STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLVIII**, Number 2, 2003

STOCHASTIC OPTIMIZATION OF QUERYING DISTRIBUTED DATABASES II. SOLVING STOCHASTIC OPTIMIZATION

D. DUMITRESCU, C. GROŞAN, AND V. VARGA

Abstract. General stochastic query optimization (GSQO) problem for multiple join — join of p relations which are stored at p different sites — is presented. GSQO problem leads to a special kind of nonlinear programming problem (P). Problem (P) is solved by using a constructive method. A sequence converging to the solution of the optimization problem is built. Two algorithms for solving optimization problem (P) are proposed.

Keywords Distributed Databases, Query Optimization Problem, Genetic Algorithms, Evolutionary Optimization, Adaptive Representation.

1. Introduction

The aim of this paper is to solve the general stochastic optimization problem for the join of p relations, stored at p different sites of a distributed database. In Part I the general stochastic optimization problem, was reduced to the following constrained nonlinear programming problem (P):

Let(X, d) be a compact metric space and

$$f_1, ..., f_p : X \to R_+$$

continuous, strictly positive functions.

The optimization problem (P) is thus:

$$(P) \begin{cases} \text{minimize } y, y \in R \\ \quad \text{subject to:} \\ y > 0, \\ f_1(x) \le y, \\ \vdots \\ f_p(x) \le y. \end{cases}$$

Received by the editors: July 3, 2003.

2000 *Mathematics Subject Classification.* 68P15, 68T99.

1998 *CR Categories and Descriptors.* C.2.4 [**Computer Systems Organization**]: Computer-Communication Networks – *Distributed Systems*; H.2.4 [**Information Systems**]: Database Management – *Systems*.

In this Part a constructive method to solve this problem is proposed. A theorem which demonstrates that the nonlinear optimization problem (P) has at least one solution is proved in Section 2.

The *Constructive Algorithm* (CA) given in Section 3 implements the method of Section 2. The *Refining Algorithm* (RA) can optimize the solution given by the Constructive Algorithm. RA starts with a minimum point x_{min} and searches for a better solution in the $[x_{\min} - \varepsilon, x_{\min}]$ interval and then in $[x_{\min}, x_{\min} + \varepsilon]$, where ε is a problem parameter.

2. A constructive method for solving general stochastic query problem

Now we are ready to give a constructive method for solving problem (P). This method generates a sequence converging to a solution of the problem (P). Theorem 2.1 ensures that the constructed sequence really converges towards a solution of the optimization problem (P).

Let $f : X \to R$ be the function defined by

$$f(x) = \max\{f_1(x), ..., f_p(x)\}.$$

and y_0 the global minimum value of the function f, i.e.

$$y_0 = \min_{x \in X} f(x).$$

Let $A_1 \subset A_2 \subset A_3 \subset \ ... \ \subset A_n \subset \ ...$ be a sequence of finite subsets of X such that $\overset{\infty}{\underset{n=1}{\cup}} A_n$ is dense (see for instance Rudin, 1976) in X, i.e. $\overline{\cup A_n} = X$ equivalent to the fact, that for $\forall x \in X$, $\exists x_n \in \underset{n \in N}{\cup} A_n$ such that $x_n \to x$.

We consider

$$\begin{aligned} A_1 &= \{u_1, u_2, ..., u_{q_1}\}, & u_i \in X, i = 1, \dots q_1, \\ A_2 &= \{v_1, v_2, ..., v_{q_2}\}, & v_j \in X, j = 1, \dots q_2, \\ &\vdots & \\ A_n &= \{w_1, w_2, ..., w_{q_n}\}, & w_k \in X, k = 1, \dots q_n, \end{aligned}$$

where $q_i \in N^*, i = 1, \dots, n$ and $q_n \to \infty$.

