Annales Univ. Sci. Budapest., Sect. Comp. 20 (2001) 107-131

STOCHASTIC QUERY OPTIMIZATION IN DISTRIBUTED DATABASES USING SEMOJOINS

T. Márkus (Budapest, Hungary)
C. Moroşanu (Iaşi, Roumania)
V. Varga (Cluj-Napoca, Roumania)

Abstract. Many algorithms were elaborated for minimizing the costs necessary to perform a single, isolated query in a distributed database system. A distributed system can receive 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. Since in distributed database systems the transmission costs are the dominant, strategies with semijoins are important. There are presented strategies to perform a singe-join and multiple-joins using semijoins. It is shown that the strategy for a single-join leads to linear programming problem. It treats the case of N queries, where one query is the join of two relations and the relations involved in the query are stored in two sites. There is treated the sequential appearance of queries and the parallel execution of them. Multiple-joins lead to nonlinear programming, there is given the problem in the case of the join of three relations. In the appendix we present an algorithm to solve this special kind of nonlinear programming, which gives a global optimum. We give the optimization problem for the case of full reducers, which leads to a nonlinear programming problem and can be solved with the given algorithm, too.

Introduction

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 efficiently, and determine an economic strategy for data transferring. For each new type of query that arrives at the system, a new optimal strategy is determined.

The capacity of distributed systems for concurrent processing motives 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 processes 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.

In distributed environment, where transmission costs must be minimized, it is efficient to work with semijoins. In this article we give strategies to perform a single-join and multiple-joins using semijoins. In Section 1 it is described the strategy for a single-join, which leads to linear programming problem. In Section 2 we extend the model for N queries, where one query is the join of two relations and the relations involved in the query are stored in two sites. It is treated the sequential appearance of queries and the parallel execution of them. Multiple joins lead to nonlinear programming, so in Section 3 we formulate the optimization problem in the case of the join of three relations. This is a special kind of nonlinear programming problem. In the Appendix we present an algorithm to solve this problem and we prove, that the algorithm gives the global optimum of the nonlinear problem. We illustrate by some examples the solutions obtained applying the algorithm. If the corresponding hypergraph of the query is acyclic, we can reduce the operand relations in order to transmit only the essential part of the relation. Section 3 continues with the stochastic optimization model for the full reducer program of three relations. This optimization problem leads to a same kind of nonlinear programming problem, which can be solved by the algorithm from the Appendix.

1. The single-join, single-query model

We will extend the single-join model from [4] for semijoins. Let Q_1 denote the single-query type consisting of the single-join

$$Q_1 = A \bowtie B,$$

where $A \cap B \neq \emptyset$ and at time $t = 0$ the relation A is stored at site 1 and relation B at site 2. Thus, the "initial state" of relations referenced by the query Q_1 in the two-site network is the next column vector

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

where the i-th component of the vector x_0 is the set of relations stored at site i $(i = 1, 2)$ 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 and relation B at site 2, 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_1.

In distributed databases the communication cost is the dominant factor in calculating the join of two or more relations, stored at different sites. The method using semijoins saves some communication costs. The queries whose hypergraphs are acyclic have optimization algorithms that in general are simpler and more efficient than in the general case, see [8], pp. 698-707. In the case of two relations join the corresponding hypergraph is acyclic.

We can calculate $A \bowtie B$ by applying semijoins following the next steps:

1. Compute $\pi_{A \cap B}(A)$ at site 1.
2. Ship $\pi_{A \cap B}(A)$ to site 2.
3. Compute $B \propto A$ at site 2, using the fact that $B \propto A = B \bowtie \pi_{A \cap B}(A)$ ($\propto$ denotes the semijoin operation).
4. Ship $B \propto A$ to site 1.
5. Compute $A \bowtie B$ at site 1, using the fact that $A \bowtie B = A \bowtie (B \bowtie A)$.

This strategy is more efficient than calculating the join in the case of distributed queries, if the following inequality holds:

$$c_0 + \min(T'_A l'_A + T''_B l''_B,\ T'_B l'_B + T''_A l''_A) < \min(T_A l_A, T_B l_B),$$

where c_0 - is a fixed overhead cost charged to each transmission;

T_X - the number of tuples of relation X, where X can be A or B and l_X is the length of it in bytes;

T'_A - the number of tuples of $\pi_{A\cap B}(A)$ and l'_A is the length of it in bytes;

T'_B - the number of tuples of $\pi_{A\cap B}$ and l'_B is the length of it in bytes;

T''_A - the number of tuples of $A \propto B$ and l''_A is the length of it in bytes;

T''_B - is the number of tuples of $B \propto A$ and l''_B is the length of it in bytes.

In the case of the projection the tuples are shorter than in the case of the join or the semijoin.

There are two possible strategies, one presented above, and the other similar, by changing the relation A with relation B and site 1 with site 2. 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 compute Q_1 using the strategy presented above in detail, when system undergoes transition from x_0 to x_{13} through x_{11}, x_{12} (see Fig. 1.1). The system may choose another strategy, which is similar, by changing relation A with relation B and site 1 with site 2. Thus, the system may choose between these two strategies. We will associate probability $p_{0,11}$ if it chooses the first strategy, and $p_{0,21}$ if it chooses the second strategy. In the case of transition to state x_{11}, the system has to follow through x_{12} and x_{13} to finish the calculation of the semijoin. The same situation is true for x_{21}, x_{22}, x_{23}. The notation for states is: x_{ij}, where i is the number of the chosen strategy (in our example $i = 1, 2$), and j is the number of states involved in computing of the semijoin. For one state of the state-transition graph the i-th line contains the relations stored at site i.

The other notations for Fig. 1.1 are:

$$P_B^A = \pi_{A\cap B}(A),\ P_A^B = \pi_{A\cap B}(B);$$
$$A_s^B = A \propto B,\ B_s^A = B \propto A;$$
$$A_j^B = A \bowtie B$$

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

$$\to \left\{ \begin{array}{l} x_{11} = \begin{pmatrix} A, P_B^A \\ B \end{pmatrix} \xrightarrow{P_B^A 1:2} x_{12} = \begin{pmatrix} A, P_B^A \\ B, B_s^A \end{pmatrix} \xrightarrow{B_s^A 2:1} x_{13} = \begin{pmatrix} A, P_B^A, A_j^B \\ B, B_s^A \end{pmatrix} \\ x_{21} = \begin{pmatrix} A \\ B, P_A^B \end{pmatrix} \xrightarrow{P_B^A 2:1} x_{22} = \begin{pmatrix} A, A_s^B \\ B, P_A^B \end{pmatrix} \xrightarrow{A_s^B 1:2} x_{23} = \begin{pmatrix} A, A_s^B \\ B, P_A^B, A_j^B \end{pmatrix} \end{array} \right.$$

Fig.1.1.

The query-processing distribution $\{p_{0,11}, p_{0,21}\}$ represents a family of strategies for scheduling the processing of the query Q_1. In order to determine a best strategy we select as a performance criterion the system throughput. We have to determine first the system's mean processing time.

Theorem 1.1. *The single-join stochastic query optimization model with semijoins defines a linear programming problem, the solution of this linear programming problem is an optimal one.*

Proof. We will associate the projection-processing and the semijoin-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

$$\begin{aligned}
T_{11}(P_B^A) &= t_1(\pi_{A\cap B}(A)),\\
T_{12}(B_s^A) &= t_2(B \propto A) + c_{12}(P_B^A),\\
T_{13}(A_j^B) &= t_1(A \bowtie B_s^A) + c_{21}(B_s^A),\\
T_{21}(P_A^B) &= t_2(\pi_{A\cap B}(B)),\\
T_{22}(A_s^B) &= t_1(A \propto B) + c_{21}(P_A^B),\\
T_{23}(A_j^B) &= t_2(A_s^B \bowtie B) + c_{12}(A \propto B),
\end{aligned}$$

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 projection or semijoin, 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 multiplied by the probability that the systems is in the corresponding state.

Let us suppose that input queries of type Q_1 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 upperbound on τ_i, for each processor i in the network. In order to maximize the system query-processing capacity, or throughput $\Delta = 1/\delta$, the systems'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_1 is given by

$$\tau_1 = T_{11}(P_B^A)p_{0,11} + T_{13}(A_j^B)p_{0,11} + T_{22}(A_s^B)p_{0,21} \leq \Delta,$$
$$\tau_2 = T_{12}(B_s^A)p_{0,11} + T_{21}(P_A^B)p_{0,21} + T_{23}(A_j^B)p_{0,21} \leq \Delta,$$

$$p_{0,11} + p_{0,21} = 1,$$
$$\min \Delta,$$
$$p_{0,11},\ p_{0,21} \geq 0;\ \Delta > 0.$$

In the linear programming problem $\Delta, p_{0,11}, p_{0,21}$ are the decision variables (i.e. the unknowns) and $T_{11}(P_B^A), T_{12}(B_s^A), T_{13}(A_j^B), T_{21}(P_A^B), T_{22}(A_s^B), T_{23}(A_j^B)$ are assumed to be constants.

For solving this linear programming problem we will distinguish different cases:

a)

$$t_1(\pi_{A\cap B}(A)) + t_1(A \bowtie B_s^A) + c_{21}(B_s^A) > t_1(A \propto B) + c_{21}(P_A^B),$$
$$t_2(\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B) > t_2(B \propto A) + c_{12}(P_B^A),$$
$$t_2(\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B) > t_1(A \propto B) + c_{21}(P_A^B),$$

this is the most frequent occurrence.

The solution in case a) is

$$p_{0,11} = \frac{T_{21}(P_A^B) + T_{23}(A_j^B) - T_{22}(A_s^B)}{T_{11}(P_B^A) + T_{13}(A_j^B) - T_{22}(A_s^B) - T_{12}(B_s^A) + T_{21}(P_A^B) + T_{23}(A_j^B)},$$

$$p_{0,21} = \frac{T_{11}(P_B^A) + T_{13}(A_j^B) - T_{12}(B_s^A)}{T_{11}(P_B^A) + T_{13}(A_j^B) - T_{22}(A_s^B) - T_{12}(B_s^A) + T_{21}(P_A^B) + T_{23}(A_j^B)},$$

$$\Delta = \frac{(T_{11}(P_B^A) + T_{13}(A_j^B))(T_{21}(P_A^B) + T_{23}(A_j^B)) - T_{22}(A_s^B)T_{12}(B_s^A)}{T_{11}(P_B^A) + T_{13}(A_j^B) - T_{22}(A_s^B) - T_{12}(B_s^A) + T_{21}(P_A^B) + T_{23}(A_j^B)},$$

where T_{ij} are defined above.

Because the probability $p_{0,21}$ has to be positive, so the numerator has to be positive, since the denominator is positive from the above preconditions. It means that

$$t_1(\pi_{A\cap B}(A)) + t_1(A \bowtie B_s^A) + c_{21}(B_s^A) > t_2(B \propto A) + c_{21}(P_B^A),$$

which is true in most of the cases, because in the left-hand side of the inequality is the sum of the necessary time to calculate the projection and the join while in the right-hand side is only the necessary time to calculate the semijoin.

b)

$$\begin{aligned}
&t_1(\pi_{A\cap B}(A)) + t_1(A \bowtie B_s^A) + c_{21}(B_s^A) > t_1(A \propto B) + c_{21}(P_A^B),\\
&t_2(\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B) > t_2(B \propto A) + c_{12}(P_B^A),\\
&t_2(\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B) < t_1(A \propto B) + c_{21}(P_A^B).
\end{aligned}$$

In this case the solution is

$$P_{0,11} = 0,\ P_{0,21} = 1,\ \Delta = t_1(A \propto B) + c_{21}(P_A^B).$$

c)

$$\begin{aligned}
&t_1(\pi_{A\cap B}(A)) + t_1(A \bowtie B_s^A) + c_{21}(B_s^A) > t_1(A \propto B) + c_{21}(P_A^B),\\
&t_2(\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B) < t_2(B \propto A) + c_{12}(P_B^A),\\
&t_2(\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B) > t_1(A \propto B) + c_{21}(P_A^B).
\end{aligned}$$

In this case the solution is

$$P_{0,11} = 0,\ P_{0,21} = 1,\ \Delta = t_2\ (\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B).$$

d)

$$\begin{aligned}
&t_1(\pi_{A\cap B}(A)) + t_1(A \bowtie B_s^A) + c_{21}(B_s^A) > t_1(A \propto B) + c_{21}(P_A^B),\\
&t_2(\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B) < t_2(B \propto A) + c_{12}(P_B^A),\\
&t_2(\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B) < t_1(A \propto B) + c_{21}(P_A^B).
\end{aligned}$$

In this case the solution is

$$P_{0,11} = 0,\ P_{0,21} = 1,\ \Delta = t_1(A \propto B) + c_{21}(P_A^B).$$

e)

$$t_1(\pi_{A\cap B}(A)) + t_1(A \bowtie B_s^A) + c_{21}(B_s^A) < t_1(A \propto B) + c_{21}(P_A^B),$$
$$t_2(\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B) > t_2(B \propto A) + c_{12}(P_B^A),$$
$$t_2(\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B) > t_1(A \propto B) + c_{21}(P_A^B).$$

In this case the minimal value for Δ is 0, which has no meaning.

f)

$$t_1(\pi_{A\cap B}(A)) + t_1(A \bowtie B_s^A) + c_{21}(B_s^A) < t_1(A \propto B) + c_{21}(P_A^B),$$
$$t_2(\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B) > t_2(B \propto A) + c_{12}(P_B^A),$$
$$t_2(\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B) < t_1(A \propto B) + c_{21}(P_A^B).$$

In this case the minimal value for Δ is 0, which has no meaning.

g)

$$t_1(\pi_{A\cap B}(A)) + t_1(A \bowtie B_s^A) + c_{21}(B_s^A) < t_1(A \propto B) + c_{21}(P_A^B),$$
$$t_2(\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B) < t_2(B \propto A) + c_{12}(P_B^A),$$
$$t_2(\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B) > t_1(A \propto B) + c_{21}(P_A^B).$$

In this case the solution is

$$P_{0,11} = 0,\ P_{0,21} = 1,\ \Delta = t_2\ (\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B).$$

h)

$$t_1(\pi_{A\cap B}(A)) + t_1(A \bowtie B_s^A) + c_{21}(B_s^A) < t_1(A \propto B) + c_{21}(P_A^B),$$
$$t_2(\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B) < t_2(B \propto A) + c_{12}(P_B^A),$$
$$t_2(\pi_{A\cap B}(B)) + t_2(A_s^B \bowtie B) + c_{12}(A_s^B) < t_1(A \propto B) + c_{21}(P_A^B).$$

