 1996; Fagin and Lotem, 2001). In Marian and Bruno (2004), an efficient algorithm for processing queries with top K answers is produced by increasing paralleling and minimizing response time of query. Another method that is used to answer ranking queries is an estimation of input values by using statistical relations, humdrum supposition and randomize

*Corresponding author. E-mail: hossein.shirgahi@gmail.com.

variables (Ilyas and Aref, 2006; Ilyas and Shah, 2004). Another innovation is making ranking rules in relational databases (Ilyas and Li, 2005). The other solution in this field is improving joint operation and using ripple join. It minimizes the query's run time with an acceptable accuracy. The main idea of ripple join is applying join algorithms based on ranking for supporting multiple queries with top K answers at relational databases (Ilyas and Aref, 2003). Of course there are some other methods to answer top k answers queries that achieve top K answers by transferring query optimizing problem to informed searches (Zhang and Hwang, 2006).

In other consequences, an efficient algorithm is produced for pruning the inputs to support top K answers in queries with two relation join (Jie and Liang, 2006). In survey methods in rank aware queries, it tried to achieve top K answers but in this paper, we introduce a new idea called suitable K answers in rank aware queries. We express a solution for this problem by using GA.

GA

GA is a tool which can help us to simulate evolutionary extension mechanism. This can be done by searching in problem space to find the suitable answer but not exactly optimal answer (Dianati and Song, 2002). GA comprises 6 steps as thus explained:

Encoding

Encoding process is one of the most important parts of solving problem by GA. Usually, when the GA algorithm is used for solving problems, it is difficult to find a suitable representation for the answer that can reply all problems related to solving problems by GA. This process that redounds producing the searching answer's population is the chromosome representation of problem's answers. There are different presentations for chromosome representation of problem's answer like binary string, alphabet string, permutation, etc (Jean, 2000).

Start

A population with M chromosome is created, the number and the length of every chromosome is dependent on problem's navigate. The production of every chromosome can be completely randomized or heuristic.

Valuation

After production of primal population valuation, an evaluation function gives a value to every number of this population. It is very important to find a good method for valuation and a suitable evaluation function in solving a problem by GA.

Creating new population

It is one of the most important steps for solving the problem by GA. In this step, a new population is created by GA like mutation, crossover and by infusion of evaluation system in the nature and then it is located in searching space because selecting the initial population and the dispersal at problem's searching space can not find new point at searching space. The basic operation of this important step is creating some new population by having a primal population M and each value is explained in the three steps below.

Restriction operation

As Darwin said, the best one will live spawn and generate new people. The GA solves the problems like that. Thus to do crossover operation in a population, we should select two chromosome among them to apply the crossover operation on them and spawn new children. Restriction is an important problem in GA. There are some different methods for restriction like randomize restriction, roulette wheel and rank (Rogers, 2002).

Crossover operation

It redounds to generate new population and distribute it in the problem space. This process pattern is based on people's genetic which can redound people's various differences in shape. It redounds population's dispersal in searching space about GA. The probability of doing crossover operation is a special number like α. It means that everybody participates in crossover operation with α probability. Crossover operation is selecting two chromosomes or people among the searching population as discussed before. Now, two selected chromosomes are crossed with each other, creating two new chromosomes or new child which are put among the population. You can see a view of primal searching population and secondary population after cross operation on two chromosomes in Figure 1. There are different methods for crossover operation like one_point, two_points, multiple_points and steady (Rogers, 2002). A sample of two_points crossover is shown in Figure 2.

Mutation operation

It is an unwelcome change in DNA's string coding that can be useful most of the time, but it is useless sometimes too. Perhaps the searching population in GA leads to local optima, here the mutation operation helps the

