American Journal of Information Systems, 2014, Vol. 2, No. 3, 52-55
Available online at http://pubs.sciepub.com/ajis/2/3/2

DOI:10.12691/ajis-2-3-2

Estimating Plans along with Cost in Multiple Query Processing Environments by Applying Particle Swarm Optimization Technique

Sambit Kumar Mishra[1,*], Srikanta Pattnaik[2], Dulu patnaik[3]

[1]Deaprtment of Computer Sc.&Engg., Ajay Binay Institute of Technology, Cuttack
[2]S.O.A. University, Bhubaneswar
[3]Government College of Engineering, Bhawanipatna
*Corresponding author: sambit_pr@rediffmail.com

Received December 12, 2014; Revised December 25, 2014; Accepted December 29, 2014

Abstract The Main idea of multiple query processing is to optimize a set of queries together and execute the common operations once. Major tasks in multiple query processing are common operation or expression identification and global execution plan construction. Query plans are generally derived from registered continuous queries. They are composed of operators, which perform the actual data processing, queries which buffer data as it moves between operators to hold state of operators. The complex part is to decompose queries and query plans and rearrange the sub queries and query plans on the network. The main functions to achieve an optimal query distribution are usually minimizing network usage and minimizing response time of queries. While dealing with query distribution problem, the challenges like modeling topology of the network, decomposing queries into some sub queries and sub query placement may be occurred. Operators are the basic data processing units in a query plan. An operator takes one or more streams as input and produces a stream as output. As in the traditional database management system, a plan for query connects a set of operators in a tree. The output of a child operator forms an input of its parent operator. In this paper it is aimed to retrieve the cost of query plans as well as cost of particles of swarm in multiple query processing environments by applying particle swarm optimization techniques.

Keywords: *query plan, swarm, NP hard, particle, SMT, personal best, global best*

Cite This Article: Sambit Kumar Mishra, Srikanta Pattnaik, and Dulu patnaik, "Estimating Plans along with Cost in Multiple Query Processing Environments by Applying Particle Swarm Optimization Technique." *American Journal of Information Systems*, vol. 2, no. 3 (2014): 52-55. doi: 10.12691/ajis-2-3-2.

1. Introduction

Complex queries being in common place usually have a lot of common sub-expressions, either within a single query, or across multiple such queries run as a batch.

Multi query processing aims at exploiting common sub-expressions to reduce evaluation cost. Multi query processing has been viewed as impractical, since earlier algorithms were exhaustive, and explore a doubly exponential search space.

Distributing on operators among a number of hosts is a NP hard problem. Particle swarm optimization and genetic algorithm may be used to compare them to each other. Encoding is a mapping from knowledge domain to the solution space where algorithm can process. Particle swarm optimization is a stochastic optimization technique. The algorithm is initialized with a population of random solutions. Each solution represents a particle. All particles move based on personal best and global best in the search space to find the best solution that ith particle has experienced so far. The algorithm repeats until a threshold is reached or it finds the optimal solution. In fact, after each iteration, the position of each particle updates with the velocity vector. Velocity vector is calculated based on personal best and global best. The basic particle swarm optimization is suitable to solving continuous problems. The particle swarm optimization algorithm first generates initial random particles and then assigns each particle to its personal best. After that it assigns the best personal best to the global best. The particle swarm optimization calculates the fitness value of each particle and then updates personal best and global best.

2. Review of Literature

Y.E. Ioannidis et.al [1] have suggested in their paper that most of the queries on relational databases require access to relations from multiple sites for their processing. The number of possible alternative query plans increases exponentially with increase in the number of relations required for processing the query.

M. Jarke et.al [2] have discussed in their paper that exploring all the query plans in this large search space, e.g. exhaustive search, is not feasible. This problem in large databases is a combinatorial optimization problem and has

been addressed by techniques like simulated annealing, iterative improvement, two-phase optimization, etc. The techniques, which reduce the search space, are based on plan transformation and have a cost model to assess the quality of query processing plans.