Let us consider the sequence $(y_n)_{n \geq 1}$ defined as folows:

$$\begin{aligned} y_1 &= \min\{\max\{f_1(u_1), f_2(u_1), ..., f_p(u_1)\}, ..., \max\{f_1(u_{q_1}), f_2(u_{q_1}), ..., f_p(u_{q_1})\}, \\ y_2 &= \min\{\max\{f_1(v_1), f_2(v_1), ..., f_p(v_1)\}, ..., \max\{f_1(v_{q_2}), f_2(v_{q_2}), ..., f_p(v_{q_2})\}, \\ &\vdots \\ y_n &= \min\{\max\{f_1(w_1), f_2(w_1), ..., f_p(w_1)\}, ..., \max\{f_1(w_{q_n}), f_2(w_{q_n}), ..., f_p(w_{q_n})\}. \end{aligned}$$

It is easy to see that sequence $(y_n)_{n \geq 1}$ is monotone decreasing and bounded. Therefore the sequence is convergent.

With respect to the convergent sequence $(y_n)_{n\geq 1}$ we can state the following Theorem.

Theorem 2.1 The sequence $(y_n)_{n\geq 1}$ converges to a solution of the problem (P).

Proof. We have

$$y_n \geq f(x_0)$$

because x_0 is the global minimum of the function f. Therefore, if $y_n \to y^*$ we have

$$y^* \geq f(x_0).$$

We distinguish two cases. First case corresponds to

$$y^* = f(x_0).$$

In this case is nothing to demonstrate. The second case corresponds to the situation

$$y^* > f(x_0).$$

We prove that this case it is impossible.

Because the set $\overset{\infty}{\underset{n=1}{\cup}} A_n$ is dense in X and the function f is continuous it results that there exists a sequence $(x_n) \subset \overset{\infty}{\underset{n=1}{\cup}} A_n$ such that

$$x_n \to x_0 \quad \text{and} \quad f(x_n) \to f(x_0).$$

Without loss of generality we may suppose that

$$x_1 \in A_1, ..., x_n \in A_n, ...$$

But we have:

$$y_n = \min\{\max\{f_1(w_1), \ldots, f_p(w_1)\}, \ldots, \max\{f_1(w_{q_n}), \ldots, f_p(w_{q_n})\}\}$$

and

$$f(x_n) = \max\{f_1(x_n), \ldots, f_p(x_n)\}$$

Therefore we have:

$$f(x_n) \geq y_n,$$

for every $n \in N^*$.

If $n \to \infty$ we have $f(x_n) \to f(x_0)$ and $y_n \to y^*$, so we obtain

$$f(x_0) \geq y^*,$$

which is a contradiction with the assumption $y^* > f(x_0)$. Therefore we obtained $y^* = f(x_0)$. This completes the proof. □

Remark. From the construction above we can see that for every $n \in N^*$, there exists an index $i_n \in \{1, ..., q_n\}$such that

$$y_n = \max\{f_1(w_{i_n}), ..., f_p(w_{i_n})\}.$$

In this way we obtain a sequence $(w_{i_n})_{n\geq 1}$. It is obvious that each accumulation point of the sequence $(w_{i_n})_{n\geq 1}$ is a solution of the problem (P).

3. Solving problem (P_p) using the proposed constructive method

In the case of solving problem (P_p) using Theorem 2.1 we have

$$X = [0,1]^n .$$

In order to obtain an approximate solution of problem (P_p) in the Constructive Algorithm we take a uniform grid G of the hypercube $[0,1]^k$.

We may choose the sets $(A_i)_{i \in N^*}$ in the folowing way:

$$\begin{aligned}
A_1 &= \left\{ \left(\frac{i_0}{n}, \frac{i_1}{n}, \ldots, \frac{i_n}{n} \right) | i_0, i_1, \ldots, i_n \in \{0, 1, \ldots, n\}, i_0 < i_1 < \cdots < i_n \right\}, \\
A_2 &= \left\{ \left(\frac{j_0}{2n}, \frac{j_1}{2n}, \ldots, \frac{j_{2n}}{2n} \right) | j_0, j_1, \ldots, j_{2n} \in \{0, 1, \ldots, 2n\}, j_0 < j_1 < \cdots < j_{2n} \right\}, \\
&\vdots \\
A_k &= \left\{ \left(\frac{l_0}{2^{k-1}n}, \frac{l_1}{2^{k-1}n}, \ldots, \frac{l_{2^{k-1}n}}{2^{k-1}n} \right) | l_0, l_1, \ldots, l_{2^{k-1}n} \in \left\{0, 1, \ldots, 2^{k-1}n\right\}, \right. \\
&\quad \left. l_0 < l_1 < \ldots < l_{2^{k-1}n} \right\}.
\end{aligned}$$

Our grid is that induced by $A_1, A_2, \ldots, A_k$. The sets $(A_i)_{i \in N^*}$ constructed in the above way verify the conditions of Theorem 5.1 of Part I of this paper. For our purposes we may consider $n = 10$.