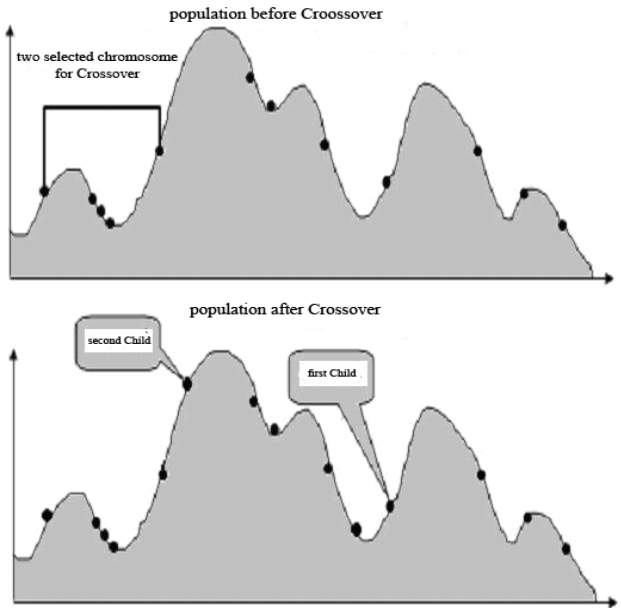

Figure 1. A view of crossover effect on population's dispersal.

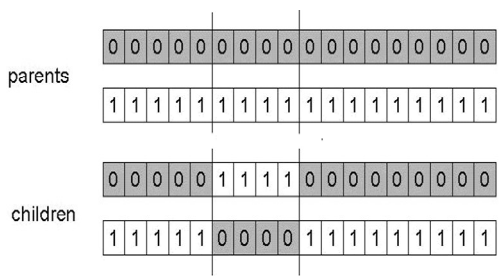

Figure 2. Two_points crossover.

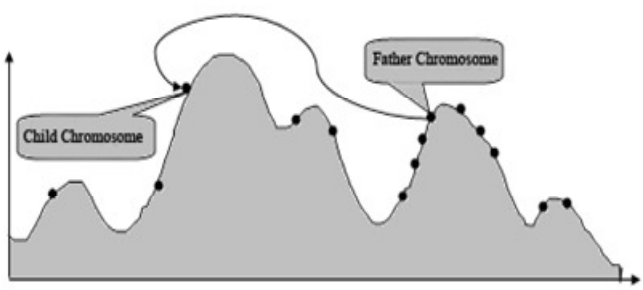

Figure 3. A view of mutation operation.

global optima answer. It banishes them from local optima by a mutation in chromosome. This method presents a completely randomize way for saving chromosome from this undesirable situation (Figure 3). There are different methods for mutation operation like one_point, two_points and multiple_points mutation Figure 4.

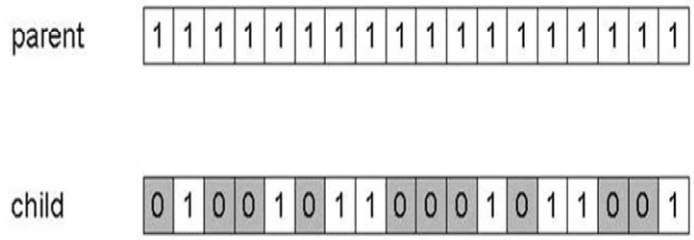

Figure 4. Multiple_points mutation.

Substitution

Creating new population and putting them in the problem space was done by genetic operations in previous sections. But if lasts continuously, the population will increase a lot, so after every creation of new generation the population will be decreased as primal population M. To do this, current population is sorted based on its fitness and the M most valuable will be selected as primal population for next step spawn and search.

Evaluation

GA's way of answering is different from other algorithms because first of all, it tries to continue its work by proposing some answers, even if they are not optimal. It does not try to find the best answers first. If these answers can reply problem's requirements, satisfy some limitations and supply user's need, the algorithm will be stopped, otherwise it will be kept on its work.

PROBLEM'S DEFINITION

When there is a rank aware query, its goal is achieving top K answers. Of course in this paper our goal is to achieve suitable K answers. In this case there is no need to make a relation between N relations completely because there are many records based on K, in the relations that have low concession. Usually they have no role in final result.

As much as possible, the records which have effect on final result should be selected for the join operation among N relations and prevent selecting ineffective records on final result. First we describe rank aware query and then express our idea about solving this problem by GA.