In this case the solution is

$$p_{0,11} = \frac{T_{22}(A_s^B) - T_{21}(P_A^B) - T_{23}(A_j^B)}{T_{22}(A_s^B) - T_{11}(P_B^A) - T_{13}(A_j^B) + T_{12}(B_s^A) - T_{21}(P_A^B) - T_{23}(A_j^B)},$$

$$p_{0,21} = \frac{T_{12}(B_s^A) - T_{11}(P_B^A) - T_{13}(A_j^B)}{T_{22}(A_s^B) - T_{11}(P_B^A) - T_{13}(A_j^B) + T_{12}(B_s^A) - T_{21}(P_A^B) - T_{23}(A_j^B)},$$

$$\Delta = \frac{T_{22}(A_s^B)T_{12}(B_s^A) - (T_{11}(P_B^A) + T_{13}(A_j^B))(T_{21}(P_A^B) + T_{23}(A_j^B))}{T_{22}(A_s^B) - T_{11}(P_B^A) - T_{13}(A_j^B) + T_{12}(B_s^A) - T_{21}(P_A^B) - T_{23}(A_j^B)}.$$

The probability $p_{0,21}$ has to be positive, so there is another necessary condition

$$t_1(\pi_{A\cap B}(A)) + t_1(A \bowtie B_s^A) + c_{21}(B_s^A) < t_2(B \propto A) + c_{21}(P_B^A).$$

In the local area networks the cases b-g are only mathematical ones, because the left-hand side of each inequality is always greater than the right-hand side, so only the case a) appears.

In the wide area networks the communication times are dominant, we have one communication time in the left-hand side and another communication cost in the right-hand side, so any case of a)-h) can appear.

2. General single-join models

2.1. Sequential operation

In a distributed database system we can execute multiple queries. In this section we will treat the sequential mode, in which queries arrive separately, one after the other, with an average interarrival time of length δ.

Let

$$Q_i = A_i \bowtie B_i, \quad i = 1, 2, \dots, N$$

denote N distinct query types (where relations A_i and B_i need not be distinct) and two sites, where these relations are stored; $A_i,\ i = 1, 2, \dots, N$ are stored at site 1 and $B_i,\ i = 1, 2, \dots, N$ are stored at site 2. So let the initial state be

$$x_0 = \begin{pmatrix} A_1, & A_2, \dots, A_N \\ B_1, & B_2, \dots, B_N \end{pmatrix}.$$

We will calculate these joins using semijoins. Let q_i denote the probability that a given query is of type Q_i, where $q_1 + q_2 + \dots + q_N = 1;\ 0 \le q_i \le 1$. It will be assumed that q_i are known.

Theorem 2.1. *The sequential single-join stochastic query optimization model with semijoin for N queries and 2 sites defines a linear programming problem that may be decomposed into N independent linear programming subproblems.*

Proof. Let τ_{li} be the mean processing time at site l for query i, in this case $l = 1, 2$.

$$\tau_{1i} = (T_{i11}(P_{B_i}^{A_i}) + T_{i13}(A_{ij}^{B_i}))p_{0,1i} + T_{i22}(A_{is}^{B_i})p_{0,i2},$$
$$\tau_{2i} = T_{i12}(B_{is}^{A_i})p_{0,i1} + (T_{i21}(P_{A_i}^{B_i}) + T_{i23}(A_{ij}^{B_i}))p_{0,i2},$$

where

$$T_{i11}(P_{B_i}^{A_i}) = t_1(\pi_{A_i \cap B_i}(A_i)); \quad P_{B_i}^{A_i} = \pi_{A_i \cap B_i}(A_i);$$
$$T_{i12}(B_{is}^{A_i}) = t_2(B_i \propto A_i) + c_{12}(P_{B_i}^{A_i}); \quad B_{is}^{A_i} = B \propto A_i;$$
$$T_{i13}(A_{ij}^{B_i}) = t_1(A_i \bowtie B_{is}^{A_i}) + c_{21}(C_{is}^{A_i}); \quad A_{ij}^{B_i} = A_i \bowtie B_{is}^{A_i};$$
$$T_{i21}(P_{A_i}^{B_i}) = t_2(\pi_{A_1 \cap B_i}(B_i)); \quad P_{A_i}^{B_i} = \pi_{A_i \cap B_i}(B_i);$$
$$T_{i22}(A_{is}^{B_i}) = t_1(A_i \propto B_i) + c_{21}(P_{A_i}^{B_i}); \quad A_{is}^{B_i} = A \propto B_i;$$
$$T_{i23}(A_{ij}^{B_i}) = t_2(A_{is}^{B_i} \bowtie B_i) + c_{12}(A_{is}^{B_i}); \quad A_{ij}^{B_i} = A_{is}^{B_i} = A_{is}^{B-I} \bowtie B_i;$$

and $p_{0,ik}$ is the network transition probability associated with a change of state x_0 to state x_{ik1}, where i is the index of the query Q_i and k is the number of the chosen strategy in the case $k = 1, 2$. τ_{li} is the mean processing time at site l for query i, in this case $l = 1, 2$. The linear programming problem is obtained by minimizing the mean interarrival time, subject to the system's overload, normalization and nonnegativity constraints, i.e.

$$\min \Delta$$

subject to

$$\sum_{i=1}^{N} q_i \tau_{li} \geq \Delta,$$

$$\sum_{i=1}^{N} \sum_{k=1}^{2} p_{0,ik} = 1,$$

$$p_{0,ik} \geq 0, \quad i = 1, 2, \ldots, N, \quad k = 1, 2,$$

$$\Delta > 0.$$

But Δ can always be written as the sum of conditional means

$$\Delta = \sum_{i=1}^{N} q_i \Delta_i, \tag{2.1}$$

where Δ_i is the mean interarrival time conditioned on query type Q_i, Δ is separable in its conditional means Δ_i. By the decomposition principle of linear programming, the original problem may be divided into subproblems by query type, for each $i = 1, 2, \ldots, N$

$$\min \Delta_i$$

subject to

$$\begin{aligned} \tau_{ki} &\le \Delta_i,\ k = 1, 2,\\ p_{0,i1} + p_{0,i2} &= 1,\\ p_{0,ik} &\ge 0 \end{aligned}$$

gives a linear programming subproblem in a set of decision variables disjoint from the decision variables of the other subproblems. Thus each such subproblem may be solved independently of the others, and the total mean can be computed by (2.1).

Example 2.1. Let $N = 2$,

$$Q_1 = A_1 \bowtie B_1, \quad Q_2 = A_2 \bowtie B_2$$

and the initial state

$$x_0 = \begin{pmatrix} A_1, & A_2\\ B_1, & B_2 \end{pmatrix}.$$

The state-transition graph for this model is shown in Fig. 2.1. The indexes for state x_{iks} are: i is for the query $i, i = 1, 2$, k is for strategy, $k = 1, 2$ and s is the number of state within one strategy.

For query Q_1, the linear programming problem is

$$\begin{aligned} \tau_{11} &= (T_{111}(P^{A_1}_{B_1}) + T_{113}(A^{B_1}_{1j}))p_{0,11} + T_{122}(A^{B_1}_{1s})p_{0,12} \le \Delta_1,\\ \tau_{21} &= T_{112}(B^{A_1}_{1s})p_{0,11} + (T_{121}(P^{B_1}_{A_1}) + T_{123}(A^{B_1}_{1j}))p_{0,12} \le \Delta_1, \end{aligned}$$

$$p_{0,11} + p_{0,12} = 1,$$
$$\min \Delta_1,$$

where

$$\begin{aligned} T_{111}(P^{A_1}_{B_1}) &= t_1(\pi_{A_1 \cap B_1}(A_1)),\\ T_{112}(B^{A_1}_{1s}) &= t_2(B_1 \propto A_1) + c_{12}(P^{A_1}_{B_1}),\\ T_{113}(A^{B_1}_{ij}) &= t_1(A_1 \bowtie B^{A_1}_{1s}) + c_{21}(C^{A_1}_{1s}),\\ T_{121}(P^{B_1}_{A_1}) &= t_2(\pi_{A_1 \cap B_1}(B_1)),\\ T_{122}(A^{B_1}_{1s}) &= t_1(A_1 \propto B_1) + c_{21}(P^{B_1}_{A_1}),\\ T_{123}(A^{B_1}_{1j}) &= t_2(A^{B_1}_{1s} \bowtie B_1) + c_{12}(A^{B_1}_{1s}), \end{aligned}$$

Δ_1 is the mean interarrival time conditioned on query type Q_1.

For query Q_2, the linear programming problem is

$$\tau_{12} = (T_{211}(P_{B_2}^{A_2}) + T_{213}(A_{2j}^{B_2}))p_{0,21} + T_{222}(A_{2s}^{B_2})p_{0,22} \le \Delta_2,$$
$$\tau_{22} = T_{212}(B_{2s}^{A_2})p_{0,21} + (T_{221}(P_{A_2}^{B_2}) + T_{223}(A_{2j}^{B_2}))p_{0,22} \le \Delta_2,$$
$$p_{0,21} + p_{0,22} = 1,$$
$$\min \Delta_2,$$

$$x_0 = \begin{pmatrix} A_1, A_2 \\ B_1, B_2 \end{pmatrix} \to$$

$$\to \left\{ \begin{array}{ll}
 & x_{111} = \begin{pmatrix} A_1, A_2, P_{B_1}^{A_1} \\ B_1, B_2 \end{pmatrix} \xrightarrow{P_{B_1}^{A_1} 1:2} \\
 & \xrightarrow{P_{B_1}^{A_1} 1:2} x_{112} = \begin{pmatrix} A_1, A_2 \\ B_1, B_2, B_{1s}^{A_1} \end{pmatrix} \xrightarrow{B_{1s}^{A_1} 2:1} x_{113} = \begin{pmatrix} A_1, A_2, A_{1j}^{B_1} \\ B_1, B_2 \end{pmatrix} \\
Q_1 & \\
 & x_{121} = \begin{pmatrix} A_1, A_2 \\ B_1, B_2, P_{A_1}^{B_1} \end{pmatrix} \xrightarrow{P_{A_1}^{B_1} 2:1} \\
 & \xrightarrow{P_{A_1}^{B_1} 2:1} x_{122} = \begin{pmatrix} A_1, A_2, A_{1s}^{B_1} \\ B_1, B_2 \end{pmatrix} \xrightarrow{A_{1s}^{B_1} 1:2} x_{123} = \begin{pmatrix} A_1, A_2 \\ B_1, B_2, A_{1j}^{B_1} \end{pmatrix} \\
 & x_{211} = \begin{pmatrix} A_1, A_2, P_{B_2}^{A_2}, \\ B_1, B_2 \end{pmatrix} \xrightarrow{P_{B_2}^{A_2} 1:2} \\
 & \xrightarrow{P_{B_2}^{A_2} 1:2} x_{212} = \begin{pmatrix} A_1, A_2 \\ B_1, B_2, B_{2s}^{A_2} \end{pmatrix} \xrightarrow{B_{2s}^{A_2} 2:1} x_{213} = \begin{pmatrix} A_1, A_2, A_{2j}^{B_2} \\ B_1, B_2 \end{pmatrix} \\
Q_2 & \\
 & x_{221} = \begin{pmatrix} A_1, A_2 \\ B_1, B_2, P_{A_2}^{B_2} \end{pmatrix} \xrightarrow{P_{A_2}^{B_2} 2:1} \\
 & \xrightarrow{P_{A_2}^{B_2} 2:1} x_{222} = \begin{pmatrix} A_1, A_2, A_{2s}^{B_2} \\ B_1, B_2 \end{pmatrix} \xrightarrow{A_{2s}^{B_2} 1:2} x_{223} = \begin{pmatrix} A_1, A_2 \\ B_1, B_2, A_{2j}^{B_2} \end{pmatrix}
\end{array} \right.$$

Fig. 2.1.

where

$$T_{211}(P_{B_2}^{A_2}) = t_1(\pi_{A_2 \cap B_2}(A_2)),$$
$$T_{212}(B_{2s}^{A_2}) = t_2(B_2 \propto A_2) + c_{12}(P_{B_2}^{A_2}),$$
$$T_{213}(A_{2j}^{B_2}) = t_1(A_2 \bowtie B_{2s}^{As}) + c_{21}(C_{2s}^{A_2}),$$

$$T_{221}(P_{A_2}^{B_2}) = t_2(\pi_{A_2 \cap B_2}(B_2)),$$
$$T_{222}(A_{2s}^{B_2}) = t_1(A_2 \propto B_2) + c_{21}(P_{A_2}^{B_2}),$$
$$T_{223}(A_{2j}^{B_2}) = t_2(A_{2s}^{B_2} \bowtie B_2) + c_{12}(A_{2s}^{B_2}),$$

Δ_2 is the mean interarrival time conditioned on query type Q_2.

We can obtain the solution for the distinct problems similar to the single-join, single-query case. The two optimization problems are completely independent, their separate solutions may be combined by $\Delta^* = q_1\Delta_1^* + q_2\Delta_2^*$.

2.2. Parallel operation

The previous section represents the sequential model, when queries arrive one after the other, but not if they arrive at approximately the same time. Let us consider that queries of types Q_1 and Q_2 are to be processed in parallel (although not necessarily at the same time).

Let

$$Q_1 = A_1 \bowtie B_1, \quad Q_2 = A_2 \bowtie B_2$$

and the initial state x_0 be

$$x_0 = \begin{pmatrix} A_1, A_2 \\ B_1, B_2 \end{pmatrix}.$$

Let Q_3 denote an aggregate query type, which occurs with probability q_3, $(q_1 + {} + q_2 + q_3 = 1)$, thus the input stream consists of mixed arrival of types Q_1, Q_2, Q_3. The state-transition graph for the parallel network machine for query Q_3 is in Fig. 2.2. We construct the states for the parallel machine in the following way: we unify one state from the processing of query Q_1 with the corresponding state from Q_2's processing, so that in the resulting state for the parallel machine in one of the sites there is something to process for one of the queries and in the other site is for the other query:

$$x_{1j} = x_{11j} \cup x_{22j}, \quad j = 1, 2, 3,$$
$$x_{2j} = x_{12j} \cup x_{21j}, \quad j = 1, 2, 3.$$

The stochastic query optimization problem for parallel processing of Q_1 and Q_2 is given by

$$\min \Delta_3$$

subject to

$$\begin{aligned}
\tau_{13} &= (T_{11}(P^{A_1}_{B_1}) + T_{12}(A^{B_2}_{2s}) + T_{13}(A^{B_1}_{1j}))p_{0,11} + (T_{21}(P^{A_2}_{B_2}) + T_{22}(A^{B_1}_{1s}) + \\
&\quad + T_{23}(A^{B_2}_{2j}))p_{0,21} \leq \Delta_3, \\
\tau_{23} &= (T_{11}(P^{B_2}_{A_2}) + T_{12}(B^{A_1}_{1s}) + T_{13}(A^{B_2}_{2j}))p_{0,11} + (T_{21}(P^{B_1}_{A_1}) + T_{22}(B^{A_2}_{2s}) + \\
&\quad + T_{23}(A^{B_1}_{1j}))p_{0,21} \leq \Delta_3,
\end{aligned}$$

$$p_{0,11} + p_{0,21} = 1,$$

where

$$\begin{aligned}
&T_{11}(P^{A_1}_{B_1}) = t_1(\pi_{A_1 \cap B_1}(A_1)); \quad T_{11}(P^{B_2}_{A_2}) = t_2(\pi_{A_2 \cap B_2}(B_2)); \\
&T_{12}(B^{A_1}_{1s}) = t_2(B_1 \propto A_1) + c_{12}(P^{A_1}_{B_1}); \\
&T_{12}(A^{B_2}_{2s}) = t_1(A_2 \propto B_2) + c_{21}(P^{B_2}_{A_2}); \\
&T_{13}(A^{B_1}_{1j}) = t_1(A_1 \bowtie B^{A_1}_{1s}) + c_{21}(B^{A_1}_{1s}); \\
&T_{13}(A^{B_2}_{2j}) = t_2(A^{B_2}_{2s} \bowtie D) + c_{12}(A^{B_2}_{2s}); \\
&T_{21}(P^{B_1}_{A_1}) = t_2(\pi_{A_1 \cap B_1}(B_1)); \quad T_{21}(P^{A_2}_{B_2}) = t_1(\pi_{A_2 \cap B_2}(A_2)); \\
&T_{22}(A^{B_1}_{1s}) = t_1(A_1 \propto B_1) + c_{21}(P^{B_1}_{A_1}); \\
&T_{22}(B^{A_2}_{2s}) = t_2(B_2 \propto A_2) + c_{12}(P^{A_2}_{B_2}); \\
&T_{23}(A^{B_1}_{1j}) = t_2(A^{B_1}_{1s} \bowtie B_1) + c_{12}(A^{B_1}_{1s}); \\
&T_{23}(A^{B_2}_{2j}) = t_1(A_2 \bowtie B^{A_2}_{2s}) + c_{21}(B^{A_2}_{2s}).
\end{aligned}$$

$$x_0 = \begin{pmatrix} A_1, A_2 \\ B_1, B_2 \end{pmatrix} \rightarrow$$

$$\rightarrow \left\{ \begin{array}{l}
x_{11} = \begin{pmatrix} A_1, A_2, P^{A_1}_{B_1} \\ B_1, B_2, P^{B_2}_{A_2} \end{pmatrix} \xrightarrow[P^{B_2}_{A_2}\,2:1]{P^{A_1}_{B_1}\,1:2} x_{12} = \begin{pmatrix} A_1, A_2, A^{B_2}_{2s} \\ B_1, B_2, B^{A_1}_{1s} \end{pmatrix} \xrightarrow[A^{B_2}_{2s}\,1:2]{B^{A_1}_{1s}\,2:1} \\
\qquad \xrightarrow[A^{B_2}_{2s}\,1:2]{B^{A_1}_{1s}\,2:1} x_{13} = \begin{pmatrix} A_1, A_2, A^{B_1}_{1j} \\ B_1, B_2, A^{B_2}_{2j} \end{pmatrix} \\
x_{21} = \begin{pmatrix} A_1, A_2, P^{A_2}_{B_2} \\ B_1, B_2, P^{B_1}_{A_1} \end{pmatrix} \xrightarrow[P^{B_1}_{A_1}\,2:1]{P^{A_2}_{B_2}\,1:2} x_{22} = \begin{pmatrix} A_1, A_2, A^{B_1}_{1s} \\ B_1, B_2, B^{A_2}_{2s} \end{pmatrix} \xrightarrow[B^{A_2}_{2s}\,2:1]{A^{B_1}_{1s}\,1:2} \\
\qquad \xrightarrow[B^{A_2}_{2s}\,2:1]{A^{B_1}_{1s}\,1:2} x_{23} = \begin{pmatrix} A_1, A_2, A^{B_2}_{2j} \\ B_1, B_2, A^{B_1}_{1j} \end{pmatrix}
\end{array} \right.$$

Fig. 2.2.

We can obtain the solution for the problem similar to the single-join, single-query case, it is a linear programming problem.

3. Multiple join models

3.1. Join of three relations

In this section we extend the state-transition model for queries involving more than one join.

Let the query be

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

and the initial state is

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

There are three different execution sequences, in which we can execute the query Q_3:

$$QS_1 = (A \bowtie B) \bowtie C;$$
$$QS_2 = A \bowtie (B \bowtie C);$$
$$QS_3 = (A \bowtie C) \bowtie B.$$

We suppose, that relations A and C have no common attributes, so we will not take it into account. For sequence 1 the state-transition graph is shown in Fig. 3.1. The state-transition graph for sequence 2 may be obtained in the same manner.

Theorem 3.1. *The stochastic query optimization model for the join of three relations solved with semijoins defines a nonlinear programming problem.*

Proof. In state x_0 we can choose between two strategies for compute $A \bowtie B$ with semijoins: in the case of the first strategy we compute $B \propto A$ in site 2 (see state x_{12} with probability $p_{0,11}$), in the case of the second strategy we compute $A \propto B$ in site 1 (see state x_{22} with probability $p_{0,21}$ ($p_{0,11} + p_{0,21} = 1$)). In state x_{13} we have $AB = A \bowtie B$ in site 1, relation C is located in site 1, too, so the join $AB \bowtie C$ takes place locally, it is the only strategy with probability 1. In state x_{23} the relation AB is located in site 2 and relation C in site 1, so we can choose between two strategies: to calculate $AB \propto C$ in site 2 (see state x_{32} with probability $p_{23,31}$) or to calculate $C \propto AB$ in site 1 (see state x_{42} with probability $p_{23,41}$ ($p_{23,31} + p_{23,41} = 1$)). The notation for state x_{ik} is: i is for the strategy and k is the number of the state within one strategy.

