IJCTA, 9(21), 2016, pp. 243-249

Efficient Query Optimization : A novel approach for generating optimal query plans using Iterative Improvement and Simulated Annealing

Pramod Kumar Yadav* SAM Rizvi**

Abstract **:** The collection of various sites, distributed over a computer network is called Distributed Database whereas the technique of finding the optimal processing method to answer a query is called Query Optimization. In Distributed Database, the sites communicate with each other through networks. There are various issues arise during evaluation of query cost, among which the processing cost and a transmission cost are the important. There are several algorithm developed to find the best possible solution for a particular query, but they all have their certain limitations. The primary objective of distributed query plan is to generate the query plan that reduces the quantity of data transfer between sites and also the distributed query response time. In distributed database system the data is scattered over various multiple sites and a single relation may be present in more than one sites. In such case the cost of query processing is effected by various cost involved such as optimization cost, communication cost, local processing cost, query localization cost etc .The optimizer is mainly concern on the cost model, search space, and the search strategy. It primarily focuses on these three factors. The query cost depends on the order of evaluation of the operators, for the same query we can have different cost if the order is changed. Hence, to find the optimal cost for a particular query is emerging as an open challenge for many researchers. Therefore the cost-based query optimization technique has emerged as an important concept for dealing with the query optimization. In this paper, the optimal query plan is generated by applying the concept of iterative improvement algorithm and simulated annealing algorithm and then the comparative study of both results are done using same heuristic to generate the optimal query plans.

Keyword : Distributed Query Optimization, Simulated Annealing, Iterative Improvement, Distributed Database, Cost based Optimization.

1. INTRODUCTION

Query Optimization has become a big challenge in today's era due to the rapid growth in the size of databases from gigabytes to pentabytes and so on. It has also become a difficult task because the databases are scattered over different distributed sites. The process of finding best optimal cost of the query plan among these distributed sites is a big challenge [1]. There are several factors which influence in evaluating the cost of a query, some of them are the optimization cost, communication cost, local processing cost, query localization cost etc[2] . The cost is also affected by the number of sites participating in the communication. The lesser number of sites involve will incurred low cost where as large number of sites would leads to higher cost .Query Optimizer evaluates the amount of memory needed for the execution of the query. It also takes the decision regarding which operation has to be executed at which site. In order to do so it selects the best query plan. The cost incurred in case of distributed query processing are the cost of CPU, Input/Output cost and the cost involved in establishing communication between different sites, out of these the cost incurred in intra-site communication is important[5]. The cost of

* KIET Group of Institutions, Ghaziabad, UP, India pramodyadavster@gmail.com

** Jamia Millia Islamia, New Delhi, India samsam_rizvi@yahoo.com

processing a distributed query would be reduced if the numbers of sites involved in intra-site communication are less otherwise it will increase. The data may be required from several sites in order to process a distributed query. The number of optimal query plans would increase as the numbers of sites containing the relations are increased [18]. Thus it is very difficult to generate optimal cost for a given query. Thus there is a need of involving a large search space comprising of all possible plans in order to compute the best optimal query plan [2]. The cost of the query will also depend on the search strategy applied. Some strategy produces good results when there are lesser number of sites, where as some produces large number of query plans when there are large number of sites [7].

1.1. Search Space

The search space consists of the collection of all possible query execution plans. It primarily focuses on finding the optimal query execution plan. The query execution plans developed by the search space produces the similar results after execution [4]. The query tree is used to represents the solution of the plan for executing the join expression and cost is associated with the every point of search. The major role of cost function is to maps the query tree to their respective costs [7]. Due to the large number of relations required for processing a distributed query, searching an optimal query plan is difficult, which ultimately increases the search cost. This problem can be solved by generating plans that minimizes the cost of the search [10].