In rank aware query, the query is defined on x features $A_1,A_2,\ldots,A_N$ and N relations as $R_1,R_2,\ldots,R_N$ that every A_i(i=1:x) belongs to a relation R_j (j=1:N). Every feature has a special domain based on its type. With a query, some relations features are used for projection operation, some of them for restriction and join operation. Ranking

```
SELECT x
FROM House, School
WHERE ( DISTANCE ( House.Location, )School.Location) < d
ORDER BY ( House.Price + 5 x School.Tuition )
Suitable 10
```

Figure 5. Query sample.

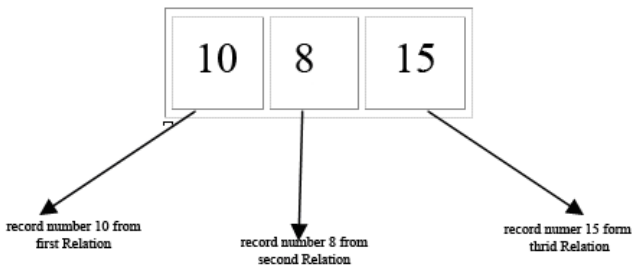

Figure 6. Chromosome's structure in proposed method.

function f is a combination of x' feature which x'<=x. We suppose that ranking function changes are steady as compared with all relations. Furthermore, the number of desired answers is determined which are suitable K answers in rank aware queries. We express a sample of rank aware query in example 1.

Example 1

A family tends to buy a house near a school and their goal is reducing costs. Consider a simple ranking function that estimates sum of house's price and school's tuition for five years. The searching operation in two relations of house and school in database should have been done based on query (Figure 5). The family needs at most ten results to decide among them, and not all results. Traditional method does join operation on two relations and then achieves all answers. Finally it sorts them based on ranking function and selects first ten answers. If the relations are huge and answers numbers are many, this method will have an expensive cost. However, this method cost is less for big Ks. In this paper there is just one change in rank aware query's definition; we have suitable K answers instead of top K answers.

In section 4 we express algorithm's analysis. In section 5 we describe problem's implementation and express experiments and their results on some sample queries. In section 6 resultants and final proposal will be explained.

ALGORITHM ANALYSIS

Input parameters in queries with N relations included N relations as $R_1,R_2,\ldots,R_N$. Every relation R_i(i=1:N) has features for R_i.projection_fields, Ri.restriction_fields, R_i.rank_fields which are determined with a user's query and relations. Furthermore these relations include some features for joining with other relations as R_i.join_fields[j],(j=1:N) which determine that every relation can join with which relations and by which features. Also there is a ranking function f with a steady increment as:

$$f(R_1.Rank\_Fields, R_2.Rank\_Fields, \ldots, R_N.Rank\_Fields) \quad (1)$$

Furthermore K is a desired user's answer. You can see a general algorithm in Figure 6 which is described in the following 10 steps:

1. First, we apply restriction operation based on restriction_fields of every relation R_1, $R_2,\ldots,R_N$ on them. So relation volume is optimized considerably.
2. If the information and queries are in distribution system or in computer networks, it is better to optimize their volume for information transmission. It is necessary to apply projection operation to remove some additional features which have no effect on query. To implement projection operation, we should select all necessary features for every relation and omit other features. After doing step 1, necessary features include rank_fields, projection_fields and join_fields will be individually for every relation. We consider them and omit other fields.
3. Every chromosome has N parts based on query. N is the number of relations in considering query. The value of each part is equivalent to record number of that relation which is in this chromosome (Figure 6).
4. Now the relations are optimized with a view to row and column to some extent. In this step we consider primal population of chromosome as M, M>=N. M primal population is selected from N relations randomly. So that it has one record number of every N relation in exchange for every chromosome. Of course for every record with N relations that is selected from every chromosome randomly, join conditions should be established between relations, otherwise this chromosome is not acceptable and should be created randomly. Another tip is that this created chromosome should not be repetitive.
5. Ranking function (Equation 1) obtains every chromosome value on the basis of record number which is in the chromosome equivalently. It is sorted based on value in ascending order.
6. In this step we should select some chromosomes based on the percent of population extension of every step. It is obvious that the probability of selecting better chromosome is more. We obtain the probability of being parent for every chromosome based on the value of every chromosome and by using rank power method per equations 2, 3. Rank power method gives a score to every chromosome. For example in a population of M, It gives value 1 to the worst chromosome, value 2 to the second worth chromosome and value M to the best chromosome. Finally selecting K^{th} chromosome's probability

is obtained from equation 3. As this, we assign multiple numbers from 1 to F to every chromosome based on its probability value. Now we select a random number from 1 to F by a randomize function, then a chromosome with this number will be selected.

