STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **XLIX**, Number 1, 2004

STOCHASTIC OPTIMIZATION OF QUERYING DISTRIBUTED DATABASES III. EVOLUTIONARY METHOD VERSUS CONSTRUCTIVE METHOD

D. DUMITRESCU, C. GROŞAN, V. VARGA

Abstract. The stochastic query optimization problem for multiple join - join of p relations, stored at p different sites - leads to a special nonlinear programming problem. General optimization problem can be solved by using evolutionary methods. The problem of four joins is a particular case of the general query optimization problem. This particular problem is solved applying the Constructive Algorithm from Part II and the Adaptive Representation Evolutionary Algorithm. The results obtained by applying these two methods are compared. Two sets of constant values are used in this comparison.

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

1. Introduction

The general stochastic optimization problem for the join of p relations, stored at p different sites of a distributed database was presented in Part I of the paper. This optimization problem in distributed databases leads to a constrained nonlinear programming problem. In Part I a theorem, which proves, that nonlinear optimization problem has at least one solution is stated. In Part II a constructive method for solving the nonlinear programming problem is proposed.

In this Part of the paper the optimization problem is solved using an evolutionary method presented in Dumitrescu, Grosan and Oltean (2001), Grosan and Dumitrescu (2002). Section 3 describes the evolutionary technique called *Adaptive Representation Evolutionary Algorithm* (AREA). In section 4 the results obtained

Received by the editors: June 6, 2004.

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*.

by applying these different approaches are presented. Two sets of constant values are used in these experiments. The solutions obtained by the two approaches are very close. The CPU time required for solving the optimization problem by using evolutionary algorithm is less than the CPU time required by the constructive method. Considered stochastic model is compared with a popular heuristic method (Section 5).

2. Solving problem (P_p) using an evolutionary algorithm

In this section an evolutionary technique used for solving problem (P_p) is proposed.

Let us denote $g_i = x_i + x_{i+1}$.

General problem (P) can be reformulated as the following constrained optimization problem:

$$(P') \begin{cases} \text{minimize } y \\ \quad \text{subject to:} \\ y > 0, \\ f_i(x) \leq y, \ i = 1, \ldots, p, \\ g_i(x) - 1 = 0, \ i = 1, \ldots n, \\ x = (x_1, \ldots, x_n). \end{cases}$$

The evolutionary method for solving problem (P') implies the next steps:

Step 1. Determine maximum from the p functions f_1,..., f_p.

Let us denote by

$$f^*(x) = \max(f_i(x)), x \in X.$$

Step 2. Minimize the function f^* by using an evolutionary algorithm.

In this case, the fitness of the solution x will be:

$$\begin{aligned} eval(x) &= f^*(x) \\ &= \max_{i=1,\ldots,n} f_i(x), \ x \in X. \end{aligned}$$

Let us denote by x^* the obtained minimum.

Step 3. The solution y of the problem can be obtained by setting

$$y = x^* + \varepsilon,$$

where $\varepsilon > 0$, is a small number.

Remark. The constraints g_i were treated by considering each x_i, i not odd, as being $1 - x_i$.
The advantage of applying an evolutionary technique for solving problem (P') is that the involved function f^* is not necessary effectively computed. Only the values of function f^* for the candidate solution are needed.

3. Evolutionary techniques for solving problem (P)

An evolutionary algorithm called *Adaptive Representation Evolutionary Algorithm* (AREA) is proposed for solving problem (P'). AREA technique will be described in what follows.

3.1. **AREA technique.** The main idea of AREA technique is use a dynamical encoding allowing each solution be encoded over a different alphabet. This approach is similar to that proposed in Kingdon and Dekker (1995). Solution representation is adaptive and may be changed during the search process as the effect of mutation operator.

3.2. **AREA representation.** Each AREA individual consists of a pair $(x,\ B)$ where x is a string encoding object variables and B specifies the alphabet used for encoding x. B is an integer number such that $B \geq 2$ and x is a string of symbols over the alphabet $\{0,\ 1,\ \ldots,\ B-1\}$. If $B = 2$, the standard binary encoding is obtained.

Each solution has its own encoding alphabet. The alphabet over which x is encoded may be changed during the search process.

An example of AREA chromosome is the following:

$C = (301453,\ 6)$.

Remark. The genes of x may be separated by comma if required (for instance when $B \geq 10$).

3.3. **Search operator.** Within AREA mutation is the unique search operator. Mutation can modify object variables as well as the last chromosome position (fixing the representation alphabet).

