An Enhanced Ant Colony-based Approach for Query Optimization

Hany A. Hanafy
ISSR, Cairo University
Computer Science Dept., ISSR, Cairo University, Giza, Egypt

Ahmed M. Gadallah
ISSR, Cairo University
Computer Science Dept., ISSR, Cairo University, Giza, Egypt

Hesham A. Hefny
ISSR, Cairo University
Computer Science Dept., ISSR, Cairo University, Giza, Egypt

ABSTRACT

One of the mandatory processes for all those types of applications is the inquiry process of the stored huge amounts of data. Such process is either a predefined or an ad-hoc query. From the logical point of view, the query process depends mainly on many algebraic operations, including selection, projection and joining operations. The most important one of them is the join operation, which represents the key factor of the inquiry process to retrieve the related information from different data tables. Many approaches have been proposed aiming to reduce the cost of join operations. Yet, there is still a need for more query optimizing processes in order to reduce the query response time. This paper proposes an enhanced optimal query processing approach for inner and outer join, where the proposed model exploits an adopted Ant Colony Optimization.

Keywords

Query Optimization, Ant Colony, Logical Optimizer, Query Access Plan, outer join.

1. INTRODUCTION

Efficient methods for query processing is one of the critical issues in the development of any efficient DBMS. Since most of the queries are complex, then the needs for a “query optimizer” is very essential to achieve an efficient query processing [14]. A complex query, which may contain a large group of relational tables, needs a long chain of join operations, which, is “equivalent to the query access plan”. This equivalent access plan needs a cumbersome and large computational effort. Therefore, to minimize the processing time of a query, we have to emphasize mainly on the minimization of processing efforts of that chain of join operations .In other words, we have to determine that join tree, which is equivalent to the best sequence of join operations. That such tree is called an optimal query access plan. Query optimizer is a part of the DBMS that is responsible for examining and assessing different alternatives of Query Access Plans(QAPs) for a given user query and choosing an efficient one [10].

The main objective of the present research work is to enhance the preprocessor for the determination of the optimal access plan of the outer join query based published research of Hany A. Hanafy and Ahmed M. Gadallah [6]. The processor is based on Ant Colony Optimization (ACO) and limited only to the logical optimization phase. The results of the published models are not sufficiently available, but for the sake of comparisons, the results of our enhanced model have been compared with that generated by SQL Server 2012.

This paper is organized as follows: section 2 explains the optimization process. The query access plan problem is given in section 3. Section 4 presents the proposed cost model. Section 5 discusses the proposed ACO algorithm approach for getting the optimal QAP. Section 6, illustrates the experimental work made to evaluate the proposed model. Finally, section 7 concludes the paper.

2. OPTIMIZATION PROCESS

We have to mention that, the query optimization step, [4] is the most critical step in query processing procedure [22], especially for complex queries, which often contains a large number of join operations [2]. Query optimization has two levels, the first level is the logical optimization and the second one concerns with physical optimization process [21].

The main task of the logical optimizer is to select the optimal access plan of the query [9]. The input of the logical optimization process is a query graph equivalent to the high-level declarative command such as SQL, and the output of the process is an optimal tree for the sequence of join operations derived from that given query graph [5]. Query graph, which represents the input of the logical optimization, consists of a set of nodes and edges as any conventional graph [7]. Each node represents one of the base relations and each edge represents the intersected common attributes of two connected nodes representing base relations. According to the commutative, equivalence and associative rules for the join operators, the logical optimizer generates firstly many different join trees, each of them is equivalent to the given query graph. For that such join tree, every base relation is represented by a leaf and the results of the join operation is represented by an inner node [3]. The assessment criteria of the logical optimizer depends mainly on the total cost of join operations included in any query access plan alternative.

On the other hand, the physical optimizer focuses on the optimal procedures to process join operations of the optimal query access tree [18]. The criterion of physical optimization is the input-output processing cost [1].

3. THE QUERY ACCESS PLAN PROBLEM

Join operation is one of the algebraic operations which needs a highly processing time [12]. Consequently, for a complex query, the processing of the query is affected mainly by highly execution effort of join operations [13]. This effort depends on the sequence of join operations of the different relations of a given query which is usually called the Query Access Plan (QAP) [15]. Therefore, the QAP problem is one of combinatorial problems where the search space contains all alternatives of QAP [19]. This type of combinatorial problem is usually represented by a graph structures which can be replaced by different alternatives of binary trees. The space contains all those binary trees represents the search space of the QAP problem, [16].