$$F = 1^2 + 2^2 + 3^2 + ... + M^2 \quad (2)$$

$$\frac{Rank_K^2}{F} \quad (3)$$

7. In this step, crossover operation is done with a special probability like α. It means that every chromosome participates in crossover operation with probability α. The crossover operation selects two people or chromosomes in the searching population. We discussed the restriction operation in previous section. Now two selected chromosomes are crossed with each other by multiple_point crossover operation. In this paper we select cross number from 2 to N/2 randomly to have a higher performance. Cross location is randomized too.

Finally two new chromosomes or new children will be created by crossover operation and located among the population. Of course join condition should be established between relation for new children too, otherwise the crossover operation should have been done in another point of two chromosomes. If there is no situation for two chromosomes' crossover or join condition equality, any new child would not be created.

8. In this section, we select multiple_point mutation operation on the selected chromosome. The mutation operation is done with a special rate called Pm that has a value between 1/M and 1/N, so every bit of chromosome will be exposed to mutation with Pm probability.

In this section join conditions should be investigated for obtained children too.

9. In this section, the value of new child's chromosome is obtained on the basis of record numbers which are put in the chromosome equivalently. We select the most valuable M chromosomes as primal population in next steps among previous and new population.

10. In this section we do evaluation to understand if the obtained answers are suitable or not? Thus we use a heuristic method and calculate the average of chromosomes' value in M new and previous population. If the difference value is less than a threshold value, this value is determined based on user's query at first (Equation 4). It will mean the suitable answers are obtained and it is the end of our work, otherwise we should go to 5th step and continue our work again.

$$\frac{\sum_{i=1}^{M} f(chromosome_{New}[i])}{M} - \frac{\sum_{i=1}^{M} f(chromosome_{Prior}[i])}{M} < t \quad (4)$$

Maybe there are no K answers in some queries, so we show the maximum available answers. In this situation, the proposed method cost is more than the traditional method. Sometimes in this method, calculation cost is more than the traditional method for big Ks. When there is no join condition between two relations, they should be multiplied with each other to be joined. So the restriction operation is done without join condition.

IMPLEMENTATION AND EXPERIMENTS

We implemented the proposed method with Delphi 7 and SQL Server 2000. This system has some parts and facilities as follows:

1. A section to determine respective data base.
2. You can add a query to system.
3. We design an analyzer which achieves a query, analyses it, determines and obtains the relations and different parts of query which include restriction, projection, join and their conditions, ranking and suitable K answers.
4. We implement the algorithm as section 3 that is applied on the input query.
5. Finally we show the final result of query.

We run this system on some sample queries and different Ks and compare its result with the traditional system. We considered a database which is designed for keeping the information of a producer system. This system has 3 relations as below:

1. Producers' relation(s): it has producers' actual and potential information. Their attributes are number(s#), name, city, degree and credit of producer.
2. Goods relation: it has information about the goods which are going to be produced. Its attributes are number (p#), name, city of preservation place(city), degree, color and weight of good.
3. Productions relation: it has information about productions. Its attributes are producer's number(S#), goods number(P#) and quantity(QTY).