L. P. Mahalingam et.al [3] have discussed in their paper that the optimization strategy based on algebraic equivalences between similarity based operations that serve as rewrite rules is outlined in. Optimization rules based on similarity based algebraic framework properties and equivalence laws.

Ch. Li, Kevin et.al [4] have introduced a novel multi-criteria query optimization techniques for performing query optimization in databases, such as multimedia and web databases, which rely on imperfect access mechanisms and top-k predicates. They have also introduced cost model and optimization algorithms.

Stefan Riezler et.al [5] have implemented query processing using natural language, e.g. plain English, and for understanding English by the machine, statistical machine translation (SMT). This approach is to bridge the lexical gap between questions and answers. SMT-based query expansion is done by i) using a full-sentence para-phraser to introduce synonyms in context of the entire query, and ii) by translating query terms into answer terms using a full-sentence SMT model trained on question-answer pairs.

Raymond T. Ng et.al [6] have focused on logic programming in deductive databases. They have also extended deductive databases with probabilisties and given fixed point semantics to logic programs annotated with probabilities, but they have used absolute ignorance to combine event probabilities.

Norbert Fuhr et.al [7] have introduced a method for evaluating queries on probabilistic databases is to use complex events. They have also reviewed its limitations. Start by expressing q as a query plan, using the operators σ,π,×. Then modify each operator to compute the event attribute E in each intermediate result: denote σi,_ π i,×i the modified operators. It is more convenient to introduce them in the functional representation, by defining the complex event ep(t) for each tuple t, inductively on the query plan p.

Praveen Seshadri et.al [8] have focused on decorrelation techniques. The use of the decorrelation techniques results in the query being transformed to a set of queries, with temporary relations being created. In this manner, the queries generated by decorrelation may have several subexpressions in common, and are therefore excellent candidates for multi-query optimization.

Subbu N. et.al [9] have implemented the correlated evaluation of queries because it may be more efficient on the query, and may not be possible to get an efficient decorrelated query using standard relational operations. In correlated evaluation, the nested query is repeatedly invoked with different values for correlation variables.

A. Pérez-Uribe et.al [10] have focused on optimization problems e.g. swarm intelligence, which is inspired by the social behavior of some insects such as ants and bees. Honey-bees mating optimization (HBMO) is a swarm intelligence optimization algorithm that models the behaviors of bees. Honeybees algorithms were used to model agent-base systems.

K. Bennett et.al [11] have experimented using PSO and genetic algorithm and found that PSO as well as genetic algorithms could be elegantly useful to optimize database query plans.

We compared honey-bees, DPSO and genetic algorithm with each other and with centralized algorithm for each scenario. The centralized method reveals the effect of distribution.

M.J. Franklin et.al [12] have focused about high fan-in query plans. They have used low fan-in query plans. They have also focused on how the fitness value of each distribution algorithm changes over time and the honey-bees algorithm does not get trapped in local minima.

A. Sokolov et.al. [13] have Used the QPC values of the query plans in their experimental analysis, and found that fitter query plans are selected using the unbiased tournament selection technique. The selected query plans undergo crossover, with probability Pc, and mutation, with probability Pm, to generate the population for the next generation. This continues until the algorithm runs for a pre-specified number of generations GP. The top-query plans are then generated based on the QPC values.

T.V. VijayKumar et.al [14] have proposed an approach that uses Genetic Algorithm to generate 'close' query plans. The approach aims to generate query plans that are optimal with respect to the number of sites involved, and the concentration of relations in these sites, for answering the user query. This in turn would result in efficient query processing.

3. Algorithm

1. Initiate random particles and assign each particle to its personal best. Find the Initial global best.
2. While have enough time, calculate fitness of particles
3. For each particle, update personal best, global best
4. For each particle do these steps, e.g. update its position, return global best particle as solution

This algorithm may be simplified and elaborated as following.