The present research is concerning mainly with the logical optimizer where the input is a query graph. Recently, many previous works are focused on algorithms of obtaining an optimal access plan of a complex query. These algorithms are Divided into four different categories: (a) Deterministic Algorithms, Every algorithm in this category creates a solution, systematically, in a deterministic way, either by applying a heuristic search technique [24], (b) Randomized Algorithms, which aim to find the state which solution gives the globally minimum cost. [25] (such as Iterative Improvement Algorithm [27], Simulated Annealing Algorithm (SA) [28], Two-Phase Optimization (2PO) [26], Toured Simulated Annealing [29], and Random Sampling [30]), (c) Genetic Algorithms, which inspire the biological evolution by using a randomized search technique, while looking for good problem solutions[31]. It begins basically with a random population and generates offspring by random crossover and mutation. The members that survive the subsequent selection are considered the “fittest” ones [32], and the next generation relies on them. And the algorithm reaches its end when there is no further improvement and the solution is represented by the fittest member of the last population [24], and (d) Hybrid algorithms, which combine the strategies of pure deterministic and pure randomized algorithms: solutions obtained by deterministic algorithms are used as starting points for randomized algorithms or as initial population members for genetic algorithms [16], [20].

In literature, reaching an optimal QAP requires achieving two consecutive operations. The first operation is concerned with constructing the QAPs search space, including a set of binary trees. Each binary tree is equivalent to a candidate QAP derived from the complete query graph of the relations involved in the query statement. Rationally, the complexity of a query statement is proportional to the cardinality of its search space. Commonly, the cardinality of the search space of a query statement with n involved relations is (n-1)! [20]. Almost, left-deep tree structure is used to represent QAPs in the search space. It consists of different levels of internal nodes. The lowest level of those internal nodes is derived by joining any chosen couple of base relations. On the other hand, any other internal node is derived by joining two nodes, the right node is one of the base relations and the left one is the just lower internal node in the query tree [20]. Consequently, the second operation is concerned with evaluating each candidate QAP according to a specific cost function in order to select a QAP with the lowest cost.

4. THE PROPOSED COST MODEL

Generally, to formulate a cost model for a QAP, we have to identify firstly the factors that affect the cost of a query. Those factors include: (a) The number of base relations, (b) the equivalent join tree, (c) the expected cardinality of each involved relation, and (d) the expected occurrence for each distinct value of the involved relations' attributes [3]. Some of these factors are known a priori, such as the number of involved relations and the structure of the join tree. In contrary, other factors are not known before processing such as the expected cardinality of each relation [11], and the expected occurrence frequency for each distinct value of each attribute. Commonly, the expected cardinality, number of tuples, of each leaf relation lies between the expected lower and upper bounds of that relation cardinality. For simplicity, the average of lower and upper bounds for each relation is taken as its expected cardinality. If the relation is an intermediate node, the cardinality of this relation generated out of the join operation may depend on the number of distinct values for each joining attribute that belongs to both of the two intersected relations. Actually, the number of distinct values of each attribute is not explicitly given. So, to estimate the number of distinct values, a probability distribution for the distinct values of each attribute may be used. The probability of each distinct value for a specific attribute depends on the occurrence frequency of this attribute. Suppose that, a specific attribute has N distinct values and the distribution of the distinct attribute values is a uniform one. Consequently, the probability of the occurrence of any distinct value of this attribute equals (1/N).Generally, the proposed cost model is based on both of the expected occurrence of each distinct attribute value which has been assumed as a uniform distribution and the number of tuples of the intermediate relations. Commonly, the cost of join operations involved in a complex query having m relations can be estimated as follows.

Assuming that t_i represents an inner node in the join tree for each *i* in [1, *m*-1] where *m* represents the involved number of relations in the join tree. Accordingly, the cost equation of any join tree equivalent to one of candidate QAP_j is given by equation (1):

$$\mathrm{Cost}\left(\mathrm{QAP}_j\right) = \sum_{i=1}^{m-1} \mathrm{n}(\mathrm{t_i}) \qquad \textbf{(1)}$$

Where, $n(t_i)$ is the expected number of tuples in relation t_i which may be a leaf relation or an intermediate relation in the join tree corresponding to a candidate QAP_j.