When the changing gene belongs to the object variable sub-string (x – part of the chromosome), the mutated gene is a symbol randomly chosen from the same alphabet.

Example

Let us consider the chromosomes can be represented over the alphabets $B_2, \ldots, B_{30}$, where $B_i = \{0, 1, \ldots, i-1\}$.

Consider the chromosome C represented over the alphabet B_8:

$C = (631751,\ 8)$.

Consider a mutation occurs on the position 3 in the x part of the chromosome and the mutated value of the gene is 4. Then the mutated chromosome is:

$C_1 = (634751, 8)$.

If the position specifying the alphabet is changed, then the object variables will be represented using symbols over the new alphabet, corresponding to the mutated value of B.

Consider again the chromosome C represented over the alphabet B_8:

$C = (631751, 8)$.

Consider a mutation occurs on the last position and the mutated value is 5. Then the mutated chromosome is:

$C_2 = (23204032, 5)$.

C and C_2 encode the same value over two different alphabets (B_8 and B_{10}).

Remark. A mutation generating an offspring worst than its parent is called a *harmful mutation*. A constant called MAX_HARMFUL_MUTATIONS is used to determinate when the chromosome part represented the alphabet will be changed (mutated).

3.4. **AREA procedure.** During the initialization stage each AREA individual (chromosome, solution) is encoded over a randomly chosen alphabet. Each solution is then selected for mutation. If the offspring obtained by mutation is better than its parent than the parent is removed from the population and the offspring enters the new population. Otherwise, a new mutation of the parent is considered. If the number of successive harmful mutations exceeds a prescribed threshold (denoted by MAX_HARMFUL_MUTATIONS) then the individual representation (alphabet part) is changed and with this new representation it enters the new population.

The reason behind this mechanism is to dynamically change the individual representation whenever it is needed. If a particular representation has no potential for further exploring the search space then the representation is changed. In this way we hope that the search space will be explored more efficiently.

The AREA technique may be depicted as follows.

AREA technique

begin
Set $t = 0$;
Random initializes chromosome population P (t);
Set to zero the number of harmful mutations for each individual in $P(t)$;
while ($t <$ number of generations) **do**
  $P(t+1) = \emptyset$;

for $k = 1$ **to** PopSize **do**
Mutate the k^{th} chromosome from $P\ (t)$. An offspring is obtained.
Set to zero the number of harmful mutations for offspring;
if the offspring is better than the parent **then**
the offspring is added to $P(t+1)$;
else
Increase the number of harmful mutations for current individual;
if the number of harmful mutations for the current individual = MAX_HARMFUL_MUTATIONS **then**
Change the individual representation;
Set to zero the number of harmful mutations for the current individual;
Add individual to $P(t+1)$;
else
Add current individual (the parent) to $P(t+1)$;
end if
end if
end for;
Set $t = t + 1$;
end while;
end

4. Experimental results. Constructive algorithm versus AREA technique

In this section we consider two numerical experiments for solving problem (P_p) using Constructive Algorithm and evolutionary technique above described. Results obtained by applying AREA technique are compared with the results obtained by applying Constructive Algorithm (and after that, Refinement Algorithm).

The obtained results in the case of a communication network with a speed of $6 \cdot 10^4$ bps are presented. The model allows different transfer speed for the distinct connections. But in our experiment the transfer speed is assumed to be constant for each connection. In the Table 1 and Table 4 appear two cases and in every case the number of bits for every relation and the necessary time to transfer the relations through the network.

We have to approximate the size of B', where
$B' = A \bowtie B$
and the size of C', where
$C' = B' \bowtie C$.

	Number of bits	Transfer time
Relation A	8,000,000	133.33s
Relation B	4,000,000	66.66s
Relation C	10,000,000	166.66s
Relation D	5,000,000	83.33s
Relation B'	10,400,000	173.33s
Relation C'	25,600,000	426.66s

TABLE 1. Relation sizes and transfer times for Experiment 1.

The database management system can take these sizes from the database statistics. In our computation we ignore the local processing time, because it is unessential compared to the transmission time.