For each point of the grid G we compute the values $f_s, s = 1, \ldots, p$. Choosing the maximum$f_s, s = 1, \ldots, p$, we ensure that each inequality in the problem (P_p) holds. Problem solution will be the minimum of all selected maximums.

The previous considerations enable us to formulate an algorithm for solving problem (P_p). This technique will be called *Constructive Algorithm* (CA) and may be outlined as below.

Constructive Algorithm

Input:
    n // the number of divisions;
    Functions f_1,f_2...,f_p // express the problem constraints.
begin
Initializations:
    $h = \frac{1}{n}$ // the length of one division;
    $valx_j = 0, j = 1, ..., k$ // initial values for x_j;
    for $s = 1$ **to** p **do** // initial values for functions f_s
        $valf_s = f_s(valx_1,\ valx_2, \ldots ,valx_k)$
    end for
    $valmax = \max\{valf_s, s = 1, ..., p\}$
    $valmin = valmax$

```
    for j = 1 to k do            // in xmin_j we store the x_j values for which we
      xmin_j = valx_j            // have the minimum of f_s
    end for
Constructing the grid:
    for i_1 = 1 to n do
      valx_1 = i_1 * h
      for i_2 = 1 to n do
        valx_2 = i_2 * h
          ⋮
          for i_k = 1 to n do
            valx_k = i_k * h
            for s = 1 to p do        // calculate the values for functions f_s for
              valf_s = f_s(valx_1, valx_2,...,valx_k)     // the current values of x_j
            end for
            valmax = max{valf_s, s = 1,...,p}
            if (valmax < valmin) then
              valmin = valmax
              for j = 1 to k do          // store in xmin_j the new x_j values
                xmin_j = valx_j          // for which we have the
              end for                    // minimum of f_s
            end if
          end for // i_k
            ⋮
        end for // i_2
      end for // i_1
end
```

Remark. *valmin* denote the minimum value of Δ_1 from problem (P_p) and $xmin_j$, $j = 1, ..., k$ denote the values for x_j, $j = 1, ..., k$ for which the minimum is reached.

The Constructive Algorithm should be repeated for a new value of n, so that the divisions have to include the old divisions, in this way we obtain a new subset A_i of the set X.

Solution obtained by the Constructive Algorithm can be refined using the *Refining Algorithm* (RA).

Let us denote by $(x_{\min 1}, x_{\min 2}, ..., x_{\min k})$ the minimum point obtained by the Constructive Algorithm. Let us define the vectors $x_{\min} - \varepsilon$, $x_{\min} + \varepsilon$:

$$\begin{aligned} x_{\min} - \varepsilon &= (x_{\min 1} - \varepsilon, x_{\min 2} - \varepsilon, ..., x_{\min k} - \varepsilon), \\ x_{\min} + \varepsilon &= (x_{\min 1} + \varepsilon, x_{\min 2} + \varepsilon, ..., x_{\min k} + \varepsilon) \quad . \end{aligned}$$

Initially Refining Algorithm searches for a better minim in the interval: $[x_{\min} - \varepsilon, x_{\min}]$. Then it searches in $[x_{\min}, x_{\min} + \varepsilon]$, where ε is a problem parameter. In case of found a better minim (to the left, or to the right) the algorithm will continue to search refining the grid by division by 2. Let *IterNr* be the maximum allowed number of iterations.

Refining Algorithm can be outlined as follows.