On the other hand, the length of any intermediate relation *t* is *n(t)*, where the relation *t* represents the result of the joint operation between the two involved relations *(r, s)*. Such length can be computed using equation (2).

$$\mathrm{n(t)} = (\text{ Cartesian product of relations r and s}) * \text{selectivity factor} \qquad \textbf{(2)}$$

In other words, the expected length *n(t)* can be given as shown in equation (3).

$$\mathrm{n(t)} = \frac{\boldsymbol{n(r)*n(s)}}{\prod_{c_j \in c} \mathbf{max}\,(v(c_j,r), v(c_j,s))} \qquad \textbf{(3)}$$

Where, *n(r)* and *n(s)* represent the count of tuples in relations *r* and *s* respectively, $v(c_j,r)$ and $v(c_j,s)$ represent the number of distinct values of each joining attribute c_j in relations *r* and *s* respectively and *c* re-presents the set of joining attributes.

As an exceptional case, if there is no joining or common attributes between relations *r* and *s* then *t* will represent the Cartesian product of relations *r* and *s*. Consequently the join selectivity factor becomes equal to 1. Thus, equation (3) will be reduced to as depicted in equation (4).

$$\mathrm{n(t)} = \mathrm{n(r)} * \mathrm{n(s)} \qquad \textbf{(4)}$$

The estimated number of distinct values for each attribute *A* in the resulted relation *t* from a join operation between two relations *r* and *s*is given by equation (5).

$$\mathrm{V(A,t)} = \begin{cases} \mathrm{V(A,r)}: \mathrm{A} \in \mathrm{r} - \mathrm{s} \\ \mathrm{V(A,s)}: \mathrm{A} \in \mathrm{s} - \mathrm{r} \\ \min\left(\mathrm{V(A,r)}, \mathrm{V(A,s)}\right): \mathrm{A} \in \mathrm{r} \text{ and } A \in s \end{cases} \qquad \textbf{(5)}$$

Where *V(A,r)*and *V(A, s)* are the estimated count of distinct values for attribute *A* in relations *r* and *s* respectively.

As described in equation (5), if an attribute *A* belongs to both relations *r* and *s* then the estimated number of its distinct values in the resulted relation *t* will be the minimum number of distinct values of attribute *A* in both *r* and *s*.

5. THE PROPOSED ACO FOR OPTIMAL QAP PREPROCESSOR

The enhanced proposed query optimization approach aims to reach an optimal query access plan from all candidate query access plans for a given query statement. It represents an ant colony-based optimization approach. Commonly, the collective and cooperative behavior of ACO generating from the interaction of different search threads makes it effective in solving combinatorial optimization problems [23]. In fact, an ACO algorithm uses a set of artificial ants (individuals) which transact to reach the solution of a given problem by exchanging information via pheromone deposited on graph edges [17]. The ACO algorithm is employed to imitate the behavior of real ants and it can be represented as depicted in Algorithm 1[8].

Algorithm 1: Ant colony optimization.

```
Initialize
Loop
Each ant is positioned on a starting node
  Loop
     Each ant applies a state transition rule to incrementally
        build a solution and a Local pheromone updating
        rule
     Until all ants have built a complete solution
     A global pheromone updating rule is applied
Until reach to optimal solution
End.
```

The proposed approach to estimate the optimal QAP by ACO has the following steps:

1. Compute the selectivity factor as shown in equation (6).

$$\eta_{ij} = \frac{1}{d_{ij}} \qquad \textbf{(6)}$$

Where, d_{ij}represents the number of tuples resulted from joining relations i and j.

2. Calculate the cardinality of the resulted relation from joining relations *i* and *j* using equation (7).

$$d_{ij} = \frac{n(i)*n(j)}{\pi_{C_r \in C} \max(V(C_r,i),V(C_r,j))} \qquad \textbf{(7)}$$

Where *v(c*$v(c_r, i)$ indicates the number of distinct values of attribute c_r in the relation i.

3. Compute the new pheromone value$\tau_{ij}$of the inter node representing the result of joiningrelations*i and j* using equation (8).

$$\tau_{ij} = (1 - P) * \tau_{ij} + \sum_{k=1}^{m} \Delta\, \tau_{ij}^{k} \qquad \textbf{(8)}$$

Where, *P* indicates the evaporation rate, m is the number of ants and $\Delta\, \tau_{ij}^{k}$ represents the amount of pheromone change made by ant k.