1. Assign the size of swarm, for example in this case it is set to 10.
2. Allocate maximum query e.g. in this case 100.
3. Cognitive parameter, c1=1.
4. Social parameter, c2, is set to 4-c1.
5. Number of relations is set to 2.
6. Number of optimization variables, npar is set to 2.
7. Generate random population of continuous values and update the particle position
8. Evaluate random population of continuous variables from allocated queries & relations.
9. Generate random velocities by considering size of swarm and number of optimization variables.
10. Evaluate the cost of particle by considering CPU time, population of continuous values and velocities of the particles.
11. Evaluate the cost of swarm by considering the cost of particles along with CPU time.
12. Update the best local position for each particle.

4. Experimental Analysis

Consider two relations EMP and ASG where attributes to the relation EMP are ENo, EName, Title and attributes to the relation ASG are ENo, PNo, Resp, Dur.

If it is asked to find the names of employees who are managing a project, the query may be written as

SELECT EName
FROM EMP,ASG
WHERE EMP.ENo = ASG.ENo AND Dur > 37

Two possible transformations of the query may be represented in the following expressions.

Expression 1:

∏EName(σDur>37∩EMP.ENo=ASG.ENo(EMP × ASG))

Expression 2: ∏EName(EMP ⋊ENO (σDur>37(ASG)))

Expression 2 avoids the expensive and large intermediate Cartesian product, and therefore typically is better.

Usually data may be horizontally) fragmented.

For example,

Site1: ASG1 = σENo≤"E3"(ASG)
Site2: ASG2 = σENo>"E3"(ASG)
Site3: EMP1 = σENo≤"E3"(EMP)
Site4: EMP2 = σENo>"E3"(EMP)
Site5: Result

Relations ASG and EMP may also be fragmented in the same way.

Relations ASG and EMP may also be locally clustered on attributes Resp and ENo, respectively.

Now consider the expression ∏EName(EMP ⋊ENO (σDur>37(ASG)))

Tuples are uniformly distributed to the fragments; 20 tuples satisfy Dur>37

size(EMP) = 400, size(ASG) = 1000

tuple access cost = 1 unit; tuple transfer cost = 10 units

ASG and EMP have a local index on Dur and ENo

For example**,** Produce ASG's: (10+10) * tuple access cost =20

Transfer ASG's to the sites of EMPs: (10+10) * tuple transfer cost =200

Produce EMP's: (10+10) * tuple access cost * 2 =40

Transfer EMP's to result site: (10+10) * tuple transfer cost =200

Total cost= 460

Query processing is done in the following sequence: (1) query decomposition, (2) data localization, (3)global optimization, (4) local optimization.

Table 4.1. Cost of Particle and Swarm(Relation R1, Relation R2)

Sl.No.	Relation (R1)		Relation(R2)	
	Cost of particle	Cost of swarm	Cost of particle	Cost of swarm
1	60.817	121.52	60.164	120.87
2	61.387	122.09	61.414	122.12
3	61.543	122.25	60.288	120.99
4	60.992	121.69	61.291	121.99
5	60.83	121.53	61.77	122.47
6	61.066	121.77	61.66	122.26
7	61.529	122.23	60.304	121.01
8	59.764	120.47	61.139	121.84
9	60.554	121.26	61.835	122.54
10	61.18	121.88	60.86	121.56

Query optimization is a crucial and difficult part of the overall query processing. The objective of query optimization is to minimize the following cost function: I/O cost + CPU cost + communication cost.

Size of the swarm = 10;

No. of relations e.g. dimension of the problem,= 2;

Maximum number of iterations e.g. maximum no. of query = 100;

No. of optimization variables, npar=2;

Cognitive parameter, c1 = 1;

Social parameter, c2 = 4-c1;

Constriction factor, C=1;