4.1. **Experiment 1.** Replacing the transfer times in formulas (3.1) of the Part I and ignoring the local processing time we obtain:

$T_{11}(B') = 66.66$,
$T_{12}(C') = 166.66$,
$T_{32}(C') = 166.66$,
$T_{13}(D') = 83.33$,
$T_{33}(D') = 83.33$,
$T_{53}(D') = 83.33$,
$T_{73}(D') = 83.33$,
$T_{21}(B') = 133.33$,
$T_{22}(C') = 173.33$,
$T_{42}(C') = 173.33$,
$T_{23}(D') = 426.66$,
$T_{43}(D') = 426.66$,
$T_{63}(D') = 426.66$,
$T_{83}(D') = 426.66$.

Let us consider the mean processing times at the four sites, given by the formulas 3.2 of the Part I.

AREA Parameters	
Population size	100
Number of generations	100
Mutation probability	0,01
Number of variable	14
Number of alphabets	30
MAX_HARMFUL_MUTATIONS	5

TABLE 2. Parameters used by AREA technique.

$$\begin{aligned}
\tau_1 &= 66.66p_{0,11} + 166.66p_{0,11}p_{11,12} + 83.33p_{0,11}p_{11,12}p_{12,13},\\
\tau_2 &= 133.33p_{0,21} + 166.66p_{0,21}p_{21,32} + 83.33p_{0,21}p_{21,32}p_{32,53},\\
\tau_3 &= 173.33p_{0,11}p_{11,22} + 173.33p_{0,21}p_{21,42} + 83.33p_{0,11}p_{11,22}p_{22,33}\\
&\quad + 83.33p_{0,21}p_{21,42}p_{42,73},\\
\tau_4 &= 426.66p_{0,11}p_{11,12}p_{12,23} + 426.66p_{0,11}p_{11,22}p_{22,43} +\\
&\quad + 426.66p_{0,21}p_{21,32}p_{32,63} + 426.66p_{0,21}p_{21,42}p_{42,83}.
\end{aligned} \tag{4.1}$$

From (4.1) we obtain the next values for constants $c_1, \ldots, c_{14}$ of Equations (4.1) of the Part I:

$c_1 = 66.66$,
$c_2 = 166.66$,
$c_3 = 83.33$,
$c_4 = 133.33$,
$c_5 = 166.66$,
$c_6 = 83.33$,
$c_7 = 173.33$,
$c_8 = 173.33$,
$c_9 = 83.33$,
$c_{10} = 83.33$,
$c_{11} = 426.66$,
$c_{12} = 426.66$,
$c_{13} = 426.66$,
$c_{14} = 426.66$.

Parameters used within AREA technique are given in Table 2:

The results obtained by applying AREA technique and Constructive Algorithm (and Refinement Algorithm after that) are outlined in Table 3.

Transfer probabilities	**Solutions obtained by AREA**	**Solutions obtained by the CA + RA**
$p_{0,11}$	0,701	0,7
$p_{0,21}$	0,298	0,3
$p_{11,12}$	0,404	0,4
$p_{11,22}$	0,595	0,6
$p_{21,32}$	0,901	0,9
$p_{21,42}$	0,098	0,1
$p_{12,13}$	0,513	0,55
$p_{12,23}$	0,486	0,45
$p_{22,33}$	0,755	0,75
$p_{22,43}$	0,244	0,25
$p_{32,53}$	0,970	0,95
$p_{32,63}$	0,029	0,05
$p_{42,73}$	0,978	0,85
$p_{42,83}$	0,0213	0,15
Δ_1	106,251	106,372

TABLE 3. Solutions obtained by AREA technique and CA for the first set of constants considered.

Remark. Final result obtained by AREA technique and CA is very similar. Only CPU time is different: CPU time obtained by AREA technique is 0.05s, and the CPU time obtained by CA is 11 minutes.

4.2. **Experiment 2.** Consider the experimental conditions given in Table 4. The values of constants $c_1,\ldots,c_{14}$ obtained from this conditions are the followings:

$c_1 =$ 16,66,
$c_2 =$ 33,33,
$c_3 =$ 16,66,
$c_4 =$ 133,33,
$c_5 =$ 33,33,
$c_6 =$ 16,66,
$c_7 =$ 173,33,
$c_8 =$ 173,33,
$c_9 =$ 16,66,
$c_{10} =$ 16,66,
$c_{11} =$ 213,33,

$c_{12} = 213{,}33$,
$c_{13} = 213{,}33$,
$c_{14} = 213{,}33$.

The parameters used by AREA in this case are presented in Table 5:

The results obtained by applying AREA algorithm and CA + RA are presented in Table 6.