4. Consequently, equation (9) is used to compute the quantity of pheromone$\Delta\, \tau_{ij}^{k}$for a specific ant k.

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k \quad \text{if ant k uses edj(i, j) for} \\ \quad \text{relations i and j in its tour} \\ 0 \qquad\qquad\qquad \text{otherwise} \end{cases} \qquad \textbf{(9)}$$

Where, *Q*is a constant, and$L_k$is the count of tuples (effort) of the joining tour constructed by ant *k*.

5. Finally, equation (10) is used to calculate the probability $P_{ij}^{k}$of joining relation j with relation *i* by ant *k*.

$$P_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} * \eta_{ij}^{\beta}}{\sum_{C_{ij} \in N(\ _sP)} \tau_{ij}^{\alpha} . \eta_{ij}^{\beta}} & \text{if } C_{ij} \in N(\ _sP) \\ 0 & \text{otherwise} \end{cases} \qquad \textbf{(10)}$$

Where,$N(s^p)$is the set of feasible access plan edge *(i, L)* where *L* is the entities not yet visited by ant *k*, and both parameters α and β control the relative importance of the pheromone versus the heuristic information η_{ij} .

6. EXPERIMENTAL RESULTS

The results of the enhanced proposed approach are compared with that generated by SQL Server 2012.The enhanced proposed cost function is taken as a factor for evaluation and comparison. The software tools that are used in this experiments are SQL Server 2012 enterprise edition, visual C#.net 2012.Northwind database is used in the case study which is a sample database in SQL Server that contains the sales data for a fictitious company that imports and exports food items around the world. The comparative criterion that is used in the evaluation is based on the formula that is used in the cost function. This criterion represents the cost of the selected QAP and which is computed using equation (1). The evaluation is based on the cost function as a comparative criterion. To generate estimated optimal query access plan in SQL Server 2012, we depend on the output of the utility "Display Estimated Execution Plan" that is included in SQL Query Analyzer tool. The used cases are given in table (2). Table (3) includes the cases and the results.

Table 1. Index of relations

Index	Relation	Index	Relation
1	Orders	7	Shippers
2	Orderdetails	8	Region
3	Employees	9	Products
4	Customers	10	Territories
5	Suppliers	11	EmployeeTerritories
6	Categories		

As shown in table (2), it can be noted that in the cases 4, [6-10], 11, [14-17], 19, [20-23] and 25, the proposed query optimization approach has lower cost than SQL Server 2012 query optimizer. That is the relative cost of the proposed approach to SQL Server 2012 is between 0.38 (in case 4) and 1.03(in case 5).While such relative cost in the cases13,18, 20 and 24 the proposed query optimizer has a very little increase than SQL Server optimizer by just 0.01. Also, there are five cases namely 1, 2, 3, 7 and 12 have the same processing costs. On the other hand, the average of relative average cost of the proposed approach to SQL server 2012 equals83% which indicates a reduction in processing cost by around 17%. Also from the results shown in table (3), it can be noted that, the overall variability range of the proposed optimizer is lower than the overall variability range of SQL Server optimizer. On the other hand, it is clear that the overall mean cost of the proposed optimizer is lower than that in SQL Server 2012 optimizer, and the variability range of the proposed approach is lower than the variability range of SQL Server.