We created relations information by designing a randomized producer. S and P relations have about 400 records and SP relation has about 10000 records. We spot two queries (Figure 7) as a sample and ran the traditional and proposed methods for different Ks and then we studied the results. Figure 8 shows the cost comparison between traditional and proposed methods. We used simulation methods to obtain executive instruction's cost. Thus, we obtain executive instruction at running the query in a program and multiply it with the average cost of each instruction. Furthermore, we calculate the average of retrieval time of records by multiplying the retrieved records number with the average of a records retrieval time which has been done by SQL Server 2000 stored procedures. The cost time is the sum of these two times. Figure 9 shows the amount of accuracy of retrieved answers information in the proposed method towards

```
Select S.Name,P.Name
From S,P,SP
Where ((S.City = 'Tehran') and (S.S# = SP.S#)
 and (SP.P# = P.P#) and (SP.QTY > 10))
Order by (SP.QTY + 5*P.Degree + 2*S.Degree)
Suitable 10
```

Sample Query 1

```
Select S.Name,P.Name
From S,P,SP
Where ( (S.S# = SP.S#) and (SP.P# = P.P#)
 and ((SP.QTY > 200) or (SP.QTY < 100)) )
Order by ((SP.QTY / 5) + (5*P.Degree) + (S.Credit / S.Degree))
Suitable 10
```

Sample Query 2

Figure 7. Two sample queries.

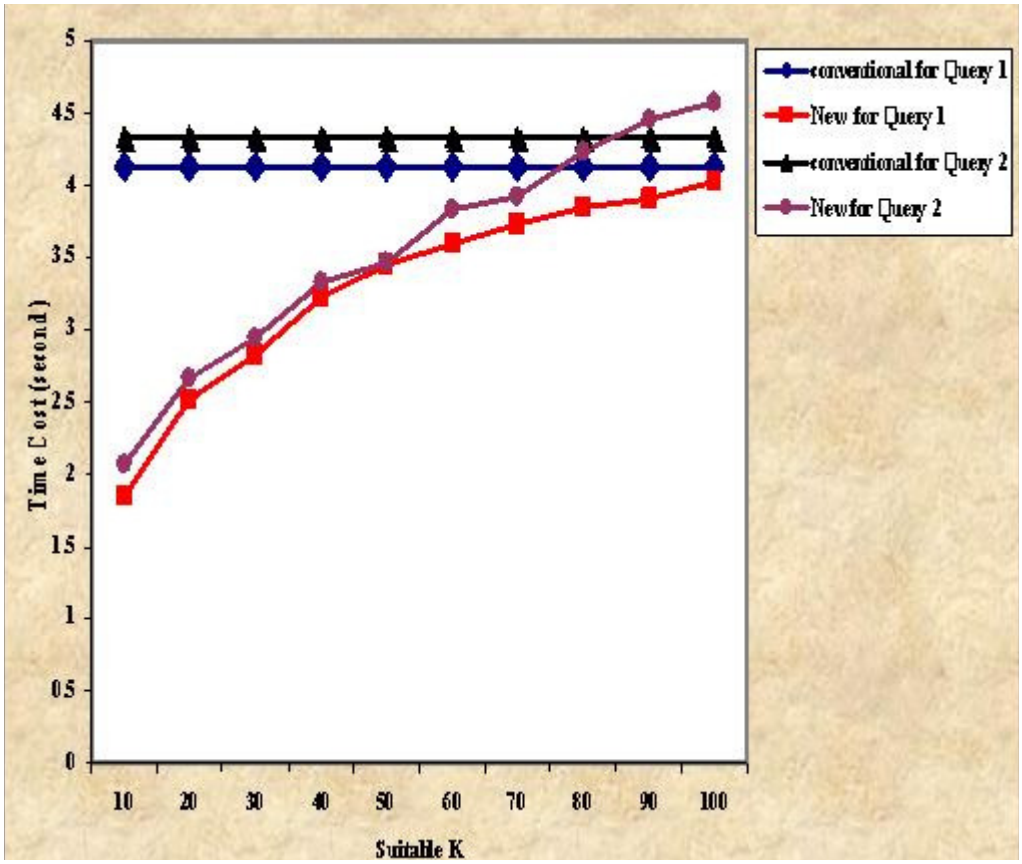

Figure 8. Cost comparison between traditional and proposed methods.

