ACSIJ Advances in Computer Science: an International Journal, Vol. 1, Issue 1, No. 1, September 2012
www.ACSIJ.org

Application of Bees Algorithm in Multi-Join Query Optimization

Mohammad Alamery *, Ahmad Faraahi *, H. Haj Seyyed Javadi **, Sadegh Nourossana *, Hossein Erfani *****

** Department of Information Technology, Payame Noor University, Tehran, Iran*

*** Department of Mathematics and Computer Science, Shahed University, Tehran, Iran*

**** Computer Engineering Department, Science and Research Branch, Islamic Azad University, Tehran, Iran*

Abstract. Multi-join query optimization is an important technique for designing and implementing database management system. It is a crucial factor that affects the capability of database. This paper proposes a Bees algorithm that simulates the foraging behavior of honey bee swarm to solve Multi-join query optimization problem. The performance of the Bees algorithm and Ant Colony Optimization algorithm are compared with respect to computational time and the simulation result indicates that Bees algorithm is more effective and efficient.

Keywords: Bees algorithm, Database Management System, Multi-join, optimization.

1 Introduction

One of difficulties in relational database management system (RDBMS) which has been solved faultily is multi-join query optimization (MJQO). In traditional applications of RDBMS, the number of join N involved by a single query is relatively small, Usually, N<10. With the expansion of the database application areas, the traditional query optimization technology cannot support some of the latest database applications. Such as, applications of decision support system (DSS), OLAP and data mining (DM), which may produce a query including more than 100 joins. In this condition, the shortfall of the traditional query optimization technology is exposed gradually. Therefore, it is necessary to explore new technology to solve MJQO problem.

MJQO is an NP hard problem [1]. With the increase of join number, the number of query execution plan (QEP) corresponding to a query grows exponentially, which lead to computational complexity of MJQO problem is very large. Recently, solving the problem with heuristic algorithm becomes a hotspot. Such as, ACO [1], Greedy Algorithm [2], GA [3], AB [4], etc. Several approaches have been proposed to model the specific intelligent behaviors of honey bee swarms and applied for solving combinatorial type problems [5–9]. In this paper, Bees algorithm was adopted to solve the problem MJQO.

2 Description for Multi-Join Query Optimization Problem

The process of RDBMS managing user query is as follows: After receiving query submitted by users, query parser checks syntax, verifies relations, translates the query into its internal form. It is usually translated into relational algebra expression, which can be denoted as query syntax tree. A relational algebra expression may have many equivalent expressions, so it also corresponds to many equivalent query syntax trees.

Then, query optimizer selects appropriate physical method to implement each relational algebra operation and finally generate query execution plan (QEP). The QEP consists of the order in which the operations in a query are to be processed, and the physical method to be used to process each operation. Amongst all equivalents QEP, query optimizer chooses the one with lowest cost output to the query-execution engine, then, the query-execution engine takes the QEP, executes that plan, and returns the answers to user. The process is depicted in Fig 1. This paper is to study how to make query optimizer select a QEP with lower cost in shorter time.