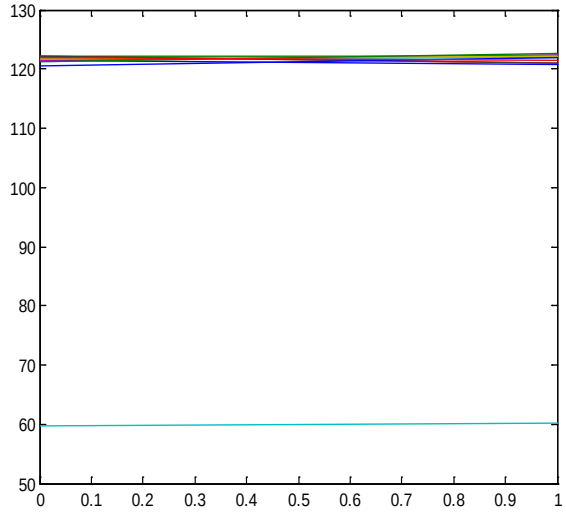

Figure 4.1. (Plan generation VS cost of swarm)

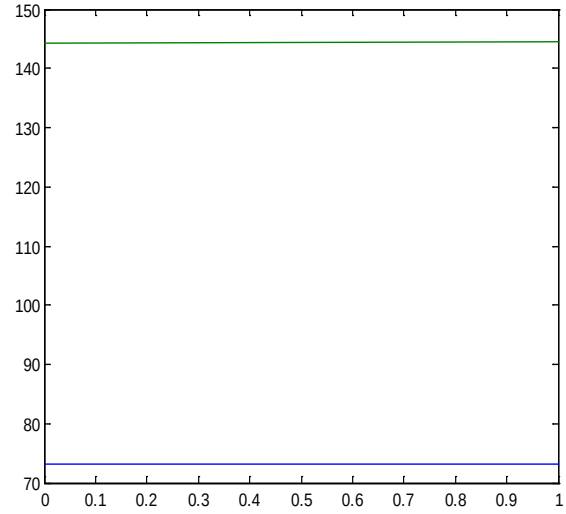

Figure 4.2. (Plan generation VS average cost of particle of swarm)

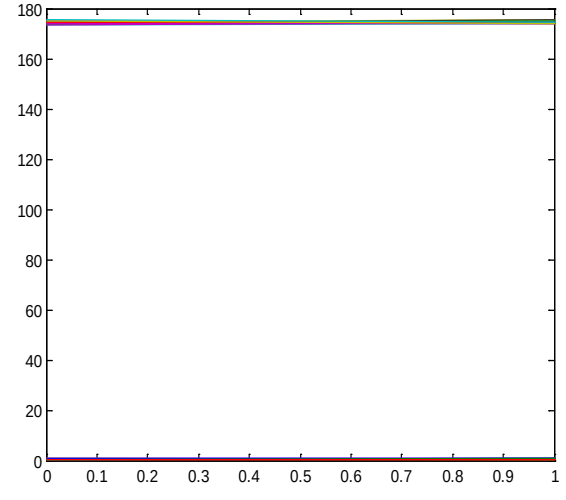

Figure 4.3. (Plan genetaion VS particle velocity)

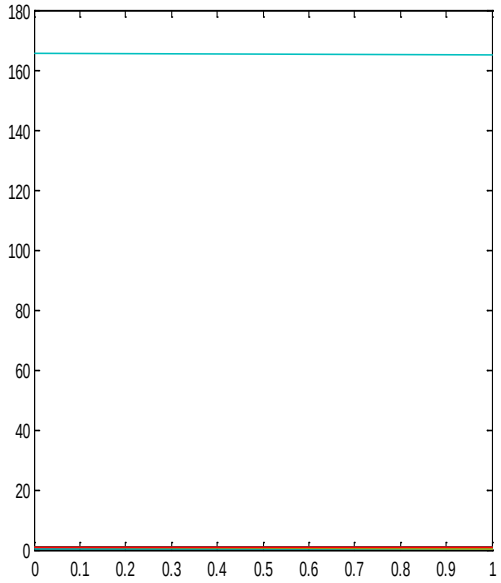

Figure 4.4. (Plan generation VS average cost of particle along with velocity)

5. Discussion & Future Direction

Usually data may be required to process the user query that may be spread over various locations with heterogeneity. So there may be a need to arrive at a query processing plan that entails an optimal cost for query processing. The query plans generated for the particular set of queries generally posed on databases distributed across various locations. The efficiency of query processing depends upon the closeness of the required data. Many query plans may be generated for a given query from multiple relations at various locations. So there may be a number of combinations of relations at various locations for query processing. As a result it may generate a quite large number of query plans. Among these query plans, optimal query plans may be identified having the required relations. It may be a complex problem if the number of relations accessed by the query is quite huge in number and each of the relations may have multiple copies across various locations. In the experimental evaluation it is seen that that the query plans generated are directly proportional to particles in the swarm as well as cost of swarm.

6. Conclusion

In this paper the query plans are generated to improve the response time of user queries. It is usually achieved by formulating the distributed query processing plan generation as a single-objective algorithm problem. It was also aimed to generate query plans with the desired data, for answering the user queries residing close to each other. It is found that the query plans generated are directly proportional to particles in the swarm as well as cost of swarm.

References

[1] Y.E. Ioannidis and Y.C. Kang, “Randomized algorithms for optimizing large join queries, ACM 1990.

[2] M. Jarke and J. Koch, “Query optimization in database systems,” ACM Computing Surveys, volume 16, no. 2, pp. 111-152, June 1984.

[3] L. P. Mahalingam and K. S. Candan, Multi-Criteria Query Optimization in the Presence of Result Size and Quality Tradeoffs, Multimedia Tools and Applications Journal 23(3) (2004), 167-183.

[4] Ch. Li, Kevin Ch.-Ch. Chang, I. F. Ilyas, and S. Song, RankSQL: query algebra and optimization for relational top-k queries. In: F. Ozcan, editor, SIGMOD Conference. ACM, 2005, 131-142.

[5] Stefan Riezler, Statistical Machine Translation for Query Expansion in Answer Retrieval, Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 464-471, Prague, Czech Republic, June 2007.

[6] Raymond T. Ng and V. S. Subrahmanian. Probabilistic logic programming. Information and Computation, 101(2):150-201, 1992.

[7] Norbert Fuhr and Thomas Rolleke. A probabilistic relational algebra for the integration of information retrieval and database systems. ACM Trans. Inf. Syst., 15(1):32-66, 1997.

[8] Praveen Seshadri, Hamid Pirahesh, and T. Y. Cliff Leung. Complex query decorrelation. In Intl. Conf. on Data Engineering, 1996.

[9] Subbu N. Subramanian and Shivakumar Venkataraman. Cost based optimization of decision support queries using transient views. In SIGMOD Intl. Conf. on Management of Data, Seattle, WA, 1998.

[10] A. Pérez-Uribe and B. Hirsbrunner,―Learning and foraging in robot-bees‖, in Meyer, Berthoz, Floreano, Roitblat andWilson (eds.)’, SAB2000 Proceedings Supplement Book, Intermit. Soc. For Adaptive Behavior, Honolulu, Hawaii, pp. 185-194.

[11] K. Bennett, M.C. Ferris, and Y.E. Ioannidis, ―A genetic algorithm for database query optimization‖, In Proc. of the 4th International Conference on Genetic Algorithms, 400-407, 1991.

[12] M.J. Franklin, S.R. Jeffery, S. Krishnamurthy, F. Reiss, S. Rizvi, et al., ―Design considerations for high fan-in systems: the HiFi approach‖, In Proc. Of the CIDR Conf., Jan. 2005.

[13] A. Sokolov and D. Whitley, “Unbiased Tournament Selection,” in proceedings of the 2005 conference on Genetic and Evolutionary Computation, pp. 1131-1138, 2005

[14] T.V. VijayKumar, Vikram Singh and Ajay Kumar Verma, “Generating Distributed Query Processing Plans using Genetic Algorithm”, In the proceedings of the International Conference on Data Storage and Data Engineering (DSDE 2010), Bangalore, February 9-10, 2010, pp. 173-177, 2010.