Refining Algorithm

```
Input:
    n                          // the number of divisions;
    eps                        // the accepted error;
    IterNr                     // the number of iterations;
    xmin_j, j = 1,...,k        // a minimum point obtained with algorithm CA;
Initializations:
    h = 1/n                    // the length of one division;
    for s = 1 to p do          // values for functions f_s;
      valf_s = f_s(xmin_1, xmin_2,...,xminx_k)
    end for
    valmin = max{valf_s, s = 1,...,p}
    for j = 1 to k do          // in xminr_j we store the x_j values for which we
      xminr_j = xmin_j         // have the minimum of f_s
    end for
Refining the minimum:
    while h >= eps do
      for iter = 1 to IterNr do
        for j = 1 to k do
          while found a better minimum to the left do
            if xmin_j - h > 0 then
              xmin_j = xmin_j - h
              valmax = max{f_s(xmin_1,..., xmin_k), s = 1,...,p}
              if (valmax < valmin) then
                valmin = valmax              // a new minimum was found;
                for j = 1 to k do            // store in xminr_j the new x_j
                  xminr_j = xmin_j           // values for which we have the
                end for // minimum of f_s;
                reloop while
              end if
            end if
          end while // found to the left
          while found a better minimum to the right do
            if xmin_j + h > 0 then
              xmin_j = xmin_j + h
              valmax = max{f_s(xmin_1,..., xmin_k), s = 1,...,p}
              if (valmax < valmin) then
```

```
          valmin = valmax                 // a new minimum was found;
          for j = 1 to k do               // store in xminr_j the new x_j
            xminr_j = xmin_j              // values for which we have the
          end for // minimum of f_s;
          reloop while
        end if
      end if
    end while // found to the right
   end for // j
  end for // iter
  h = h/2 // refine the division;
end while // h >= eps
```

Algorithms CA and RA can be used to solve the general stochastic optimization problem (P). The problem of four relations join is formulated as the problem (P_1) of Part I, which is a particularization of general problem (P).

Numerical experiments for solving problem (P_1) using the Constructive Algorithm and Refining Algorithm are presented in Part III.

References

[1] C. J. Date (2000): *An Introduction to Database Systems*, Addison-Wesley Publishing Company, Reading, Massachusetts.

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

[3] P. E. Drenick, R. F. Drenick (1987): *A design theory for multi-processing computing systems*, Large Scale Syst. Vol. 12, pp. 155-172.

[4] P. E. Drenick, E. J. Smith (1993): *Stochastic query optimization in distributed databases*, ACM Transactions on Database Systems, Vol. 18, No. 2, pp. 262-288.

[5] D. Dumitrescu, C. Grosan, M. Oltean (2001): *A new evolutionary adaptive representation paradigm*, Studia Universitas “Babes-Bolyai”, Seria Informatica, Volume XLVI, No. 1, pp. 15-30.

[6] C. Grosan, D. Dumitrescu (2002): *A new evolutionary paradigm for single and multiobjective optimization*, Seminar on Computer Science, “Babes-Bolyai” University of Cluj-Napoca.

[7] J. Kingdon,L. Dekker (1995): *The shape of space*, Technical Report, RN-23-95, Intelligent System Laboratories, Department of Computer Science, University College, London.

[8] S. LaFortune, E. Wong (1986): *A state transition model for distributed query processing*, ACM Transactions on Database Systems, Vol. 11, No. 3, pp. 294-322.

[9] T. Markus, C. Morosanu, V. Varga (2001): *Stochastic query optimization in distributed databases using semijoins*, Annales Universitatis Scientiarum Budapestinensis de Rolando Eötvös Nominatae Sectio Computatorica 20, pp. 107-131.

[10] M. T.Özsu, P. Valduriez (1999) : *Principles of Distributed Database Systems*, Prentice-Hall.

[11] R. Ramakrishnan (1998): *Database Management Systems* WCB McGraw-Hill.

[12] W. Rudin(1976): *Principles of Mathematical Analysis*, McGraw-Hill, New York.

[13] J. D. Ullman (1988): *Principles of Database and Knowledge-Base Systems,* Vol. I-II, Computer Science Press.

[14] V. Varga (1998): *Stochastic optimization for the join of three relations in distributed databases I. The theory and one application*, Studia Universitas "Babes-Bolyai", Seria Informatica, Volume XLIII, No. 2, pp. 37-46.

[15] V. Varga (1999): *Stochastic optimization for the join of three relations in distributed databases II. Generalization and more applications*, Studia Universitas "Babes-Bolyai", Seria Informatica, Volume XLIV, No. 1, pp. 55-62.

E-mail address: ddumitr@cs.ubbcluj.ro

E-mail address: cgrosan@cs.ubbcluj.ro

E-mail address: ivarga@cs.ubbcluj.ro

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania