STUDIA UNIV. "BABEŞ-BOLYAI", INFORMATICA, Volume XLIII, Number 2, 1998

STOCHASTIC OPTIMIZATION FOR JOIN OF THREE RELATIONS IN DISTRIBUTED DATABASES I. THE THEORY AND ONE APPLICATION

VIORICA VARGA

ABSTRACT. The query optimization problem for a single query in a distributed database system was treated in great detail in the literature. Most of the articles search for a deterministic strategy assigning the component joins of a relational query to the processors of a network that can execute the join efficient and determine an economic strategy for the data transferring. For each new type of query that arrives at the system, a new optimal strategy is determined. A distributed system can receives different types of queries and processes them at the same time. In this case the determination of the optimal query processing strategy is a stochastic optimization problem. Query processing strategies may be distributed over the processors of a network as probability distributions. The stochastic query optimization problem was solved for a single-join and a multiple join of three relations, where the operand relations are stored at two sites. In this article we treat the join of three relations, which are stored at three different sites. This problem leads to a nonlinear programming problem, which is solved in this article. We intend to solve the general problem of the sequential and the parallel operation for two queries of the specified type, in a next article, these problems leads to the same type of nonlinear programming problem.

1. INTRODUCTION

Many algorithms were elaborated for minimizing the costs necessary to perform a single, isolated query in a distributed database system. The capacity of distributed systems for concurrent processing motivates the distribution of a database in a network. There is a different approach to query optimization if the system is viewed as one, which receives different types of queries at different times and process more than one query at the same time. The multiple-query problem is not deterministic; the multiple-query input stream constitutes a stochastic process. The strategy for executing the multiple-query is distributed over the sites of the network as a probability distribution. The "decision variables" of the stochastic query optimization problem are the probabilities that a component operator of the query is executed at a particular site of the network.

In [4] the authors extend the state-transition model proposed by Lafortune and Wong [5] and the original multiprocessing model of [3] and [2]. The main objective of the model is to give query-processing strategies, which are globally optimal.

Defintion 1. *xxx*

In this article in section 3 we present the stochastic model for the join of three relations, which are stored at three different sites. This stochastic query optimization problem leads to a nonlinear programming problem, which is specific one. In the section 2 we give an algorithm to solve this nonlinear programming problem. In the next article we will present general models, containing sequential and parallel operation of two queries of type treated in section 3. These leads to the same type of nonlinear programming problem as the problem from section 3, which we solve for different cases and the results are presented in tables. From the examples we can see the global optimality of the stochastic query optimization model.

2. The nonlinear programming problem

Let $f_1, f_2, \ldots, f_p : [0,1]^n \to \mathbb{R}_+$ be strictly positive polynomial functions and let $q_1, \cdots, q_n : \mathbb{R}^n \to \mathbb{R}$ linear functionals of form $q_i(x_1, \cdots, x_n) = \sum_{j=1}^{n} a_{ij} x_j$, where $a_{ij} \in \{0,1\}$ and $i = \overline{1,r}$ We consider the next problem:

$$\begin{aligned} & f_1(x_1, \cdots, x_n) \leq y \\ & f_2(x_1, \cdots, x_n) \leq y \\ & \quad \vdots \qquad\qquad (\mathrm{P}) \\ & f_p(x_1, \cdots, x_n) \leq y \\ & q_1(x_1, \cdots, x_n) = 1 \\ & \quad \vdots \\ & q_r(x_1, \cdots, x_n) = 1 \\ & \min y,\ y \in \mathbb{R},\ y > 0. \end{aligned}$$

In the following we state two results, with aid we solve the nonlinear programming problems in the next sections. In the rest of this section we use the notation $x = (x_1, \cdots, x_n)$.

Theorem 2.1: *The problem (P) has at least one solution.*

Proof: We consider the set $X = [0,1]^n \cap q_1^{-1}(1) \cap \cdots \cap q_r^{-1}(1)$. Because the set $[0,1]^n$ is compact subset of $\mathbb{R}^n$ we get that the set X is a compact subset of $\mathbb{R}^n$ because the functions $q_1, \cdots, q_r$ are continuous.