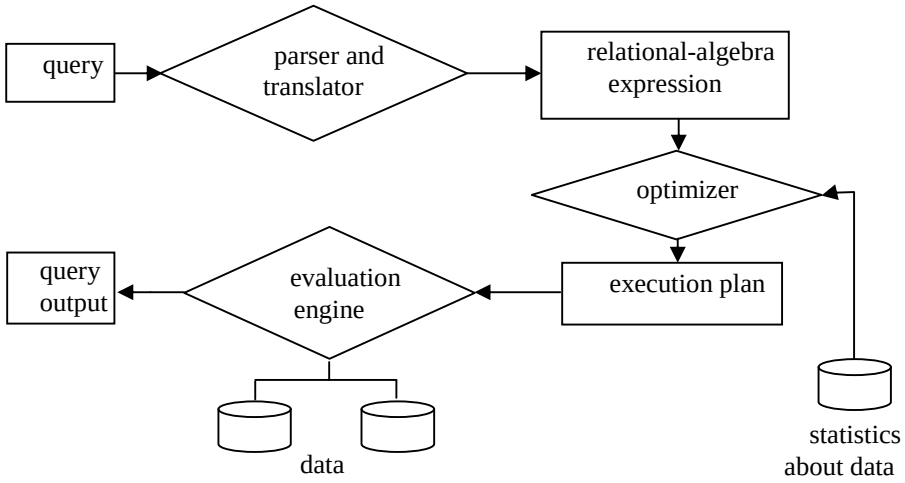

Fig. 1.Process of query execution

After being implemented optimizing operation of pushing down the select operation and project operation, one query that include project, select and join operations will transform to relational algebra expression constituted by N join operations which can be denoted as join tree. An example multiple query Q include A, B, C, D, E five relations, which can be denoted as three kinds of join tree shown in Fig 2: (a) left-deep tree (b) bushy tree (c) right-deep tree. The leaf nodes are relations constituting query Q and the internal nodes express join operation and intermediate results. Executive order is bottom-up execution. The different order of N join and the different physical methods selected to implement join operation lead to the cost of join trees have great differences. Assume that each join operations are implemented by the same physical method; Multi-join query optimization problem is simplified as setting a good join order, making the join tree has the lowest cost. Hence, tree in the left linear space can take full advantage of the index, and often contain the best strategy or the strategy whose cost is similar to the best strategy at least, therefore, consider left linear space as a search space.

In order to reduce the search space furthermore, avoiding the emergence of Cartesian product often is considered as constraint of the issue. An example multiple join query Q include A, B, C, D, E five relations. The attributes associating between five relations which are founded from statistical information of the database catalog, could be denoted as a query graph G= (V, E), shown as Figure 3. Nodes in query graph are relations and an edge connecting two relations, indicates attributes associating between two relations.

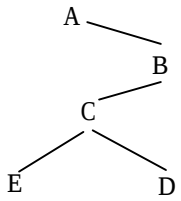

Fig. 3. Query graph

Q1 and Q2 are two join trees in the left linear space of query Q, depicted in Figure 4. Taking into account the "avoiding Cartesian product" restrictive conditions, Q1 do not accord with the restrictive conditions and Q1 is invalid join tree; Q2 do accord with the restrictive conditions and Q2 is valid join tree.

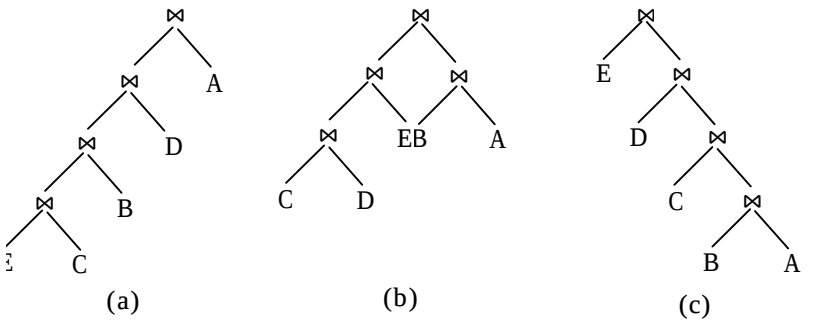

Fig. 2. Three Kinds of join tree

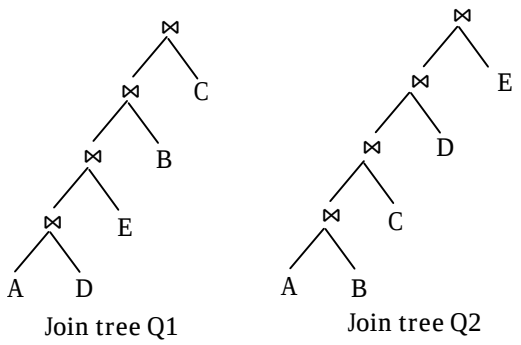

Fig. 4. Two join trees

Each relation in query graph corresponds to a set of parameters given by:

n(r) : Tuples number of relation r

V(C,r) : Number of distinct values for attribute C in relation r

In this paper, a simple model of the estimated cost is used, which applied in [1], based on two assumptions: Firstly, attribute values in symmetrical distribution. Secondly, the sum of the tuples number about intermediate results decides the cost of QEP. For example, t = r join s, C is public attribute over r, s. Then, *n(t)* and *V(A,t)* are defined by the following formulas:

$$n(t) = \frac{n(r) \times n(s)}{\prod_{C_i \in C} \max(V(C_i, r), V(C_i, s))} \quad (2.1)$$

$$V(A,t) = \begin{cases} V(A,r) & A \in r - s \\ V(A,s) & A \in s - r \\ \min(V(A,r), V(A,s)) & A \in r, A \in s \end{cases} \quad (2.2)$$

Assume that there are N relations in a join tree; the cost of QEP is the sum of the tuples number of internal nodes t_i in join tree. $n(t_i)$is number of tuples about intermediate result t_i . For a query Q, Z is collection of all the possible QEP corresponding to Q. Each member z in collection Z has query execution cost --- *Cost*(z) , then, Z_0 meeting $Cost(Z_0) \approx \min_{z \in Z} Cost(z)$ should be found.

3 Bees Algorithm for MJQO Problem

3.1 Bees Algorithm

The foraging bees are classified into three categories; employed bees, onlookers and scout bees [10]. All bees that are currently exploiting a food source are known as employed. The employed bees exploit the food source and they carry the information about food source back to the hive and share this information with onlooker bees. Onlookers bees are waiting in the hive for the information to be shared by the employed bees about their discovered food sources and scouts bees will always be searching for new food sources near the hive. Employed bees share information about food sources by dancing in the designated dance area inside the hive. The nature of dance is proportional to the nectar content of food source just exploited by the dancing bee. Onlooker bees watch the dance and choose a food source according to the probability proportional to the quality of that food source. Therefore, good food sources attract more onlooker bees compared to bad ones. Whenever a food source is exploited fully, all the employed bees associated with it abandon the food source, and become scout. Scout bees can be visualized as performing the job of exploration, whereas employed and onlooker bees can be visualized as performing the job of exploitation. In the Bees algorithm [11], each food source is a possible solution for the problem under consideration and the nectar amount of a food source represents the quality of the solution represented by the fitness value. The number of food sources is same as the number of employed bees and there is exactly one employed bee for every food source. This algorithm starts by associating all employed bees with randomly generated food sources (solution). In each iteration, every employed bee determines a food source in the neighborhood of its current food source and evaluates its nectar amount (fitness). The i^{th} food source position is represented as $Xi = (x_{i1}, x_{i2}, \ldots, x_{id})$. $F(X_i)$ refers to the nectar amount of the food source located at X_i. After watching the dancing of employed bees, an onlooker bee goes to the region of food source at X_i by the probability p_i defined as

$$p_i = \frac{F(X_i)}{\sum_{k=1}^{S} F(X_k)} \quad (3.1)$$

where S is total number of food sources. The onlooker finds a neighborhood food source in the vicinity of X_i by using

$$X_i(t+1) = X_i(t) + \delta_{ij} * u \quad (3.2)$$

where δ_{ij} is the neighborhood patch size for j^{th} dimension of i^{th} food source defined as

$$\delta_{ij} = x_{ij} - x_{kj} \quad (3.3)$$

where k is a random number $\in (1, 2, \ldots, S)$ and $k \neq i$, u is random uniform variate $\in [-1, 1]$. If its new fitness value is better than the best fitness value achieved so far, then the bee moves to this new food source abandoning the old one, otherwise it remains in it sold food source. When all employed bees have finished this process, they share the fitness information with the onlookers, each of which selects a food source according to probability given in Eq.(3.1). With this scheme, good food sources will get more onlookers than the bad ones. Each bee will search for better food

source around neighborhood patch for a certain number of cycle(limit), and if the fitness value will not improve then that bee becomes scout bee.

