An Efficient Multi Join Query Optimization for DBMS Using Swarm Intelligent approach

Ahmed Khalaf Zager Al saedi
Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia
ahmedkhalafalsady@yahoo.com

Prov Madya Dr.Rozaida bt.Ghazali
Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia
roziada@uthm.edu.my

Prof. Dr. Mustafa Bin Mat Deris
Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia
mmustafa@uthm.edu.my

Abstract- **In the era of information technology, various professions are Multi Join Query Optimization (MJQO) in database management system (DBMS) such as Search engine, Data mining, Decision support system, Data warehouse and Banking system, Information retrieval (IR) make very important field to study .The increase in the amount of database and number of tables and blocks in database as well as inadequate technology have caused the multi join query optimization (MJQO) problem to remain unsolved. The problems of MJQO include the large size of queries (measure of numbers in join relation), high processing cost and long query execution time. Using swarm intelligent approaches as method to solve (MJQO) problem is practical and provide benefits for DBMS. This paper gives out a Swarm intelligence (Bees Algorithm) method towards the optimization of DBMS queries, And the propose method find Reasonable solution more efficiency than PSO algorithm ,which fastest convergence rate among all known solution for MJQO .It reduces the response time of query processing. Simulation shows proposed algorithm can solve MJQO problem in less amount of time than particle swarm optimization (PSO).**

Keywords- ***Artificial bees colony, Multi Join Query Optimization; Query Execution Plan; Query Execution Time; Database Management system; particle swarm optimization (PSO).***

I. INTRODUCTION

Nowadays, Multi-Join Query optimization has garnered considerable attention in Database management system, it important technique for design and implement (DBMS) and it's deceive factor effect the capability of database (DB). In this study propose one of swarm intelligent approaches (bee algorithm) that simulates the forging behavior of honey bee swarm to solve (MJQO) problem and Exploit performance of bee algorithm to compute time and simulate the result. The problems of MJQO include the large size of queries (measure of numbers in join relation), high processing cost and long query execution time. Therefore, it is very important to find an intelligent approach for this issue in order to help users to obtain desired information in a reasonable period of time.

Proposed genetic algorithm to search for the best possible plan from among the semantically equivalent plans that can be generated for any given query [1]. It therefore seems logical to consider query optimization in terms of search algorithms. Various search algorithms have been applied by researchers to find an optimal plan for query execution. Some recent researchers used different models to solve MJQO problem. However, they were not able to provide a better solution in terms of (query execution time) and (cost) and Traditional methods are not able to solve this optimization problem effectively because of the increased size of data and the large number of tables [2]. Deterministic algorithms, Greedy algorithms and heuristic algorithm based approaches have tried to approximate the optimum solution but their performance is not up to the mark [3]. This problem is then tried with genetic approaches and randomized approaches, such as tabu search, ant colony, bee colony [4] etc. which gave better performance [5]. But betterment in performance with improvement in quality of solution is still required. The implications of these criteria are important in order to increase speed of query and reduce the cost in DBMS. Therefore, a new intelligent approach like swarm intelligent approach (AB algorithm) with good performance with shorter query execution time (QET) and low cost is needed.

When a user inputs a query, it is first analyzed by parser for syntax errors, if there is no error it is then transformed into standard format i.e. a query graph [5]. Next, query Optimizer takes this query graph as input and prepares different query execution plans for that query and selects an optimal query.

II. MULTI JOIN QUERY OPTIMIZATION (MJQO)

Query optimization is the task of improving the strategy for processing a database query. It thus forms an important step in query processing. Query processing refers to the range of activities involved in extracting data from a database. These activities include translation of queries into expressions that can be implemented at the file system's level since these queries are submitted to the DBMS in a high level language, query optimization steps, transformations and query evaluation.Multi join Query optimization is a complex problem, not only in SQL server but in any other relational database system.

When a user input a query, it is first analyse by parser for syntax error, if there is no error it is then transformed in to standard format i.e. a query graph [5] .Next, query

optimizer take this query graph as input and prepare different query execution plane for that query and selects an optimal query execution plan amongst them, this optimal query plan is forwarded to query execution engine which evaluates it and returns the query result.