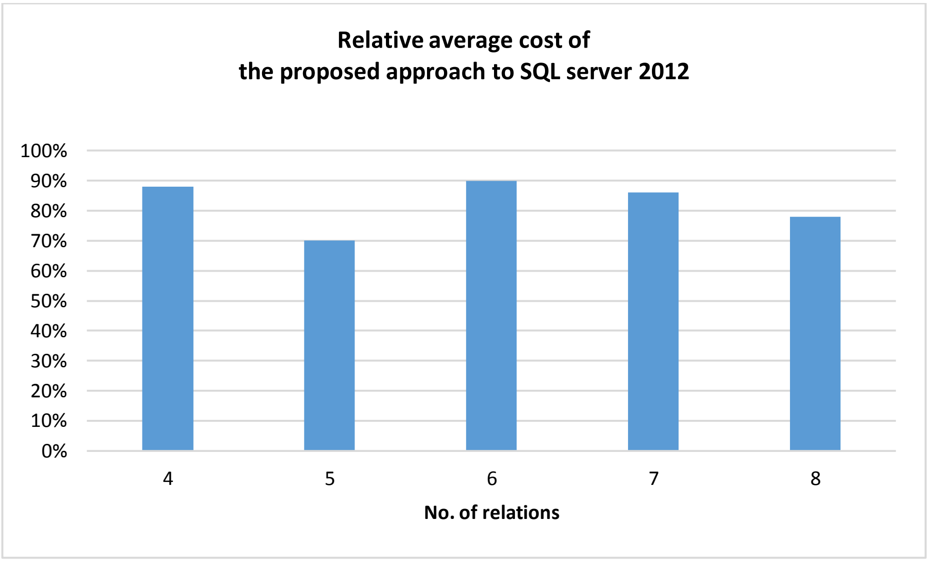
Table 2. Comparison of between the proposed optimizer and SQL Server optimizer

Case No.	No. of relations	Sequence of inserted relations in the corresponding case	Estimated path by SQL Server 2012	Cost of estimated path in SQL Server 2012	Estimated path by the proposed approach	Cost of estimated path in the proposed approach	Relative cost between the proposed approach and SQL Server 2012	Relative average cost of the proposed approach to SQL server 2012
1		3/2/1/9	1/3/2/9	9300	1/3/2/9	9300	1	
2		4/3/2/1	3/1/4/2	6600	4/1/3/2	6600	1	
3	4	6/2/1/9	6/9/2/1	8080	6/9/2/1	8080	1	88%
4		8/2/1/3	8/2/1/3	108000	3/1/2/8	41300	0.38	
5		3/4/1/7	7/1/3/4	2600	7/3/1/4	2670	1.03	
6		3/11/2/1/9	2/1/9/3/11	26285	1/3/2/9/11	23585	0.90	
7		4/3/2/1/9	1/3/4/2/9	10600	1/3/4/2/9	10600	1	
8		5/6/2/1/9	2/1/9/5/6	16000	9/6/5/2/1	8160	0.51	70%
9	5	4/7/2/1/9	2/1/9/4/7	16000	4/7/1/2/9	10000	0.63	
10		4/8/2/1/9	1/8/4/2/9	95400	1/4/2/9/8	45300	0.47	
11		4/3/11/2/1/9	1/2/9/4/3/11	30285	1/4/3/2/9/11	24885	0.82	
12		5/4/3/2/1/9	1/3/4/2/9/5	14600	1/3/4/2/9/5	14600	1	
13	6	5/4/6/2/1/9	6/9/5/2/1/4	12160	5/6/9/2/1/4	12350	1.01	90%
14		4/7/6/2/1/9	2/1/9/4/6/7	20000	1/4/7/2/9/6	14600	0.73	
15		5/4/8/2/1/9	2/1/9/4/5/8	52000	9/5/2/1/4/8	48080	0.92	
16		6/4/3/11/2/1/9	6/9/2/1/3/11/4	40650	1/4/2/9/6/3/11	31585	0.78	
17		5/4/3/11/2/1/9	2/1/9/4/5/3/11	34285	1/4/2/9/5/3/11	31585	0.92	
18	7	5/4/3/6/2/1/9	9/6/5/2/1/3/4	16160	6/5/9/2/1/3/4	16350	1.01	86%
19		5/4/7/6/2/1/9	2/1/9/4/5/6/7	24000	6/5/9/2/1/7/4	16350	0.68	
20		5/4/8/6/2/1/9	2/1/9/4/5/6/8	56000	9/6/2/1/5/4/8	52080	0.93	
21		6/4/3/11/2/1/9/5	9/6/5/2/1/3/11/	40730	4/3/1/2/9/6/5/11	33545	0.82	
22		5/4/3/11/2/1/9/7	1/2/9/4/5/3/11/	48570	4/3/1/2/9/5/7/11	33545	0.69	
23	8	5/6/3/11/2/1/9/7	1/2/9/5/3/11/6/	58855	9/5/6/2/1/7/3/11	30445	0.52	78%
24		5/4/7/6/2/1/9/3	1/7/3/4/2/9/6/5	19900	9/5/6/2/1/3/4/7	20160	1.01	
25		5/4/8/6/2/1/9/3	2/1/9/4/5/3/6/8	60000	9/5/6/2/1/3/4/8	52160	0.87	