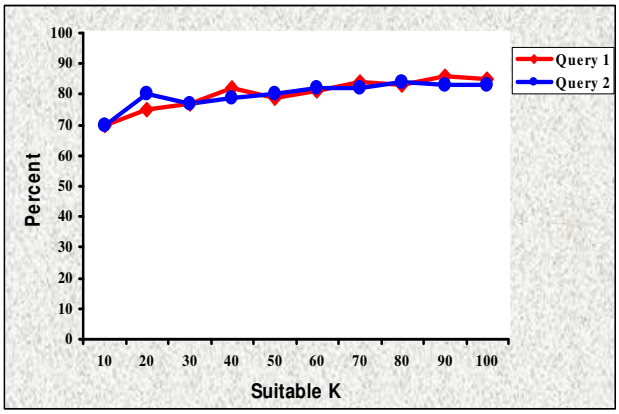

Figure 9. Obtained information's amount of accuracy towards top K answers.

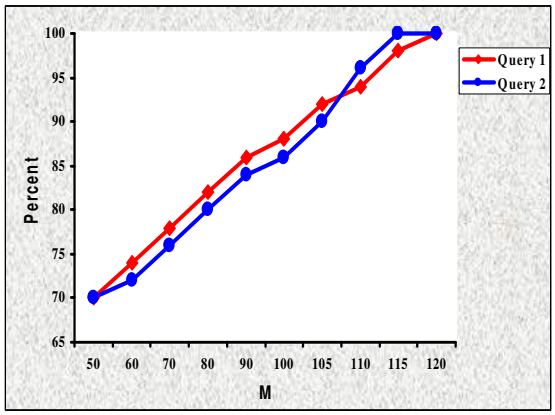

Figure 10. Obtained information's amount of accuracy for different values of M.

suitable K answers in the traditional method which shows it for different Ks. Figure 10 shows percent of obtained answer in exchange for K=50 and different values of M in the proposed method toward traditional method.

CONCLUSION AND PROPOSALS

The proposed method is suitable for big databases which have huge information and their goal is performing rank aware queries with small Ks. This method decreases time cost on the average between 35 to 55% according to Ks value and query. Also it achieves on the average 80 percent of best answers based on comparison done.

However, it tends to 100% as primal set increases. If K increases, the executive time cost will increase too. This method can be generalized for distri-buted and parallel databases as further works to use the advantages of this method better. Furthermore, some restriction methods can be used to improve the results (Dianati and Song, 2002). This method can be used for optimizing Fuzzy queries too.

ACKNOWLEDGEMENT

This work was supported by a research funding from the Islamic Azad University, Sari Branch, Iran. We thank the Islamic Azad University, Sari Branch for thier financial support.

REFERENCES

Dianati M, Song I (2002). An Introduction to Genetic Algorithms and Evolution Strategies". University of Waterloo, Canada.

Fagin R (1996). Combining Fuzzy Information from Multiple Systems," In Proceedings of PODS Montreal, Canada, June.

Fagin R, Lotem A (2001). Optimal Aggregation Algorithms for Middleware". In Proceedings of PODS, Santa Barbara, USA, May.

Ilyas IF, Aref WG (2006). Adaptive Rank aware Query Optimization in Relational Databases", ACM Transactions on Database Systems

Ilyas IF, Shah R (2004). Rank-aware query optimization", SIGMOD, Paris, France.

Ilyas IF, Aref WG (2003). Supporting Top-k Join Queries in Relational Databases". Proceedings of the 29th VLDB Conference, Berlin, Germany.

Ilyas IF, Li C (2005). Ranksql: Query algebra and optimization for relational top-k queries". SIGMOD June 14-16, Baltimore, Maryland, USA.

Jean P (2000). Genetic Algorithm Viewer: Demonstration of a Genetic Algorithm". SIGMOD, May.

Jie L, Liang F (2006). A Pruning-based Approach for Supporting Top-K Join Queries". ACM Edinburgh, Scotland.

Marian A, Bruno N (2004). Evaluating Top-K Queries over Web Accessible Databases". ACM Transactions on Database Systems, 29 (2): 319- 362.

Rogers A (2002). Symbolic AI Lecture 8 – Introduction to Genetic Algorithms". CM2408, December.

Zhang Z, Hwang S (2006). Boolean+Ranking: Querying a Database by K Constrained Optimization". SIGMOD, June 27. 29, Chicago, Illinois, USA.