Let $f : X \to R$ be the function defined by $f(x) = \max\{f_1(x), f_2(x), \ldots, f_p(x)\}$. Because the function f is continuous and X is a compact metric space, using the Weierstrass theorem, then there exists a point $x_0 \in X$ such that $f(x_0) = \min_{x \in X} f(x)$. We prove that $f(x_0) = \min y$. We suppose, that exists $y_0 \in R_+^*$ such that $f(x_0) > y_0$ and y_0 satisfied the inequalities from the problem (P) for $x_0^* \in X$, i.e.:

$$f_1(x_0^*) \leq y_0$$
$$f_2(x_0^*) \leq y_0$$
$$\vdots$$
$$f_p(x_0^*) \leq y_0$$

We obtain $f(x_0^*) = \max\{f_1(x_0^*), f_2(x_0^*), \ldots, f_p(x_0^*)\} \leq y_0 < f(x_0)$ which contradicts with the fact that $x_0 \in X$ is the global minimum of the function f.

Now we give an algorithm, which give the solution of the problem (P). Let $A_1 \subseteq A_2 \subset \cdots \subset A_n \subset \ldots$ be a sequence of finite subsets of X such that $\bigcup_{n=1}^{\infty} A_n$ is dense in X thus

$$A_1 = \{a_1^1, \ldots, a_{q_1}^1\}$$
$$A_2 = \{a_1^2, \ldots, a_{q_2}^2\}$$
$$\vdots$$
$$A_n = \{a_1^n, \ldots, a_{q_n}^n\}$$
$$\vdots$$

We calculate

$$y_1 = \min\{\max\{f_1(x_1^1), f_2(x_1^1), \ldots, f_p(x_1^1)\}, \ldots, \max\{f_1(x_{q_1}^1), f_2(x_{q_1}^1), \ldots, f_p(x_{q_1}^1)\}\}$$
$$y_2 = \min\{\max\{f_1(x_1^2), f_2(x_1^2), \ldots, f_p(x_1^2)\}, \ldots, \max\{f_1(x_{q_2}^2), f_2(x_{q_2}^2), \ldots, f_p(x_{q_2}^2)\}\}$$
$$\vdots$$
$$y_n = \min\{\max\{f_1(x_1^n), f_2(x_1^n), \ldots, f_p(x_1^n)\}, \ldots, \max\{f_1(x_{q_n}^n), f_2(x_{q_n}^n), \ldots, f_p(x_{q_n}^n)\}\}$$

We obtain the sequence $(y_n)_{n \in N^*}$ which is monoton and decreasing, therefore is convergent. We have the following result:

Theorem 2.2: *The sequence $(y_n)_{n \in N^*}$ converge to the solution of the problem (P).*

Proof: We suppose that $y_n \to y^* > f(x_0)$. Because the set $\bigcup_{n=1}^{\infty} A_n$ is dense in X and the function f is continuous results that, there exists a sequence $\{x_n\} \subset \bigcup_{n=1}^{\infty} A_n$ such that $x_n \to x_0$ and $f(x_n) \to f(x_0)$. Without loss of generality we suppose that $x_1 \in A_1, \ldots, x_n \in A_n, \ldots$. Then $f(x_n) \geq y_n$ for every $n \in N^*$. If $n \to \infty$ we obtain $f(x_0) \geq y^*$, which is a contradiction with the presumption $y^* > f(x_0)$.

Remark. From the algorithm we see that for every $n \in N^*$, there exists $i_n \in \{1, \ldots, q_n\}$ such that $y_n = \max\{f_1(x_{i_n}^n), \ldots, f_p(x_{i_n}^n)\}$. Thus we obtain a sequence $\{x_{i_n}^n\}_{n \in N^*}$. Then every accumulation point of this sequence gives a solution of the problem (P).

3. JOIN OF THREE RELATIONS

We present the stochastic optimization model for the join of three relations, when the relations are stored at different sites. Let Q_3 denote the single-query type consisting of two joins:

$$Q_3 = A \bowtie B \bowtie C$$

where $A \cap B \neq \emptyset$; $B \cap C \neq \emptyset$ and relation A is stored at site 1, relation B at site 2 and relation C at site 3. So the initial state of relations referenced by the query Q_3 in the three-site network is the next column vector:

$$x_0 = \begin{pmatrix} A \\ B \\ C \end{pmatrix}$$

where the i-th component of the vector x_0 is the set of relations stored at site $i (i = 1, 2, 3)$ at time $t = 0$. The initial state x_0 is given with time-invariant probability $p_0 = p(x_0)$, i.e. p_0 is the probability that relation A is available at site 1, relation B at site 2 and relation C at site 3 and there are not locked for updating or is unavailable for query processing for any other reason. We assume that the input to the system consists of a single stream of type Q_3.