Table 3. Some statistical results extracted from the above results

No. of relations	Mean		Median		Variability	
	The Proposed approach	SQL Server 2012	The Proposed Approach	SQL Server 2012	The Proposed approach	SQL Server 2012
4	13590	26916	8080	8080	2670 : 41300	2600 : 108000
5	19529	32857	10600	16000	8160 : 45300	10600 : 95400
6	22903	25809	14600	20000	12350 : 48080	12160 : 52000
7	29590	34219	31585	34285	16350 : 52080	16160 : 56000
8	33971	45611	33545	48570	20160 : 52160	19900 : 60000
Average	23916.6	33082.4	20160	24000	2670 : 52160	2600 : 108000

Figure 1. Comparison of between the proposed optimizer and SQL Server optimizer



7. CONCLUSION

The query optimization problem is one of big problems in the area of database management systems. Query optimization efforts depends mainly on two items; the first one is the plan or the strategy to access different relations involved in the query, and the other item is the algebraic relational operations required to execute that query. The cost of processing join operations dominates cost of other relational algebraic operations. Therefore our enhanced proposed logical optimization model emphasize basically on how to determine the optimal query access plan and focusing only on inner and outer join operations. In essence the optimal query access plan is a combinatorial problem where the search space of the problem includes all alternatives access plans. As many combinatorial problems, the query itself can be illustrated by an equivalent graph. Its nodes are standing as the involved relations, and every edge connects any two nodes represents the common attributes in the two relations introduced by the two nodes of the edge. The enhanced proposed model transforms the graph into a right binary tree which is called join tree, Therefore the search space of the optimization problem consists of different candidates of join tree. An enhanced ant colony optimization model is exploited, as a random search procedure, to determine the optimal join tree equivalent to the optimal access strategy. For the sake of simplicity the model exploits the unconstrained graph type to represent a given query. The model is applied on many experimental cases to check the validity and correctness of the model. A comparative study has been developed between the results of the proposed model and those optimal access plans resulted by using the SQL SERVER DBMS, and most of the results were in favor of the proposed model. The enhanced model can be extended to cover also the physical aspects of the query processing. It is worthy to study the relative effectiveness of using the constrained graph type rather than the unconstrained one used in the enhanced proposed model.

8. FUTURE WORK

The proposed approach can be extended to consider the physical optimizer rather than the logical one. Also, it is worthy to study the relative effectiveness of using the constrained graph type rather than the unconstrained one used in the proposed approach.

9. REFERENCES

[1] Chaudhuri, S.: An Overview of Query Optimization in Relational Systems, ACM, pp. 34-43 (1998).

[2] Doka,K., Tsoumakos,D. ,Koziris,N.: Online querying of d-dimensional hierarchies. J. Parallel Distrib. Comput. 71, 424–437 (2011).

[3] Dong ,H. and Liang,Y.: Genetic Algorithms for Large Join Query Optimization, ACM,pp.1211-1218(2007).

[4] Elmasri ,R.andNavathe,S.B: Fundamentals of database systems, third edition, Addison Wesley(2000).

[5] Galindo-Legaria, C. and Rosenthal, A .: Query Graphs, Implementing Trees, and Freely Reorderable Outer joins, In Proc. of ACM SIGMOD, Atlantic City,pp.291-299(1990).

[6] Hany A. Hanafy and Ahmed M. Gadallah.:Ant Colony-Based Approach for Query Optimization, DMBD 2016, LNCS 9714, pp. 425–433, (2016).

[7] Haritsa,J.: Query Optimizer Plan Diagrams: Production, Reduction and Applications, Database Systems Lab, Indian Institute of Science, ICDE Conference, 1374-1377(2011).

[8] Hlaing,Z. and Khine,A.: Solving Traveling Salesman Problem by Using Improved Ant Colony Optimization Algorithm, International Journal of Information and Education Technology, Vol. 1, No. 5,404-409, December (2011).

[9] Jarke, M. and Koch, J.: Query Optimization in Database Systems. ACM,Computing Surveys, Vol. 16, No. 2,pp.111- 152, June (1984).

[10] Jin,L. and Li,C.: Selectivity Estimation for Fuzzy String Predicates in Large Data Sets, Proceedings of the 31st VLD Conference,Trondheim, Norway, pp.397-408(2005).