1.2. Search Strategy

The main task of the search strategies is to find the execution plans in an efficient and cost effective manner. For most of the search strategies the Dynamic programming forms the base [1]. There are basically two approaches which can be used to solve the problems related to search strategy [2]. The first and foremost approach is deterministic approach which builds a plan on base relations and then join it with one or more relation at each step until the complete plan is obtained. The partial plan that may not produce the optimal plans is pruned [1] [2], due to which the reduction in the optimization cost is achieved. The second approach which is referred to as randomized strategy is implemented. In this, it is mainly concern on finding the optimal solution around some particular point, though it does not guarantee the optimal solution. Randomized strategy does not cost much for optimizing the optimal plans in terms of memory and time consumption [6]. The working tradition of randomized strategy is that it starts with some randomly selected plans and then tries to find the neighbor plan and then compare the cost of plan with neighbor plan. If the cost of the neighbor plans is less than the starting plan, then the neighbor plan is consider as a solution plan [10]. This phenomenon is continues till it finds a plan which has no neighbor having lesser cost, for a pre-defined number of neighbors. Constant space overhead is one of the most important advantages of random strategy [10].

2. QUERY OPTIMIZATION METHOD

Query Optimization is emerging as one of the important area of research, as the size of the databases increases from terabyte to zeta byte. The need of various optimization techniques is on high demand. Optimization plays a vital role in finding the solution from large databases [1] [5]. With the help of query optimization it would be possible to find optimal execution plans, with minimum cost. In this technique the set of all possible query execution plans are taken into consideration by the optimizer and the cost of the query plan is evaluated using cost function. There are several important algorithms which are used to find the optimal cost of the query like Genetic Algorithm, Randomized Based Algorithm, Greedy Algorithm, Dynamic Programming, and Iterative Dynamic Programming. Iterative improvement and Simulated Annealing are the two important algorithm based on the concept of the Randomized Algorithm[3]. These algorithms are widely used for providing the solution of Query optimization in Distributed environment. The concept of distributed query optimization primarily deals with generating the best optimal cost query plan[10].

2.1. Cost function

The data in distributed environment are scattered among very distributed sites. Now when the user runs a query in SQL, the data may be residing at various distributed site. The cost of the query processing will be low if

the data would reside at closer in the distributed environment, otherwise the cost of query processing would be high, if the data reside at far distance[7]. On the other hand, if the data required to answer user query resides in disparate sites, then the query processing efficiency would be low as it would require joining data from each site to generate the result. The closeness is defined in terms of number of sites participating to answer the query. If the number of sites are less than query processing will be more efficient and if the number of sites participating is more than the query processing will be less efficient[10][8]. To understand the closeness, which determines the efficiency of query, let us consider the relation and site matrix shown in Table 1

Table 1.

Query Relation	***Distributed Sites***			
QR1	10	16	17	19
QR2	10	16	18	12
QR3	10	15	18	14
QR4	10	13	14	12

From Table 1, it can be observed that relation QR1 is on site 10, 16, 17 and 19. The relation QR2 is on the sites 10, 16, 18, and 12. The relation QR3 is on the sites 10, 15, 18 and 14. The relation QR4 is on the sites 10, 13, 14 and 12.

Suppose the user query UQ in SQL is as given below

SELECT QR1.B, QR2.C
FROM QR1, QR2, QR3, QR4
WHERE QR1.A = QR3.A and QR2.D = QR4.D

In the above query, it can be seen that there are four query relations involve in the FROM clause. Suppose the query optimizer is required to generate a query plan that contains these relations are at the same time optimal [8][10]. It can be seen that the relation QR1 is on sites 10, 16, 17 and 19, relation QR2 is on the sites 10, 16, 18, and 12, relation QR3 is on the sites 10, 15, 18 and14 and relation QR4 is on the sites 10, 13, 14 and 12. The primary objective is to generate the optimal query plan that reduces the cost of query processing [18].

For a query, a query plan denotes the sites for the relations in the SQL query [8]. For example, the query plan given below implies that for the user query given above, QR1 is at site 16, QR2 is at site 18, QR3 is at site 14 and QR4 is at site 12. The query plan is considered valid if the sites given in the plan contain the corresponding relations [10].