For the purpose of stochastic query optimization we enumerate all logically valid joins in the order in which they may be executed. We suppose that Q_3 has two valid execution sequences:

$$Q_3S_1 = (A \bowtie B) \bowtie C$$

$$Q_3S_2 = A \bowtie (B \bowtie C)$$

where S_1 and S_2 denote the sequences, so under sequence S_1 $B' = A \bowtie B$ is computed before $C' = B' \bowtie C$, and under sequence S_2 $C' = B \bowtie C$ is executed before $B' = A \bowtie C'$. In this way, the computation of multiple-join queries may be defined in terms of precedence of the join operator. The symbols Q_3S_1 and Q_3S_2 will be regarded as subtypes of the query type Q_3 and using the decomposition principle of separable nonlinear programming, the symbols will be used as conditioning variables that divide the original problem into a set of subproblems.

The state-transition graph for sequence S_1 of Q_3 is given in Figure 1. For one state of the state-transition graph the i-th line contains the relations stored at site i. We will associate a transition probability to each transition arc of the state-transition model. Let p_{ij} denote the conditional, time-invariant probability that the system undergoes transition from state x_i to state x_j. Given the initial state x_0, we can execute the first step of S_1Q_3 transferring relation B from site 2 to site 1, or transferring relation A from site 1 to site 2. Using the first strategy the system undergoes transition from state x_0 to x_{11} with probability $p_{0,11}$. The system may choose the second strategy with probability $p_{0,21}$ when the system undergoes transition to state x_{21}. The notation for states is: x_{ij}, where i is the

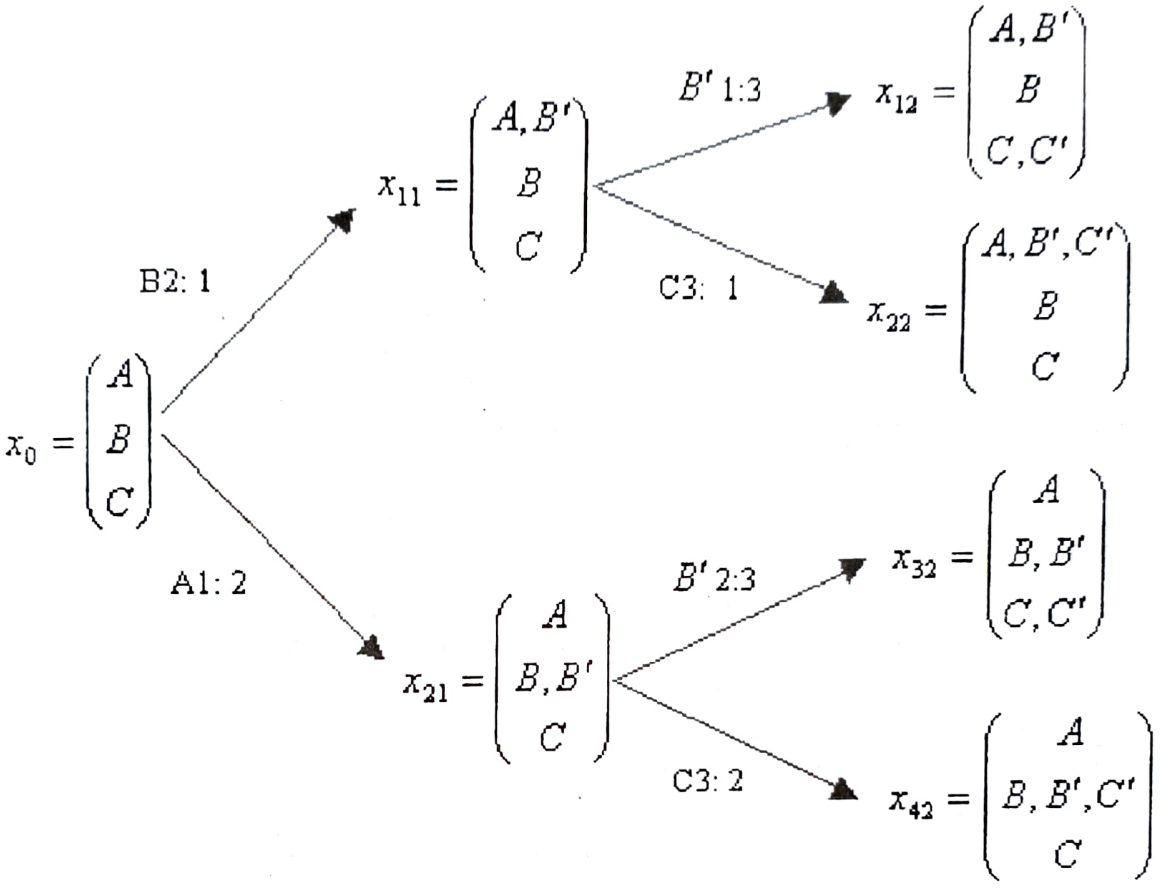

Fig. 1

number of the chosen strategy, (in our example $i = 1, 2, 3, 4$) and j is the step in computing the multiple join. In order to compute $C' = B' \bowtie C$, if the system is in state x_{11} it may transfer relation B' from site 2 to site 1 or relation C from site 3 to site 1 and if the system is in state x_{21} it may transfer relation B' from site 2 to site 3 or relation C from site 3 to site 2.

Theorem 3.1 *The stochastic query optimization model for the multiple join query of type Q_3 defines a nonlinear programming problem.*

Proof: We will associate the join-processing times with the nodes of the state-transition graph and communication times to the arcs of the graph. Let $T_i(X)$ denote the total processing time required for computing in state i. So we have:

$$
\begin{aligned}
T_{11}(B') &= t_1(A \bowtie B) + c_{21}(B) \\
T_{21}(B') &= t_2(A \bowtie B) + c_{12}(A) \\
T_{12}(C') &= t_2(B' \bowtie C) + c_{13}(B') \\
T_{22}(C') &= t_1(B' \bowtie C) + c_{31}(C) \\
T_{32}(C') &= t_3(B' \bowtie C) + c_{23}(B') \\
T_{42}(C') &= t_2(B' \bowtie C) + c_{32}(C)
\end{aligned}
$$

where $t_i(E)$ denotes the necessary time to calculate the expression E in site i and $c_{ij}(R)$ is the time of transmission the relation R from site i to site j. The expected delay due to computing the join is the product of the delay and the corresponding transition probability. The mean processing time τ_i at site i can be obtained by summing for each state for which there is something to work in the site i, the product of the necessary time for processing multiply by the probability that the system is in the corresponding state.

Let us suppose that input queries of type Q_3 arrive at the system at average intervals of length δ and successive inputs are statistically independent. It is reasonable to require that none of the processors in the network be allowed to take longer on the average than the period δ to execute its task. If it did, the cumulative delay at each site could increase indefinitely due to queuing, requiring infinite buffer storage at each site. The system may be regarded as overloaded if the mean processing time τ_i is permitted to exceed δ at any site. Such overload can be avoided if the inequalities:

$$\tau_i \leq \Delta < \delta$$

are satisfied, where Δ represents a common upper bound on τ_i, for each processor i in the network. In other to maximize the system query-processing capacity, or throughput $\Delta = 1/\delta$, the system's mean interarrival time Δ may be minimized, where $(\delta - \Delta) > 0$ is chosen sufficiently large to provide adequate buffer storage requirements.

The stochastic query optimization problem for Q_3 is given by:

$$
\begin{aligned}
\tau_1 &= T_{11}(B')p_{0,11} + T_{22}(C')p_{0,11}p_{11,22} \leq \Delta \\
\tau_2 &= T_{21}(B')p_{0,21} + T_{42}(C')p_{0,21}p_{21,42} \leq \Delta \\
\tau_3 &= T_{12}(C')p_{0,11}p_{11,12} + T_{32}(C')p_{0,21}p_{21,32} \leq \Delta \\
& p_{0,11} + p_{0,21} = 1 \\
& p_{11,12} + p_{11,22} = 1 \\
& p_{21,32} + p_{21,42} = 1 \\
& \min \Delta
\end{aligned}
$$

We solve this nonlinear programming problem with the algorithm from section 2 and in the table from Figure 2 we give some examples to illustrate the results obtained.

The continuous functions, which are the inputs for the algorithm are:

$f_1(x_1, x_2, x_3, x_4, x_5, x_6) = c_1 x_1 + c_2 x_1 x_4$
$f_2(x_1, x_2, x_3, x_4, x_5, x_6) = c_3 x_2 + c_4 x_2 x_6$
$f_3(x_1, x_2, x_3, x_4, x_5, x_6) = c_5 x_1 x_3 + c_6 x_2 x_5$

where $x_1 = p_{0,11}; x_2 = p_{0,21}; x_3 = p_{11,12}; x_4 = p_{11,22}; x_5 = p_{21,32}; x_6 = p_{21,42};$
$c_1 = T_{11}(B'); c_2 = T_{22}(C'); c_3 = T_{21}(B');$
$c_4 = T_{42}(C'); c_5 = T_{12}(C'); c_6 = T_{32}(C');$

We consider a communication network with a speed of 60 Kbits/s, which is very slow compared with the speed of local read/write of 3Mbyte/s, so the required time for local processing is unsignificant regard to the necessary time for data transfer. In the computation of the results we ommit the local processing time. We consider every connection has speed 60 Kbits/s. The model is more general, can take into account connections with different speed, i. e. the speed of transfer between the site 1 and site 2 is different from the speed of transfer between the site 2 and site 3. We consider relation A with tuples of length 100 bytes, relations B and C with tuples of length 50 bytes.

Case	a)	b)	c)	d)
Nr. of bits for A	8.000.000	8.000.000	1.000.000	1.000.000
Nr. of bits for B	4.000.000	1.000.000	10.000.000	30.000.000
Nr. of bits for C	10.000000	5.000.000	1.000.000	1.000.000
Δ	85,259	51,544	26,53	30,99
$p_{0,11}$	0,715	0,762	0,145	0,06
$p_{0,21}$	0,285	0,238	0,855	0,94
$p_{11,12}$	0,685	0,39	0,022	0,005
$p_{11,22}$	0,315	0,61	0,978	0,995
$p_{21,32}$	0,005	0	0,137	0,025
$p_{21,42}$	0,995	1	0,863	0,975

Fig. 2

Let us analyze the results obtained:

- In case a) the size of relation A is double regard to the size of relation B, so thus the result show to choose state x_{11} in the most of the cases if the system is in state x_0, when the system has to transfer relation B, being the smaller. The efficiency of the model can be seen in the following way: without this model a heuristic for query execution can be: we transfer always the smaller relation from the operands of a join to the site of the other relation operand, taking into account that the required time for local processing is unsignificant regard to the necessary time for data

transfer. In this case the heuristic can be: relation B is transferred from site 2 to site 1, being the smaller, so the system in state x_0 always chooses state x_{11} with transition probability 1, which needs 16,66 seconds. In state x_{11} the system will transfer relation C from site 3 to site 1, because it is smaller than the result of join $A \bowtie B$, the necessary time is 166,66 seconds, so totalized 233,32 seconds. If query Q_3 is executed in every case it appear with this "pure" strategy the mean processing time will be 233,32 seconds and both of the queries is executed in site 1. The mean processing time in the case of the stochastic optimization model is 85,259 seconds, which gives the next strategies: from 100 queries of type Q_3 for 72 $A \bowtie B$ is executed in site 1 relation B being transferred from site 2 in site 1, the mean processing time being 47,99 seconds, and 28 queries are executed in site 2 relation A being transferred from site 1 in site 2, the mean processing time being 38,33 seconds. From 72 queries $68 B' \bowtie C$ is executed in site 3 transferring relation B' from site 1 to site 3, the mean processing time being 84,93 seconds, and for 32is site 1 transferring relation C from site 3 in site 1, the mean processing time being 38,33 seconds. In site x_{21} the system always choose relation C to transfer from site 3 in site 2 and execute the join in site 2, the mean processing time being 46,66 seconds. If we calculate the mean processing time for each site, then take the maximum of them we obtain 85,26 s.

- Case b) doesn't differ significant from case a), so the obtained result is similar, because the relations are smaller, the mean processing time obtained is smaller too. If in this case we choose a "pure" strategy with the same heuristic such as in case a), the system undergoes transition from state x_0 to state x_{11}, the required time being 16,66 seconds, from state x_{11} to state x_{22} the transferring time is 83,33 seconds, so the total time is 99,99 seconds, which is nearly the double of 51,544 seconds, the mean processing time obtained with the stochastic optimization model.
- In case c), relation B being the larger the results will be inverse. A "pure" strategy in this case can be: relation A is transferred to site 2 in the first step, relation C in site 2 too in the second step, these relations being the smaller from the operands of joins. The required time for this "pure" strategy is 33,33 seconds, which is appropriate to the 26,53 seconds of the stochastic optimization model, because the difference between the size of relations is great and in this case the stochastic model can give a "pure" strategy too.
- In case d) we try to appropriate to a "pure" strategy, but for these great differences too, the stochastic model gives a better strategy, but we can see the approach to the "pure" strategy.

In the table from Figure 3 we will give the obtained results in case of a communication network with a speed of data transfer of 2,4Mbps and the speed of

local read/write is 3Mbytes/s, So the rate between the local processing and data transfer is 10:1. In this case we consider the local processing time too in our computation.

Case	a)	b)	c)
Nr. of bits for A	8.000.000	8.000.000	1.000.000
Nr. of bits for B	4.000.000	1.000.000	10.000.000
Nr. of bits for C	10.000000	5.000.000	1.000.000
Δ	2,93	2,03	1,88
$p_{0,11}$	0,7	0,725	0,283
$p_{0,21}$	0,3	0,275	0,717
$p_{11,12}$	0,735	0,51	0,013
$p_{11,22}$	0,265	0,49	0,987
$p_{21,32}$	0	0	0,395
$p_{21,42}$	1	1	0,605

Fig. 3

There is no general way to estimate the cardinality of a join without additional information. There is a case, which occurs frequently, where the estimation is simple, we use this one. If the join attribute for relation A and B is K, which is primary key for relation B and is foreign key for relation A, so cardinality of the result can be estimated by

$$card(A \bowtie_{A.K=B.K} B) = card(A)$$

because each tuple of A matches with at most one tuple of B. However, this estimation is an upper bound since it assumes that each tuple of relation A participate in the join. The length of a tuple of the join is the sum of the length of the operand relations minus the length of the join attribute. A DBMS can take the join selectivity factor from the statistical information of the database. The necessary time for join execution can differ in function of the method used. If there exists an index file for relation B with the join attribute as key, the indexed loop join is an adequate method, but it needs in some cases the transfer of index file too. We consider the unindexed loop join method in our calculations.

The analysis of results can be made similar with the precedent case, but for the "pure" strategy we have to take into account the local processing time too.

ACKNOWLEDGMENT. The third author would like to thank Professor A. Benczur for introducing her to reference [4] and for the interesting discussions.

REFERENCES

[1] Date C. J.: *An Introduction to Database Systems*, Addison- Wesley Publishing Company, 1995.

[2] Drenick, R. F.: *A Mathematical Organization Theory*, Elsevier, New York, 1986.

[3] Drenick, P. E., Drenick R. F.: *A Design Theory for Multi-processing Computing Systems*, in Large Scale Syst. Vol. 12, 1987, pp. 155-172.
[4] Drenick, P. E., Smith E. J.: *Stochastic Query Optimization in Distributed Databases*, in ACM Trans. on Database Systems, Vol. 18, No. 2, June 1993, pp. 262-288.
[5] LaFortune, S., Wong, E.: *A State Transition Model for Distributed Query Processing*, in ACM Trans. on Database Systems, Vol. 11, No. 3, Sept. 1986, pp. 294-322.
[6] M. T. Özsu, P. Valduriez: *Principles of Distributed Database Systems*, Prentice-Hall, 1991.
[7] Ramakrishnan, R.: *Database Management Systems*, WCB McGraw-Hill, 1998.
[8] Ullman, J. D.: *Principles of Database and Knowledge-Base Systems*, Vol. I-II, Computer Science Press, 1988.

Babeş-Bolyai University, Faculty of Mathematics and Informatics,
RO 3400 Cluj-Napoca, str. Kogălniceanu 1, Romania
E-mail address: ivarga@cs.ubbcluj.ro