The stochastic query optimization problem for QS_1 is given by

$$\begin{aligned}\tau_1 &= (T_{11}(P_B^A) + T_{13}(A_j^B) + T_{14}(AB_j^C))p_{0,11} + T_{22}(A_s^B)p_{0,21} + \\ &\quad + (T_{31}(P_{AB}^C) + T_{33}(AB_j^C))p_{0,21}p_{23,31} + T_{42}(C_s^{AB})p_{0,21}p_{23,41} \le \Delta,\\ \tau_2 &= T_{12}(B_s^A)p_{0,11} + (T_{21}(P_A^B) + T_{23}(A_j^B))p_{0,21} + T_{32}(AB_s^C)p_{0,21}p_{23,31} + \\ &\quad + (T_{41}(P_C^{AB}) + T_{43}(AB_j^C))p_{0,21}p_{23,41} \le \Delta,\end{aligned}$$

$$\begin{gathered} p_{0,11} + p_{0,21} = 1,\\ p_{23,31} + p_{23,41} = 1,\\ \min \Delta,\end{gathered}$$

where

$$\begin{aligned} T_{11}(P_B^A) &= t_1(\pi_{A\cap B}(A));\\ T_{21}(P_A^B) &= t_2(\pi_{A\cap B}(B));\\ T_{12}(B_s^A) &= t_2(B \propto A) + c_{12}(P_B^A);\\ T_{22}(A_s^B) &= t_1(A \propto B) + c_{21}(P_A^B);\\ T_{13}(A_j^B) &= t_1(A \bowtie B_s^A) + c_{21}(B_s^A);\\ T_{23}(A_j^B) &= t_2(A_s^B \bowtie B) + c_{12}(A_s^B);\\ AB &= A \bowtie B;\\ T_{31}(P_{AB}^C) &= t_1(\pi_{AB\cap C}(C));\\ T_{41}(P_C^{AB}) &= t_2(\pi_{AB\cap C}(AB));\\ T_{32}(AB_s^C) &= t_2(AB \propto C) + c_{12}(P_{AB}^C);\\ T_{42}(C_s^{AB}) &= t_1(C \propto AB) + c_{21}(P_C^{AB});\\ T_{33}(AB_j^c) &= t_1(AB_s^C \bowtie C) + c_{21}(AB_s^C);\\ T_{43}(AB_j^C) &= t_2(AB \bowtie C_s^{AB}) + c_{12}(C_s^{AB}).\end{aligned}$$

This is a special kind of nonlinear programming problem. The problem has five unknowns: $p_{0,11}, p_{0,21}, p_{23,31}, p_{23,41}$ and Δ and its overload constraints are multilinear of degree two.

We propose an algorithm for solving this nonlinear programming problem in the Appendix. We illustrate the results obtained applying the algorithm through some examples.

The strategy with semijoin is efficient in the case of wide area networks, where the transmission cost is dominant, in the sense that the transmission speed is much smaller, than the speed of local read/write. In the computation of the results we omit the local costs. In communication networks, where the transmission speed is appropriate to the speed of local read/write, the strategy with simple join is better. We suppose a communication network with a speed of 60Kbits/s. We propose three relations: A, B and C with different length in bits. The strategy with semijoin is better than the strategy with join if the operand relations have dangling tuples, so we suppose that the relations have some dangling tuples.

For every case we estimate the length in bits of relations: $\pi_{A\cap B}(A)$, $\pi_{A\cap B}(B), A \propto B$, $B \propto A$, $A \bowtie B$, $\pi_{AB\cap C}(C)$, $\pi_{AB\cap C}(AB)$, $C \propto AB$, $AB \propto$ $\propto C$. A DBMS can take these values from the statistics tables of the database. We calculate for these relations the necessary time to transmit them through the communication network in seconds, so we obtain the value for Δ in seconds, too. The results obtained appear in the table from Fig. 3.3.

Let us analyze the results:

- in case a) the relations from site 1 are much larger than relation B from site 2, even if they have some dangling tuples, so the chosen strategy is: from state x_0 the system undergoes transition in state x_{11}, then in states x_{12}, x_{13}, x_{14}. The values for probabilities $p_{23,31}$ and $p_{23,41}$ are not interesting. The necessary time to execute the query is $26,77$ seconds;

- case b) is the inverse of case a), the relation from site 2 is larger than the two relations from site 1. The obtained result is the corresponding one, so the chosen strategy is: from state x_0 the system undergoes transition in state x_{21}, then in states x_{22}, x_{23}, where it is chosen state x_{41}, because the result relation $AB = A \bowtie B$ will be too much larger than relation C, the other operand in the join operation;

- cases c), d) and e) give examples, when the strategy is not a "pure" one, so the strategies will be chosen with the given probabilities, because the repartition of the date is uniform enough between the two sites.

To show the efficiency of the strategy with semijoin, when the number of dangling tuples grows, we calculate for case c), f) and other cases, where the sizes of relations are the same, only the number of dangling tuples grows, the strategy with semijoin and the strategy with simple join. The strategy with simple join is given in [4]. The sizes of relations are: A - 500.000 bits, B - 1.000.000 bits, C - 500.000 bits. In the table from Fig. 3.2 we give the results for the strategies with semijoin and in the last row of the table are the results for the strategy with join for this example. We can see in the table, if the number of dangling tuples grows, the value for Δ decreases in the case of the strategy with semijoin. If we compare the value for Δ obtained in the case of strategy with join (8.33) and the lowest value for strategy with semijoin (3.32), we can see the efficiency of the strategy with semijoin, if there are a lot of dangling tuples.

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

$$\rightarrow \begin{cases} x_{11} = \begin{pmatrix} A, C, P_B^A \\ B \end{pmatrix} \xrightarrow{P_B^A 1:2} x_{12} = \begin{pmatrix} A, C \\ B, B_s^A \end{pmatrix} \xrightarrow{B_s^A 2:1} x_{13} = \begin{pmatrix} A, C, A_j^B \\ B \end{pmatrix} \rightarrow \\ x_{21} = \begin{pmatrix} A, C \\ B, P_A^B \end{pmatrix} \xrightarrow{P_A^B 2:1} x_{22} = \begin{pmatrix} A, C, A_s^B \\ B \end{pmatrix} \xrightarrow{A_s^B 1:2} x_{23} = \begin{pmatrix} A, C \\ B, A_j^B \end{pmatrix} \rightarrow \end{cases}$$

$$\rightarrow x_{14} = \begin{pmatrix} A, C, AB, AB_j^C \\ B \end{pmatrix}$$

$$\rightarrow \begin{cases} x_{31} = \begin{pmatrix} A, C, P_{AB}^C \\ B, AB \end{pmatrix} \xrightarrow{P_{AB}^C 1:2} \\ x_{41} = \begin{pmatrix} A, C \\ B, AB, P_C^{AB} \end{pmatrix} \xrightarrow{P_C^{AB} 2:1} \end{cases}$$

$$\xrightarrow{P_{AB}^C 1:2} x_{32} = \begin{pmatrix} A, C \\ B, AB, AB_s^C \end{pmatrix} \xrightarrow{AB_s^C 2:1} x_{33} = \begin{pmatrix} A, C, AB_j^C \\ B, AB \end{pmatrix}$$

$$\xrightarrow{P_C^{AB} 2:1} x_{42} = \begin{pmatrix} A, C, C_s^{AB} \\ B, AB \end{pmatrix} \xrightarrow{C_s^{AB} 1:2} x_{43} = \begin{pmatrix} A, C \\ B, AB, AB_j^C \end{pmatrix}$$

Fig. 3.1

This method gives a globally optimal strategy to execute a multiple join using semijoin.

Case	Nr. of dangling tuples			$p_{0,11}$	$p_{0,21}$	$p_{23,31}$	$p_{23,41}$	Δ
	Rel. A	Rel. B	Rel. C					
1.	-	250	500	0.388	0.612	0	1	8.42
2.	500	500	250	0.512	0.488	0	1	7.39
3.	750	500	500	0.355	0.645	0	1	6.55
4.	500	1000	500	1	0	0	1	3.33
5.	1000	1000	1000	0	1	0	1	3.32
Join	-	-	-	0.5	0.5	0	1	8.33

Fig. 3.2

3.2. Full reducer program

In this paragraph we consider the same problem as in the previous one. The ideal condition for calculating a multiple join is to be no dangling tuples in any of the relations participating in the query. In this case every tuple we ship between sites is essential. We call A reduced with respect to the relations B, C if $A = \pi_A(A \bowtie B \bowtie C)$.

We proposed, that relations A and C have no common attributes, so the corresponding hypergraph of the query is acyclic, we can apply the full reducer program for this query, see [8]. We will give the stochastic query optimization model for this case of full reducers. See the model at Figure 3.4.

In the state x_0 we can eliminate either relation A as ear, or relation C. If we eliminate relation A first, the remaining hypergraph is $\{B, C\}$. In this point (x_{12}) we can choose relation B as ear, so in the next step we eliminate it. The full reducer program in this case is

$$\begin{aligned} B &:= B \propto A, \\ C &:= C \propto B, \\ B &:= B \propto C, \\ A &:= A \propto B. \end{aligned}$$

In Fig.3.4 in every state we indicate only the operation we have to perform in the corresponding state. If in the second point we choose C as ear, the full reducer program is

$$\begin{aligned} B &:= B \propto A, \\ B &:= B \propto C, \\ C &:= C \propto B, \\ A &:= A \propto B. \end{aligned}$$

In state x_0 we can choose C as ear In this case the remaining hypergraph is $\{A, B\}$. In this point (x_{22}) we can choose relation A as ear, and the corresponding full reducer program is

$$\begin{aligned} B &:= B \propto C, \\ B &:= B \propto A, \\ A &:= A \propto B, \\ C &:= C \propto B. \end{aligned}$$

case	Relation A			Relation B		
	Nr. of Records	Nr. of bits	Nr. of dangling records	Nr. of Records	Nr. of bits	Nr. of dangling records
a)	12.500	10.000.000	2.500	2.500	1.000.000	-
b)	1.250	1.000.000	-	12.500	5.000.000	2.500
c)	1.250	500.000	-	1.250	1.000.000	250
d)	2.500	1.000.000	-	1.250	1.000.000	-
e)	2.500	2.000.000	-	1.250	500.000	-
f)	1.250	500.000	500	1.250	1.000.000	500

case	Relation C			$p_{0,11}$	$p_{0,21}$	$p_{23,31}$	$p_{23,41}$	Δ
	Nr. of Records	Nr. of bits	Nr. of Records					
a)	5.000	2.000.000	2.000	1	0	0	1	26,770
b)	5.000	2.000.000	2.000	0	1	0	1	26,600
c)	1.250	500.000	500	0,388	0,612	0	1	8,428
d)	1.250	500.000	-	0,273	0,727	0	1	13,025
e)	1.250	500.000	-	0,875	0,125	0	1	8,120
f)	1.250	500.000	250	0,512	0,488	0	1	7,390

Fig. 3.3.

$$x_0 = \begin{pmatrix} A,C \\ B \end{pmatrix} \rightarrow \begin{cases} x_{11} = \begin{pmatrix} P_B^A \\ \ \end{pmatrix} \longrightarrow x_{12} = \begin{pmatrix} \ \\ B_s^A \end{pmatrix} \longrightarrow \\ x_{21} = \begin{pmatrix} P_B^C \\ \ \end{pmatrix} \longrightarrow x_{22} = \begin{pmatrix} \ \\ B_s^C \end{pmatrix} \longrightarrow \end{cases}$$

$$\rightarrow \begin{cases} x_{31} = \begin{pmatrix} \ \\ P_C^B \end{pmatrix} \rightarrow x_{32} = \begin{pmatrix} C_s^B \\ \ \end{pmatrix} \rightarrow x_{33} = \begin{pmatrix} P_B^C \\ \ \end{pmatrix} \rightarrow x_{34} = \begin{pmatrix} \ \\ B_s^C \end{pmatrix} \rightarrow \\ x_{41} = \begin{pmatrix} P_B^C \\ \ \end{pmatrix} \rightarrow x_{42} = \begin{pmatrix} \ \\ B_s^C \end{pmatrix} \rightarrow x_{43} = \begin{pmatrix} \ \\ P_C^B \end{pmatrix} \rightarrow x_{44} = \begin{pmatrix} C_s^B \\ \ \end{pmatrix} \rightarrow \end{cases}$$

$$\rightarrow \begin{cases} x_{51} = \begin{pmatrix} P_B^A \\ \ \end{pmatrix} \rightarrow x_{52} = \begin{pmatrix} \ \\ B_s^A \end{pmatrix} \rightarrow x_{53} = \begin{pmatrix} \ \\ P_A^B \end{pmatrix} \rightarrow x_{54} = \begin{pmatrix} A_s^B \\ \ \end{pmatrix} \rightarrow \\ x_{61} = \begin{pmatrix} \ \\ P_A^B \end{pmatrix} \rightarrow x_{62} = \begin{pmatrix} A_s^B \\ \ \end{pmatrix} \rightarrow x_{63} = \begin{pmatrix} P_B^A \\ \ \end{pmatrix} \rightarrow x_{64} = \begin{pmatrix} \ \\ B_s^A \end{pmatrix} \rightarrow \end{cases}$$

$$\begin{aligned}
&\longrightarrow x_{35} = \begin{pmatrix} \\ P_A^B \end{pmatrix} \longrightarrow x_{36} = \begin{pmatrix} A_s^B \\ \\ \end{pmatrix}\\
&\longrightarrow x_{45} = \begin{pmatrix} \\ P_A^B \end{pmatrix} \longrightarrow x_{46} = \begin{pmatrix} A_s^B \\ \\ \end{pmatrix}\\
&\longrightarrow x_{55} = \begin{pmatrix} \\ P_C^B \end{pmatrix} \longrightarrow x_{56} = \begin{pmatrix} C_s^B \\ \\ \end{pmatrix}\\
&\longrightarrow x_{65} = \begin{pmatrix} \\ P_C^B \end{pmatrix} \longrightarrow x_{66} = \begin{pmatrix} C_s^B \\ \\ \end{pmatrix}
\end{aligned}$$

Fig. 3.4

If in the state x_{22} we choose B as ear, the corresponding full reducer program is

$$\begin{aligned}
B &:= B \propto C,\\
A &:= A \propto B,\\
B &:= B \propto A,\\
C &:= C \propto B.
\end{aligned}$$

In the case of every strategy the final state is

$$\begin{pmatrix} A_s^B, & C_s^B \\ B_s^A, & B_s^C \end{pmatrix}.$$

The stochastic query optimization problem is

$$\begin{aligned}
\tau_1 = {} & T_{11}(P_B^A)p_{0,11} + T_{21}(P_B^C)p_{0,21} + (T_{32}(C_s^B) + T_{33}(A_s^B) +\\
& T_{36}(A_s^B))p_{0,11}p_{12,31} + (T_{41}(P_B^C) + T_{44}(C_s^B) + T_{46}(A_s^B))p_{0,11}p_{12,41} +\\
& + (T_{51}(P_B^A) + T_{54}(A_s^B) + T_{56}(C_s^B))p_{0,21}p_{22,51} +\\
& + (T_{62}(A_s^B) + T_{63}(P_B^A) + T_{66}(C_s^B))p_{0,21}p_{22,61} \le \Delta,\\
\tau_2 = {} & T_{12}(B_s^A)p_{0,11} + T_{22}(B_s^C)p_{0,21} + (T_{31}(P_C^B) + T_{34}(B_s^C) +\\
& + T_{35}(P_A^B))p_{0,11}p_{12,31} + (T_{42}(B_s^C) + T_{43}(P_C^B) + T_{45}(P_A^B))p_{0,11}p_{12,41} +\\
& + (T_{52}(B_s^A) + T_{53}(P_A^B) + T_{55}(P_C^B))p_{0,21}p_{22,51} +\\
& + (T_{61}(P_A^B) + T_{64}(B_s^A) + T_{65}(P_C^B))p_{0,21}p_{22,61} \le \Delta,
\end{aligned}$$

$$\begin{aligned}
p_{0,11} + p_{0,21} &= 1,\\
p_{12,31} + p_{12,41} &= 1,\\
p_{22,51} + p_{22,61} &= 1,
\end{aligned}$$

$$\min \Delta,$$

where

$$\begin{aligned}
&T_{11}(P_B^A) = t_1(\pi_{A\cap B}(A));\ T_{21}(P_B^C) = t_1(\pi_{B\cap C}(C));\\
&T_{12}(B_s^A) = t_2(B \propto A) + c_{12}(P_B^A);\ T_{22}(B_s^C) = t_2(B \propto C) + c_{21}(P_B^C);\\
&T_{31}(P_C^B) = t_2(\pi_{B\cap C}(B));\ T_{41}(P_B^C) = t_1(\pi_{B\cap C}(C));\\
&T_{51}(P_B^A) = t_1(\pi_{A\cap B}(A));\ T_{61}(P_A^B) = t_2(\pi_{A\cap B}(B));\\
&T_{32}(C_s^B) = t_1(C \propto B) + c_{21}(P_C^B);\ T_{42}(B_s^C) = t_2(B \propto C) + c_{12}(P_B^C);\\
&T_{52}(B_s^A) = t_2(B \propto A) + c_{12}(P_B^A);\ T_{62}(A_s^B) = t_1(A \propto B) + c_{21}(P_A^B);\\
&T_{33}(P_B^C) = t_1(\pi_{B\cap C}(C));\ T_{43}(P_C^B) = t_2(\pi_{B\cap C}(B));\\
&T_{53}(P_A^B) = t_2(\pi_{A\cap B}(B));\ T_{63}(P_B^A) = t_1(\pi_{A\cap B}(A));\\
&T_{34}(B_s^C) = t_2(B \propto C) + c_{12}(P_B^C);\ T_{44}(C_s^B) = t_1(C \propto B) + c_{21}(P_C^B);\\
&T_{54}(A_s^B) = t_1(A \propto B) + c_{21}(P_A^B);\ T_{64}(B_s^A) = t_2(B \propto A) + c_{12}(P_B^A);\\
&T_{35}(P_A^B) = t_2(\pi_{A\cap B}(B));\ T_{45}(P_A^B) = t_2(\pi_{A\cap B}(B));\\
&T_{55}(P_C^B) = t_2(\pi_{B\cap C}(B));\ T_{65}(P_C^B) = t_2(\pi_{B\cap C}(B));\\
&T_{36}(A_s^B) = t_1(A \propto B) + c_{21}(P_A^B);\ T_{46}(A_s^B) = t_1(A \propto B) + c_{21}(P_A^B);\\
&T_{56}(C_s^B) = t_1(C \propto B) + c_{21}(P_C^B);\ T_{66}(C_s^B) = t_1(C \propto B) + c_{21}(P_C^B).
\end{aligned}$$

The solution of the nonlinear programming problem gives a global optimum for the execution of the full reducer program. The problem can be solved with the algorithm given in the appendix.

After this program the final state is the same for every strategy and for calculating the join QS_1 we can apply the global optimization model for multiple join from [4].

Appendix

Let (X, d) be a compact metric space and $f_1, \ldots, f_p : X \to R_+$ continuous, strictly positive functions. We have the next problem:

$$(P) \qquad \begin{aligned}
&f_1(x) \le y,\\
&\vdots\\
&f_p(x) \le y,\\
&\min y, \quad y \in R,\ y > 0.
\end{aligned}$$

Theorem 1. *The problem (P) has at least one solution.*

Proof. Let $f : X \to R$ be the function defined by $f(x) = \max\{f_1(x), \ldots, f_p(x)\}$. Because the function f is continuous and X is a compact metric space, then using the Weierstrass theorem, there exists a point $x_0 \in X$ such that $f(x_0) = \min\limits_{x \in X} f(x)$. 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 satisfy the inequalities from the problem (P) for $x_0^* \in X$, i.e.

$$\begin{gathered} f_1(x_0^*) \le y_0, \\ \vdots \\ f_p(x_0^*) \le y_0. \end{gathered}$$

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

Now we give an algorithm, which gives the solution of the problem (P).

Let $A_1 \subseteq A_2 \subset A_3 \subset \ldots \subset A_n \subset \ldots$ be a sequence of finite subsets of X such that $\bigcup\limits_{n=1}^{\infty} A_n$ is dense in X, thus

$$\begin{gathered} A_1 = \{x_1^1, \ldots, x_{q_1}^1\}, \\ A_2 = \{x_1^2, \ldots, x_{q_2}^2\}, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ A_n = \{x_1^n, \ldots, x_{q_n}^n\}. \\ \ldots\ldots\ldots\ldots\ldots\ldots \end{gathered}$$

We calculate

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

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

Theorem 2. *The sequence $\{y_n\}_{n \in N^*}$ converges to the solution of the problem (P).*

Proof. We suppose that $y_n \to y^* > f(x_0)$. Because the set $\bigcup\limits_{n=1}^{\infty} A_n$ is dense in X and the function f is continuous results, that there exists a sequence $\{x_n\} \subset \bigcup\limits_{n=1}^{\infty} A_n$ such that $x_n \to x_0$ and $f(x_n) \to f(x_0)$. Without loss of generality we suppose that $x_1 \in A_1, \dots, x_n \in A_n, \dots$. 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 presumtion $y^* > f(x_0)$.

Remark. From the algorithm we see that for every $n \in N^*$, there exists $i_n \in \{1, \dots, q_n\}$ such that $y_n = \max\{f_1(x_{i_n}^n), \dots, f_p(x_{i_n}^n)\}$. 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).

Acknowledgement. The third author would like to thank Professor A. Benczúr 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. and Drenick R.F.,** A design theory for multi-processing computing systems, *Large Scale Syst.,* **12** (1987), 155-172.

[4] **Drenick P.E. and Smith E.J.,** Stochastic query optimization in distributed database, *ACM Trans. on Database Systems*, **18** (2) (1993), 262-288.

[5] **LaFortune S. and Wong E.,** A state transition model for distributed query processing, *ACM Trans. on Database Systems,* **11** (3) (1986), 294-322.

[6] **Özsu M.T. and Valduriez P.,** *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.

(Received February 8, 2000)

T. Márkus
Eötvös Loránd University
Budapest, Hungary

C. Moroşanu
A.I. Cuza University
Iaşi, Roumania

V. Varga
Babeş-Bolyai University
Cluj-Napoca, Roumania