3.2. Pseudo code for Bees algorithm

1: Initialize
2: REPEAT.
3: Move the employed bees onto their food source and evaluate the fitness
4: Move the onlookers onto the food source and evaluate their fitness
5: Move the scouts for searching new food source
6: Memorize the best food source found so far
7: UNTIL (termination criteria satisfied)

3.3. Foraging (Neighborhood Search)

Each preferred path which a bee takes is a complete QEP which passes contains all relations. So each relation is connected to other relations who are called nearby neighborhoods of this relation.
The decision of each bee for changing these nearby neighborhoods results in the invention of new QEP which are considered as neighborhood QEP of the preferred path.
In the suggestive model of the neighborhood search, each bee tries to follow its own preferred path with the probability ω , and with the probability (1-ω) tries to make better paths by changing the nearby neighborhoods of its preferred path relation. The value of ω is calculated by Eq.(3.4).

$$w = \frac{Problem\ \ Size - Search\ \ range}{Problem\ \ Size} \tag{3.4}$$

Where *Problem size* is the number of all relations of the problem, and *Search range* is a positive parameter which identifies the extension of the neighborhood searching area.

This way, each bee begins to make a new QEP. It will be randomly located in a relation and selects the next relation by following the below rules:
(a) When a bee has decided to follow its preferred path, and none of the nearby neighborhoods have been visited. In this case it will choose one of them randomly and moves to it.
(b) When a bee has decided to follow its preferred path, but there is only one nearby neighborhood unvisited. So it will move to this unvisited relation.
(c) When a bee has decided to follow its preferred path, but all of the nearby neighborhoods have been already visited. In this case the bee will select the next relation based on the probability function (3.5).

$$I(i,j) = \begin{cases} 0 & j \in l \\ \dfrac{[m(i,j)][1/h(i,j)]^b}{\sum_{s=1, s\notin l}^{n} [m(i,j)][1/h(i,s)]b} & j \notin l \end{cases} \tag{3.5}$$

Where I (i , j) is the probability with which the bee moves from relation i to j , h(i , j) the distance between i and j relation, b positive parameter, whose values determine the relative importance of memory versus heuristic information, n the number of relations, and l a list of all the visited relations so far.
(d) When a bee has decided not to follow its preferred path and choose a new nearby neighborhood, in this case it will do the same as in rule c.

4 Experimental Results

In order to illustrate the effect of Bees-MJQO in solving this problem, experiments have been implemented on computer with Pentium4 2.93G + 1024 RAM + Windows XP Pro. A database including 50 relations that have attributes association with each other has been used as test data. ACO [1] and Bees are used to solve this problem respectively.

Bees	No. of bees = 16 No. of iterations= (No. of Relation)^2
ACO	No. of Ants = 10 α=1, β=3, q0=0.2, ρ = 0.9

Table 1. Algorithmic parameters.