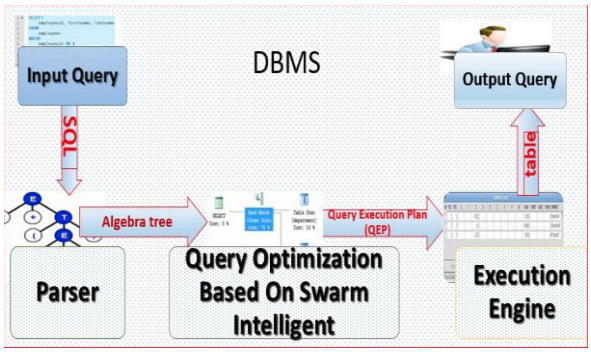

Figure 1: Query Evaluation

Individual queries are transformed in to relation algebra expression (algebra tree) and are represented as query graph. Then, query optimizer selects appropriate physical method to implement each relational algebra operation and finally generated query execution plane (QEP).

Amongst all equivalent QEP, optimizer choses the one with lowest cost output to the query execution engine, then, the query execution engine take the QEP, executes that plane, and return the answers to user. The process showed in Figure 1.

III. SEARCH SPACE

Characteristics In relational database systems each query execution plan can be represented by a processing tree where the leaf nodes are the base relations and the internal nodes represent operations. Different tree shapes have been Considered: left-deep tree, right-deep tree, and bushy tree. The Figur.2 explain tree structures of relational operators associated with the mlti-join query R1∞R2 ∞R3∞R4 ∞R4.

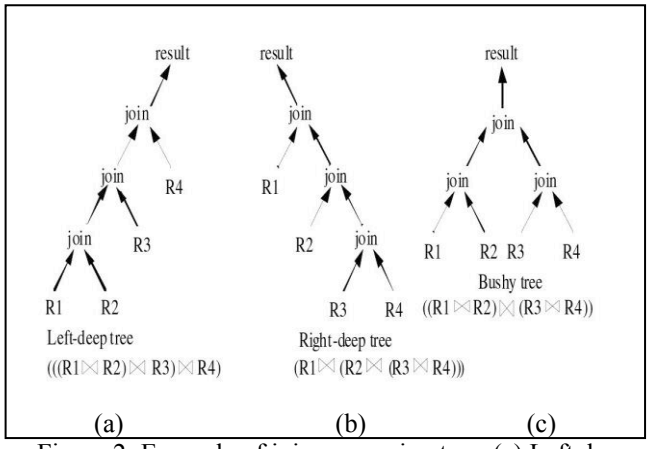

Figure 2: Example of join processing tree. (a) Left deep tree, (b) Right deep utree, (c) Bush deep tree

A search space can be restricted according to the nature of the execution plans and the applied search strategy. The nature of execution plans is determined according to two criteria: the shape of the tree structures (i.e. left-deep tree, right-deep tree and bushy tree) and the consideration of plans with Cartesian products.

For each join processing tree physical optimizer produces several operator trees by selecting a physical operator for a join operator [6]. In operator trees internal node is a physical operator i.e. an algorithm executes the join operator. Finally, the cost of each operator tree is estimated and the operator tree with lowest cost is selected as an optimal QEP. If it is assumed that all join operations are implemented by same physical method, than multi join optimization problem is simplified as finding the optimal join order which makes the cost lowest [7].

For any query graph there can be three possible join processing trees viz. left deep tree, right deep tree and bushy tree [5]. Five relations called R1, R2, R3, R4 and R5 are in a multiple join query Q. Figure 2 shows three possible join processing trees; a left deep tree (a), a bushy tree (b) and a right deep tree (c) of query Q. It categorized the search space further into three subspaces. The left deep tree can be considered as the subspace for MJQO problem. Left join process- ing tree can take the full advantage of index [7].

The solution space of the MJQO problem is the set of all possible join processing trees (i.e. Query Execution Plans) for a query graph. The goal is to find out the minimal cost join ordering tree in the mentioned solution space [8].

The queries with a large number of join predicates make the difficulty to manage associated search space which becomes too large. That is the reason why some authors chose to eliminate bushy trees. Each relation in query graph required parameters are: n(r): number of tuples in relation r; v (A, r): number of distinct of attribute as in relation r. The formula to calculate cost of a join processing tree is [9].

$$\text{Cost} = \sum_{i=1}^{n-1} n(t_i) \qquad (1)$$

For inner node t, if r and s are relations represented respect timely by left child and right child of t, and C is a common attribute group in relation r and s, then:

$$n(t) = \frac{n(r)+n(s)}{\prod \max(v(c_j,r),v(c_j,s))} \qquad (2)$$

n (t) is the size result relation of join operation of tow relation r and s; which is equal to the number of rows having similar values of attribute common in both relation, r and s .It is obtained by dividing the Cartesian product of

relations r and s by number of rows having distinct values of common attribute .In equation (2) n(r) x n(s) is the Cartesian product of relation r and s, which represent all combination over common attributes. $\Pi \max(v(c_j,r), v(c_j,s))$ Calculate multiplication of maximum distinct values of each common attribute (c_j) in r and s .Division of these two gives the total number of rows in the result relation of join operation between r and s.

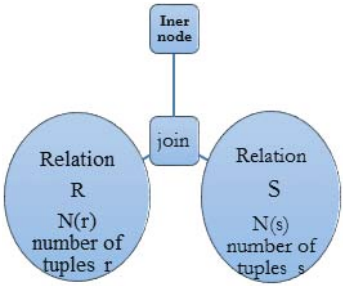

Figure 3: join operation between tow relation r and s.

The value of function v (A,t) which is used in Equation (2) can calculated by equation (3)[10].

$$V(A,t) = \begin{cases} v(A,r) & A \epsilon r - s \\ v(A,s) & A \epsilon s - r \\ \min(v(A,r), v(A,s)) & A \epsilon r \; A \epsilon S \end{cases} \quad (3)$$

V (A, t) is the number of distinct values of attributes A that appear in the relation t .in multi join queries intermediate space is very important because it is the space that decides the time to process that intermediate result . If the number of rows in intermediate result relation are more we require more time to evaluate this result in next step but if its size is small required less time.

The intermediate space is directly proportional to execution time of query .So if we can estimate the size of intermediate results, we can easily select the better QEP. Equation (2), (3) are used to compute the size (number of tuples) and number of distinct values for attributes of the inner node (intermediate result relation). The cost of join processing tree can be calculated by summing the cost of all intermediate nodes by using equation (1) so the cost estimating of a join tree consumes much computation time.

IV. SWARM INTELLIGENT APPROACH

Artificial Bee Colony (ABC) is one of the most recently defined algorithms by [11]. Motivated by the intelligent behavior of honey bees. This algorithm is based on two assumption:

- Attribute values in symmetrical distribution.
- The sum of the tuples number about intermediate results decides the cost of QEP .For example , t= r join s, C the public attribute over r,s .Then n(t) and v(A,T) are define by the (2) ;(3) formula .
- 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 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 [11,12,13,14,15,16,17,18,19,20] .

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 neighbourhood of its current food source and evaluates its nectar amount (fitness). The i th food source position is represented as Xi = (xi1, xi2, . .xid) . F(Xi) refers to the nectar amount of the food source located at Xi. After watching the dancing of employed bees, an onlooker bee goes to the region of food source at Xi by the probability pi defined as

$$p_i = \frac{f(x_i)}{\sum_{k=1}^{s} f(x_k)} \quad (4)$$

Where S is total number of food sources. The on- looker finds a neighbourhood food source in the vicinity of Xi by using

$$x_i(t+1) = x_i(t) + b_{ii} \quad (5)$$

Where b is the neihporhood patch size for j domination of I food source define as

$$b_{ij} = x_{ij} - x_{kj} \quad (6)$$

Where k is a random number ∈(1, 2, . . . ; S) and k ≠i, u is random uniform variant ∈[-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.(4). With this scheme, good food sources will get more onlookers than the bad ones. Each bee will search for better food.

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:

- When bee has decided to follow its preferred path, but there is only one nearby neighbourhood unvisited. It will move to unvisited relation.
- 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 .when a bee has decided not to follow its preferred path, but all nearby neighbourhoods have already been visited, in this case the bee will select the next relation based on the probability function (1).

$$I(i,j) = \frac{[m(i,j)][\frac{1}{n(i,j)}]^B}{\sum_{s=1,s\neq 1}^{n}[m(i,j)][\frac{1}{n(i,s)}]^B} \qquad (5)$$

- where I(i, j) probability in which the bee moves from relation (i) to(j) , h(i , j) distance between i , j relation , b positive parameter ,whose values the related importance of memory versus heuristic information, n the number of relations ,and I a list of all visited relation so far.
- When a bee has decided not to follow its preferred path and chooses a new nearby neighbourhood, in this case it will do the same as in rule 3.

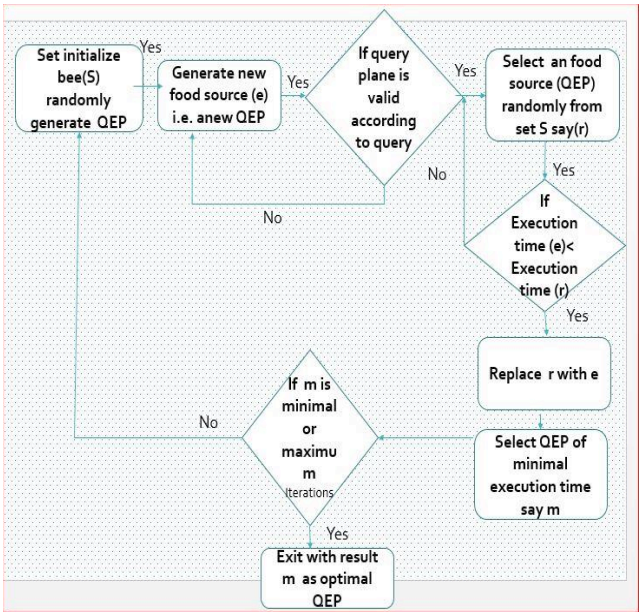

Figure 5. Proposed approach

V. PSEUDO CODE FOR BEES ALGORITHM

- Initialize
- REPEAT
- Move the employed bees onto their food source and evaluate the fitness
- Move the onlooker onto the food source and evaluate their fitness
- Move the scouts for searching new food source
- Until (termination criteria satisfied)

After analyzing the results of experiment this can be concluded that the proposed approach in this paper is more effective and efficient than PSO solution which is the best known solution till now. Proposed approach calculates optimal solution faster than PSO solution and also provides better quality of solution.

VI. Experimental results

In order to explain the effect of bees on MJQO to solving this problem experiment have been done on computer Pentium 5 2.40 GHz .generate database of 50 relation where each relation Cardinality in [10,110]. The relation cardinality is the number of tuples in a relation .The query categorized into ten sets of queries of different size (i.e. number of relation in query is of 5, 10, 15, 20, 25, 30, 35, 40, 45, 50). Every query made with an independent set of relation.

Table 1: Algorithm parameter

Bees	No of bees 10 No of iteration= No of relation
PSO	No of particle =5

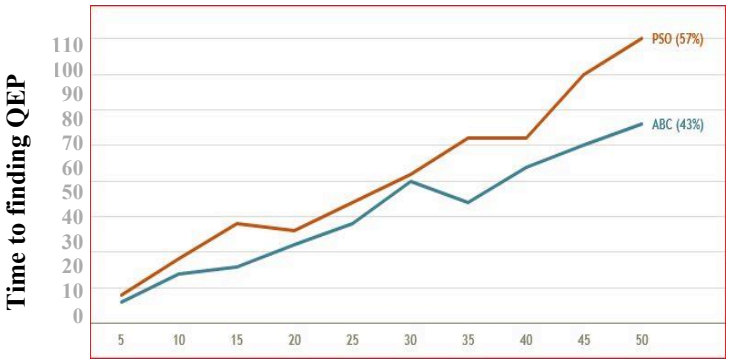

Figure 6:Comparisons of Execution Time

VI. CONCLUSION

Multi join query optimization useful and motivating research problem in the field of database .The propose method find Reasonable solution more efficiency than PSO algorithm, which fastest convergence rate among all known solution for MJQO.

It reduces the response time of query processing .Swarm intelligence (*Bees Algorithm)* towards the optimization of DBMS queries is still a novice field. There are still many opportunities to generate optimized solutions and to refine search strategies using of swarm intelligence algorithms for the Queries in RDBMS especially when the size and complexity of the relations increase with a number of parameters influencing the query.

The success of any database management system (DBMS) depends on how the query model is exploited. MJQO is very important in database research field. A good optimization algorithm not only improves the efficiency of queries but also reduces query execution time.

VII. REFERENCE

[1]. Chande, S.V. and M. Sinha,(2010)," Optimization of relational database queries using genetic algorithms". Proceedings of the International Conference on Data Management, IMT Ghaziabad.

[2]. Steinbrunn, M., Moerkotte, G., & Kemper, A. (1997). Heuristic and randomized optimization for the join ordering problem. The Very Large Data Bases Jour- nal, 6(3), 191–208. doi:10.1007/s007780050040.

[3]. M. Almery and A.Farahad (2012),"Application of bees algorithm in Multi Join Query Optimization," indexing and retrieval," in ACSIJ international journal of computer science ,vol.1,no.1, 2012.

[4]. Kadkhodaei, H., & Mahmoudi, F. (2011). A combination method for join ordering problem in relational databases using genetic algorithm and ant colony. In Proceedings of the 2011 IEEE International .

[5]. Dervis K,, Bahriye Akay,(2009)" A comparative study of Artificial Bee Colony algorithm" Applied Mathematics and Computation 214 108–132.

[6]. V. Tereshko and A. Loengarov,(2005) "Collective Decision-Making in Honey Bee Foraging Dynamics," School of Computing, University of Paisley, Paisley PA1 .2BE, Scotland, 2005.

[7]. Dong, H., & Liang, Y. (2007). Genetic algorithms for large join query optimization. In Proceedings of the 9th Annual Conference on Genetic Evolutionary Computation (GECCO '07) (pp. 1211–1218).

[8]. Li, N., Liu, Y., Dong, Y., & Gu, J. (2008). Application of ant colony optimization algorithm to multi-join query optimization. In L. Kang, Z. Cai, X. Yan, & Y. Liu (Eds.). In Proceedings of the 3rd International Symposium on Advances in Computation and Intel-ligence (ISICA '08) (pp. 189-197). Retrieved from http://www.springer.com/computer/information+sy stems+and+applications/book/978-3-540-92136-3.

[9]. Mukul J, Praveen S,2013" Query Optimization: An Intelligent Hybrid Approach using Cuckoo and Tabu Search" International Journal of Intelligent Information Technologies, 9(1), 40-55.

[10]. Alamery, M., Faraahi, A., Javadi, H., & Nouros- Sana, S. (2010). Multi-join query optimization using the bees algorithm. In Proceedings of the 7th International Symposium on Distributed Computing and Artificial Intelligence (pp. 449-457). Retrieved from http://www.springer.com/engineering/compu tational+intelligence+and+complexity/book/978- 3-642-14882-8.

[11]. Alamery, M., Faraahi, A., Javadi, H., & Nouros- Sana, S. (2010). Multi-join query optimization using the bees algorithm. In Proceedings of the 7th International Symposium on Distributed Computing and Artificial Intelligence (pp. 449-457). Retrieved from http://www.springer.com/engineering/compu tational+intelligence+and+complexity/book/978- 3-642-14882-8

[12]. Mukul J, Praveen S,2013" Query Optimization: An Intelligent Hybrid Approach using Cuckoo and Tabu Search" International Journal of Intelligent Information Technologies, 9(1), 40-55.

[13]. Steinbrunn, M., Moerkotte, G., & Kemper, A. (1997). Heuristic and randomized optimization for the join ordering problem. The Very Large Data Bases Jour- nal, 6(3), 191–208. doi:10.1007/s007780050040

[14]. Scientific & Engineering Research Volume 2, Issue 9, ISSN 2229-5518.

[15]. Pandao and. A. D. Isalkar, (2012). Multi query optimization using heuristic approach

[16]. S, V. Chande and M. snik , (200). Genetic Optimization for the Join Ordering Problem of Database Queries. Jaipur, India, Department of Computer Science International School of Informatics and Management.

[17]. M.pandao and A.Isalkar, (2012)"multi query optimization using heuristic approach, "international journal of computer science and network, ISSN 2277-5420, 2012.

[18]. E.Zafarni,"present new method for optimizing join queris processing in heterogeneous distributed database,"IEEE in knowledge discovery and data mining

[19]. B.souley and D.mohamed,"performing analysis of query optimizers under varity hardware component in RDBMS" in journal computer engineering and information technology, 2013.

[20]. T. Krink, B. and R. Thomsen, 2004 "Noisy optimization problems a particular challenge for differential evolution" in Proceedings of 2004 Congress on Evolutionary Computation, IEEE Press, Piscataway,NJ, 2004, pp. 332–339.

[21]. D.E. Goldberg, 1989," Genetic Algorithms in Search Optimization and Machine Learning", 0201157675Addison-Wesley Pub. Co. (1989).