[11] Kabra,N.,DeWitt,D.J.: Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans, ACM SIGMOD ,98 Seattle, WA, USA , PP.106-117(1998).

[12] Kabra,N.,DeWitt,D.J.: Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans, ACM SIGMOD ,98 Seattle, WA, USA , PP.106-117 (1998).

[13] Krynicki,K. and Jean,J.: AntElements: An Extensible and Scalable Ant Colony Optimization Middleware, GECCO '15,1109-1116, July 11 - 15, Madrid, Spain (2015).

[14] Li,Z.,Ross,K.A.: Fast joins using join indices, The VLDB Journal,Vol 8, PP.1-24 (1999).

[15] Mahajan,S. and Jadhav,V.: An Analysis of Execution Plans in Query Optimization, International Conference on Communication, Information & Computing Technology (ICCICT), Oct. 19-20, Mumbai, India, 1-5 (2012).

[16] Mahmoud,F.,Shaban,S.,Abd El-Naby,H.: A proposed Query Optimizer Based on Genetic Algorithms,The Egyptian Computer Journal,Vol.37,No.1(2010).

[17] Mavrovouniotis,M., Müller,F,Yang,S.: An Ant Colony Optimization Based Memetic Algorithm for the Dynamic Travelling Salesman Problem, GECCO '15, 49-56, July 11 - 16, Madrid, Spain (2015)

[18] Mishra P, Eich, MH.: Join processing in relational databases. ACM Computing Surveys Vol. 24, pp.63–113 (1992).

[19] Rahman,S. , Ahsan Feroz,A.M. , Kamruzzaman,M. , Faruque,M.: Analyze Database Optimization Techniques, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.8, August (2010).

[20] Steinbrunn ,M., Moerkotte,G., Kemper,A.: Heuristic and randomized optimization for the join ordering problem, The VLDB Journal Vol.6, PP.191–208 (1997).

[21] Youssefi, K., Wong, E.: Query processing in a relational database management system, In ProcConf VLDB, Rio de Janeiro, Brazil, PP.409-417 (1979).

[22] Yu,P.S.,Cornell,D.W.: Buffer Management Based on Return on Consumption In a Multi-Query Environment, VLDB Journal,Vol2, PP.1-37(1993).

[23] Yu,X.,Chen,W.,Zhang,J.: A Set-based Comprehensive Learning Particle Swarm Optimization with Decomposition for Multiobjective Traveling Salesman Problem, GECCO '15, 89-96, July 11 – 15, Madrid, Spain (2015).

[24] Selinger,P.G., Astrahan,M.M., Chamberlin,D.D., Lorie,R.A., Price,T.G.: Access Path Selection in a Relational Database Management System, ACM.Inc.(1979).

[25] Favaretto, D., Moretti,E., Pellegrini,P.: An Ant Colony System Approach for Variants of the Traveling Salesman Problem with Time Windows, Journal of Information & Optimization Sciences Vol.27, No.1, PP.35-54(2006).

[26] Ioannidis,Y.E., Kang,Y.C.: Randomized Algorithms for Optimizing Large Join Queries, ACM(1990).

[27] Swami,A.: Optimization of Large Join Queries: Combining Heuristics and Combinatorial Techniques, ACM, SIGMOD Conference,PP. 367-376 (1989).

[28] Ioannidis,Y.E., Wong,E.: Query Optimization by Simulated Annealing, ACM, SIGMOD Conference, PP.9-22(1987).

[29] Lanzelotte,R., Valduries, P., Zait,M.: On the Effectiveness of Optimization Search Strategies for Parallel Execution Spaces. In proceedings of the Conference on Very Large Databases, PP.493-504(1993).

[30] Galindo-Legaria,C., Pellenkoft,A., Kersten,M.: Fast, Randomized Join-Order Selection Why Use Transformations?. In proceedings of the 20th International Conference on Very Large Databases, PP.85-95(1994).

[31] Choi,I-C., Kim,S-I., Kim,H-S.: A Genetic Algorithm with a Mixed Region Search for the Asymmetric Traveling Salesman Problem, Computers & Operations Research 30, PP.773-786(2003).

[32] Bennett,K., Ferris,M.,C., Ioannidis,Y.,E.: A Genetic Algorithm for Database Query Optimization, Copmuter Science Technical Report #1004(1991).