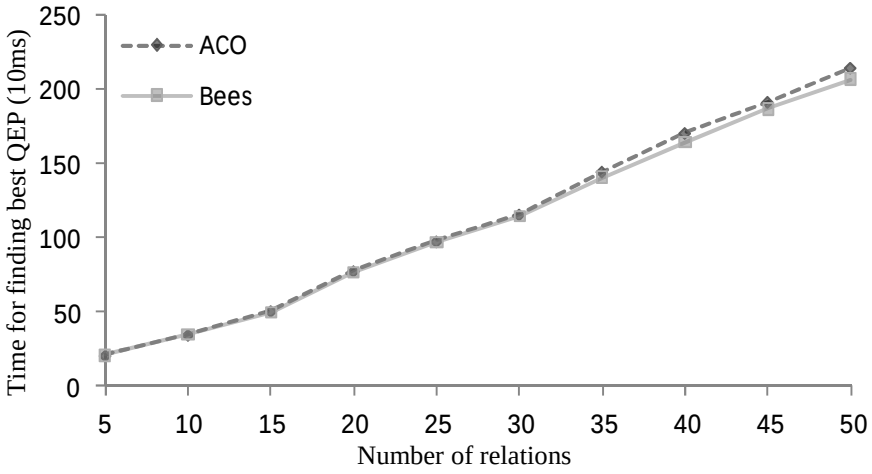

Fig. 5. Comparison of execution time

As is shown in Fig. 5, number of relations corresponding to query Q is taken as X-axis and time for generating optimal solution—query execution plan is taken as Y-axis. The simulation results show that Bees finds optimum solutions more effectively in time than ACO. The figure indicates ACO algorithm spends more time than Bees algorithms on finding optimal solution especially with the incensement of relation number.

5 Conclusions

MJQO problem is hotspot in database research field. A good optimization algorithm not only can improve the efficiency of queries but also reduce query execution costs. In this paper, the Bees algorithm, which is a new, simple and robust optimization algorithm, was proposed to solve the problem of MJQO. The performance of the proposed algorithm is compared with the ACO algorithm. The results reveal that Bees algorithms converge faster compared to ACO algorithm for this problem.

The simulation results show that Bees Algorithm finds optimum solutions more effectively in time than ACO especially with the incensement of relation number.

References

1. Li N., Liu Y., Dong Y., and Gu J. (2008) Application of Ant Colony Optimization Algorithm to Multi Join Query Optimization, Springer-Verlag Berlin Heidelberg.

2. Shekita, E., Young, H., Tan, K.L. (1993) Multi-join optimization for symmetric multiprocessors. In: Proc. Of the Conf. on Very Large Data Bases (VLDB), Dublin, Ireland: 479–492

3. Cao, Y., Fang, Q. (2002) Parallel Query Optimization Techniques for Multi-Join Expressions Based on Genetic Algorithms. Journal of Software 13: 250–256

4. Swami, A., Iyer, B. (1993) A polynomial time algorithm for optimizing join queries. In: Proc. IEEE Conf. on Data Engineering, Vienna, Austria: 345–354

5. Tereshko, V., Loengarov, A. (2005) Collective Decision-Making in Honey Bee Foraging Dynamics. Comput. Inf. Sys. J., 9(3): 1–7

6. Teodorovi´c, D. (2003) Transport Modeling By Multi-Agent Systems: A Swarm Intellgence Approach, Transport. Plan. Technol. 26(4): 289–312

7. Teodorovi´c, D., Dell'Orco, M. (2005) Bee colony optimization—a cooperative learning approach to complex transportation problems. In: Proceedings of the 10th EWGT Meeting, Poznan, 13–16 September 2005

8. Benatchba, K., Admane, L., Koudil, M. (2005) Using bees to solve a data-mining problem expressed as a max-sat one, artificial intelligence and knowledge engineering applications: a bioinspired approach. In: Proceedings of the First International Work-Conference on the Interplay between Natural and Artificial Computation, IWINAC 2005

9. Wedde, H.F., Farooq, M., Zhang, Y. (2004) Bee Hive: an efficient fault-tolerant routing algorithm inspired by honey bee behavior, ant colony, optimization and swarm intelligence. In: Proceedings of the 4th International Workshop, ANTS 2004

10. Sabat, S.L., et al. (2010) Artificial bee colony algorithm for small signal model parameter extraction of MESFET. Engineering Applications of Artificial Intelligence

11. Karaboga, D., Basturk, B. (2008) On the performance of artificial bee colony (ABC) algorithm. Appl. Soft Comput. 8 (3): 687–697