Remark. According to Table 6 the final solutions obtained by these two algorithms are very close. CPU time obtained by AREA technique is 0.05 s, less than the CPU time obtained by CA, which is 15 minutes. Evolutionary algorithms seem to be useful technique for practical optimization proposes.

5. Stochastic model versus heuristic strategy

In this section we compare our stochastic optimization model and a very popular transfer heuristic [see Özsu, P. Valduriez, 1999]. According to the transfer heuristic the smaller relation from the operands of a join is transfered to the other operand relation. A query against a database is executed several times (not only once). The proposed stochastic model takes it into consideration and tries to share the execution of the same query between the sites of the network.

We say a strategy is "*pure*" if the execution path of the query in the state-transition graph is the same in every case the query is executed. If the query is executed several times, one of the joins of the query is executed in every case by the same site, and this is valid for every join of the query.

In the following we compare the results of the stochastic model with the results given by a "*pure*" strategy.

In step 1 of Experiment 1 the transmission of relation B is chosen several times, because it's size is smaller than the size of relation A. Therefore the system undergoes transition from state s_0 in state s_{11} in 7 cases from 10.

	Number of bits	Time to transfer
Relation A	8,000,000	133.33s
Relation B	1,000,000	16.66s
Relation C	2,000,000	33.33s
Relation D	1,000,000	16.66s
Relation B'	10,400,000	173.33s
Relation C'	12,800,000	213.33s

Table 4. Relation sizes and transfer times for Experiment 2.

AREA Parameters	
Population size	100
Number of iterations	10000
Mutation probability	0,01
Number of variables	14
Number of alphabets	30
MAX_HARMFUL_MUTATIONS	5

TABLE 5. Parameters used by AREA algorithm.

Transfer probabilities	**Solutions obtained by AREA**	**Solutions obtained by the CA + RA**
$p_{0,11}$	0,778	0,75
$p_{0,21}$	0,221	0,25
$p_{11,12}$	0,75	0,75
$p_{11,22}$	0,249	0,25
$p_{21,32}$	0,962	0,875
$p_{21,42}$	0,037	0,125
$p_{12,13}$	0,75	1
$p_{12,23}$	0,25	0
$p_{22,33}$	0,996	0,875
$p_{22,43}$	0,003	0,125
$p_{32,53}$	0,891	0,25
$p_{32,63}$	0,108	0,75
$p_{42,73}$	0,771	0,875
$p_{42,83}$	0,228	0,125
Δ_1	39,7492	40,3232

TABLE 6. Solutions obtained by AREA technique and CA for the second set of constants considered.

>From state s_{11} states s_{12} and s_{22} are chosen in a balanced mode. This because the size of relation B' is approximately equal to the size of relation C.

This balance is not so evident from state s_{21}. This can be explained by the approximative character of our methods.

In the third step of query strategy, which is the join of relation D with the result relation C', nearly in every case relation D is chosen for transfer. This because the size of D is much smaller than the size of relation C'.

Consider the "*pure*" strategy (deduced from transfer heuristic): s_0, s_{11}, s_{12}, s_{13}. Every join is executed in site 1, and the necessary time is: 66,66s + 166,66s + 83,33s = 316,65s, which is much greater than the mean processing time given by the stochastic query optimization model (This time is 106,251s).

In Experiment 2 the situation is similar. As the size of relation B is smaller than in case of Experiment 1 the transfer of it is chosen more often than in case of Experiment 1.

In step 3 in most cases the transfer of relation D is chosen. The size of D is much smaller than the size of the result relation C'. A "*pure*" strategy in this case with the same heuristic may be s_0, s_{11}, s_{12}, s_{13}. The necessary time for this "pure" strategy in site 1 is: 16,66s +33,33s + +16,66s = 66,65s, which is greater than the mean processing time given by the stochastic optimization model. (This time is 39,7492s)

6. Conclusions

In this paper the stochastic model is extended to the join of four relations. These four relations are stored in four different sites. The stochastic query optimization problem in case of four relations leads to a constrained nonlinear programming problem. For solving this problem two different approaches are considered: a constructive (exhaustive) one and an evolutionary one. The results obtained by applying these two methods are very similar. The difference consist in CPU time: by considering evolutionary method for solving the problem the execution time is less than the running time obtained by applying the constructive method.

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.

Department Of Computer Science, Faculty of Mathematics and Computer Science, Babes-Bolyai University, Kogălniceanu 1, 400084 Cluj-Napoca, Romania

E-mail address: {ddumitr, cgrosan,ivarga}@cs.ubbcluj.ro