The query plan given below is a valid query plan for UQ.

16	18	14	12

It can be seen from table 1 that relation QR1 is in four sites and relation QR2 is in four sites, relation QR3 is in four sites in the same way QR4 is in four site[10]. There are in all 4 * 4 * 4 * 4 =2 56 valid query plans for UQ. If the number of sites containing the relations, which are in the from clause of SQL, increases than the number of valid plans would also increase[8]. The primary objective is to find optimal query plans among all possible query plans for User Query. Let us consider the valid query plans given in table 2[8].

Table 2

	Query Plans			
QP1	1	1	1	1
QP2	1	1	1	3
QP3	1	1	4	4
QP4	7	8	1	1
QP5	9	2	5	3

QP1 have all the four relations in the same site *i.e.* 1. This is the most optimal query plan as all the relations are in the same site [8]. In QP2, the first three relations are in site 1 and relation 4 is site 3. It means there are two sites participating. Also in QP3, there are two sites participating where relation R1 and relation R2 are on site 1 and relation R3 and relation R4 is on site 4. So in QP2 and QP3 two sites are participating now the one which has higher concentration of relation among them would be considered optimal. QP2 will be considered as optimal because the concentration of relation on a site is more as compared that of QP3, as in case of VQ2 site 1 is having three relation and whereas in VQ3 site 1 is having only two relation. So the concentration of VQ2 is more than VQ3.

This heuristic is defined by a cost function given below

$$\text{Query Processing Cost} = \sum_{i=1}^{M} \frac{Si}{N}\left(1 - \frac{Si}{N}\right)$$

Where the symbol M, Si & N denotes number *f* distributed sites, number of times ith sites and number of relation accessed respectively.

The query processing cost is computed for the query plans given in table 2. These are given in table 3.

Table 3

	Query Plans	***QPC***
QP1	1 | 1 | 1 | 1	4/4(1–4/4) = 0
QP2	1 | 1 | 1 | 3	3/4(1–3/4) + 1/4(1–1/4) = 6/16
QP3	1 | 1 | 4 | 4	2/4(1–2/4) + 2/4(1–2/4) = 8/16
QP4	7 | 8 | 1 | 1	2/4(1–2/4) + 1/4(1–1/4) + 1/4(1–1/4) = 10/16
QP5	9 | 2 | 5 | 3	1/4(1–1/4) + 1/4(1–1/4) + 1/4(1–1/4) + 1/4(1–1/4) = 12/16

From table 3, it is noted that QP1 has the minimum QPC followed by QP2, QP3, QP4 and QP5 has the maximum QPC. The aim is to generate query plans with lower Query Processing Cost [8][10].

The iterative improvement and simulated annealing techniques are used to generate such optimal query plans. The iterative improvement technique is discussed in the next section followed by simulated annealing in the subsequent section

3. ITERATIVE IMPROVEMENT

The general iterative improvement algorithm is given in table 4.The concepts of iterative improvement algorithm is basically based on applying two loops[10] .The inner loop is primarily used for the process of local optimization. In case of Local optimization, the algorithm first selects a random state from the vast search space and then improves the solution iteratively until it reaches the state of local minimum [8]. This process of improving the solution iteratively is continued till it reaches the stopping situation. After the stopping condition is achieved, then the algorithm returns the minimum cost attained by that distributed sites [10]. If there is a chance of infinite time then the algorithm would attain global minimum, which is depended on the cost of the function and its neighboring function [10]

Table 4

```
Procedure Iterative Improvement ( )
{
MinState = State„
While not (stopping condition) do
{
State = random state,
While not (local minimum (State)) do
{
State' = random state in neighbors (State),
If cost (State') < cost (State) then State = State';
}
If cost (State) < cost(minState) then minState = State,
}
Return (minState),
}
```

The concept of Iterative improvement algorithm was initially used to solve the travelling sales problem. This algorithm primarily utilizes the concept of local optimization and begins with a random state and enhances the solution by choosing randomly downhill until it reaches the local minimum[3]. This process of local optimization continues till termination condition is met, which can be explain by either the Query Plan Constraint or the Time Constraint [19][10].

4. SIMULATED ANNEALING

The Iterative improvement algorithm has its own major drawbacks. The first and foremost drawback is that it performs local optimization at downhill moves, and this drawback arise when there are large number of starting points with unpredictable results *i.e*. solution space contains large number of high cost local minima. Simulated annealing algorithm is an enhancement of iterative improvement algorithm that removes this drawback. It is a better technique for problems having discrete search space[6]. The primary objective of this technique is to find the good solution in a fixed amount of time inspite of the best possible solution. It acknowledges uphill moves with some probability, trying to avoid being trapped in a high cost [19][10]. The Simulated Annealing algorithm is discussed in table 5. The uphill moves are accepted with some probability. The algorithms also take care of the situation where it does not get stuck in local minima state [10]. The uphill moves are being control by the inner loop. It also does the local optimization. The Simulated Annealing algorithm controls the probability by the temperature which is fixed for each stage in the inner loop. Here the inner loop represents the stage. The reduction function is used to reduce the temperature. In every stage the uphill moves is determined by the differences in the cost of new state and the original state and the temperature[12]. The processing of inner loop stops when it reaches the equilibrium states. This algo would reaches the states of global minimum when the temperature is approaching towards zero. [10].

Table 5

```
Procedure Simulated Annealing ()
{
State = State0,
Temp = Temp0,
MinState = State,
WHILE NOT (frozen) do
```

{
WHILE NOT (equilibrium) DO
{
State′ = random state in neighbors (State),
"C=cost (State¢) – cost (State),
IF ("C d" 0) THEN State = State′
IF ("C e" 0) THENS = State′ with probability $\frac{-\Delta_c}{e^{\mathrm{T}}}$
IF cost (State) < cost (minState) THEN minState = State,
}
Temp = reduce (Temp)
}
RETURN (minState),
}

5. EXPERIMENT RESULTS

The Iterative Improvement algorithm and simulated annealing algorithm uses the concept of randomized algorithm. The comparison of both algorithms are done on two factors, the number of optimal query plans generated by the algorithm and the average Query Processing Cost attained by these generated optimal query plans. The comparison is based on changing the time from 1 to 9 second by step of 1 second. However the results of these experiments show that large numbers of query plans are generated using iterative improvement as compared to simulated annealing. This indicates more local minimas are attained as compared to simulated annealing, where as simulated annealing, as compared to iterative improvement, can generate better optimal query plans. The concept of Two-Phase query optimization can also be use to generate the best optimal query plans as the two-phase optimization algorithm is a combination of both iterative improvement and Simulated annealing. In this first iterative improvement algorithm is used followed by the simulated annealing.

6. CONCLUSION

After applying the concepts of the Iterative Improvement algorithm and simulated annealing algorithm we came to conclusion that Iterative Improvement algorithm generates large number of query plans where as Simulated annealing algorithm generates better plans. We can also use the concepts of two-phase query optimization algorithm which is the combination of applying iterative improvement algorithm followed by simulated annealing algorithm, which would generates more better optimal query plan with low cost. Hence in future we can apply the combination of these algorithms to find the best optimal query plan in distributed database.

7. REFERENCES

1. Yadav, Pramod Kumar, Rizvi, S.A.M *"Query Optimization: Issues and Challenges in Mining of Distributed Data"*, publish in Springer, March, 2016.
2. Vikash Mishra and Vikram Singh "*Generating Optimal Query Plans for Distributed Query Processing using Teacher-Learner Based Optimization"*, Elsevier Procedia Computer Science 54 (2015) 281 – 290.
3. Donald kossmann and konrad stocker, *"Iterative Dynamic Programming: A New Class of Query Optimization Algorithms*", ACM Transactions on Database Systems, Vol. 25, No. 1, March 2000, Pages 43–82.
4. SURAJIT CHAUDHURI, KYUSEOK," *Optimization of Queries with User Defined Predicates*", ACM Transactions on Database Systems, Vol. 24, No. 2, June 1999, Pages 177–228.

5. FRAGKISKOS PENTARIS and YANNIS IOANNIDIS, *"Query Optimization in Distributed Networks of Autonomous Database Systems*", ACM Transactions on Database Systems, Vol. 31, No. 2, June 2006, Pages 537–583.

6. P. E. DRENICK and E. J. SMITH," *Stochastic Query optimization in Distributed Database*", ACM Transactions on Database Systems, Vol. 18, No. 2, June 1993, Pages 262-288.

7. DONALD KOSSMANN," *The State of the Art in Distributed Query Processing*", ACM Computing Surveys, Vol. 32, No. 4, December 2000, pp. 422–469.

8. T.V. Vijay Kumar, Vikram Singh and Ajay Kumar Verma,"*Distributed Query Processing Plans Generation using Genetic Algorithm*", International Journal of Computer Theory and Engineering, Vol.3, No.1, February, 2011, 1793-8201.

9. BEE-CHUNG CHEN and RAGHU RAMAKRISHNAN."*Bellwether Analysis: Searching for Cost-Effective Query-Defined Predictors in Large Databases*", ACM Transactions on Knowledge Discovery from Data, Vol. 3, No. 1, Article 5, Publication date: March 2009.

10. Giri, A. K., and R. Kumar. "Distributed query processing plan generation using iterative improvement and simulated annealing", 2013,3rd IEEE International Advance Computing Conference (IACC), 2013.

11. Michael S. Lew, Nicu Sebe, Chabane Djeraba, Ramesh Jain, "*Content-based Multimedia Information Retrieval: State of the Art and Challenges*", In ACM Transactions on Multimedia Computing, Communications, and Applications, Feb. 2006.

12. Rajvardhan Patil1, Zhengxin Chen1 and Yong Shi, "*Database Keyword Search: A Perspective from Optimization*",2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, pages 30-33.

13. Raymond Kosala, Hendrik Blockeel, "*Web Mining Research: A Survey"*, 2000 ACM SIGKDD, July 2000. Volume 2, Issue 1 - page 1-15.

14. Shu-Hsien Liao , Pei-Hui Chu, Pei-Yuan Hsiao, "*Data mining techniques and applications – A decade review from 2000 to 2011"*, 2012 Elsevier Expert Systems with Applications 39 (2012) 11303–11311.

15. Amold W M Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, Ramesh Jain,"*Content-Based Image Retrieval at the End of the Early Years*",in IEEE Transactions on pattern Analysis and Machine Intelligence, Vol 22,No:12, December.2000.

16. Manuel Barrena, Elena Jurado, Pablo Márquez-Neila, Carlos Pachón,*" a flexible framework to ease nearest neighbor search in multidimensional data spaces", in Elsevier journal on Data & Knowledge Engineering 69 (2010) 116–136*.

17. Gultekin Ozsoyoglu, Ismail Sengor Altingovde, AbdullahAal-hamdani, Selma Ayse Ozel and Ozgur Ulusoy, Zehra Meral Ozsoyoglu, *"Querying Web Metadata: Native Score Management and Text Support in Databases"*, ACM Transactions on Database Systems, Vol. 29, No. 4, December 2004, Pages 581–634.

18. Kumar, T. V. Vijay, Biri Arun, and Lokendra Kumar. "Distributed Query Plan Generation Using HBMO",Lecture Notes in Computer Science, 2013.

19. Jagjit Bhatia. "Analytical Evaluation of Different Query Optimization Techniques", The International Journal Of Science & Technoledge (ISSN 2321 – 919X) ,Vol 3 Issue 3 March, 2015.