Optimizing Access Strategies for a Distributed Database Design using Genetic Fragmentation

Rajinder Singh Virk\textsuperscript{1} Dr. Gurvinder Singh\textsuperscript{1}
\textsuperscript{1}As. Prof., Dept. Of Computer Science & Engineering, Guru Nanak Dev University Amritsar.

Summary
A key component of any Relational Distributed Database Query Optimizer is to fragment various tables and distribute fragmented Data over the sites of network. Then find an near optimal or best possible subquery operation allocation plan in a stipulated time period. In this paper we have proposed a Genetic Algorithm (GA) for finding near optimal fragmentation plan for selecting the various nodes or sites for placing recursively the vertically fragmented data attributes in two components for a Query Transaction on the Database. We discuss advantages of using proposed Genetic Algorithm (PGA) over various other prevalent Algorithms and un partitioned case. Experimental results for a simulated Distributed Database over a Wide Area Network show encouraging results for the use of PGA over other techniques.

Keywords
Distributed Database, Query Optimization, Database Fragmentation, Access Strategies, Genetic Algorithms.

1. Introduction
Two critical components of a Distributed Database Query Optimization Model are i) Determining the database fragmentation scheme and ii) Determining the subquery allocation plan. Most of research has contributed to second aspect i.e. on finding the operation allocation plan and order of execution of various subqueries of a bigger transaction query. In this paper emphasis is on exploring the former aspect i.e. fragmentation of various attributes vertically and recursively, using an innovative Genetic Algorithm(PGA). Queries can be categorised mainly in two groups first OLTP (Online Transaction Processing) and second as DSS(Decision Support System)[1].OLTP Queries are generally repetitive in nature. So Optimizer should seek good thru put and should emphasis on reducing the total cost of query. On the other hand DSS queries are not so touchy about high thru put but seek response time minimization . The proposed Genetic Algorithm ‘PGA’ works to minimize two important cost components wiz Local Processing Cost and Total Communication Cost for processing whole of query. It starts by generating an initial pool of solution by random generation of integer sequences which corresponds to partitioning a relation into two or more components. It corresponds to first Population. Further Populations are generated using standard principles of GA’s i.e. by applying selection, crossover & mutation[2]. Objective Function is based on Total Disk Accesses Minimization for different access path strategies. Selection takes care that better the Objective Function (Cost Minimization) fitter is a the chromosome and gets more number of probabilistic chances of reproducing offspring in next population. Crossover is used to let offspring share features of both parents and possibly improve over fitness value. Mutation is given a very meagre chance so that some important features of parent population are not lost by crossover. Reproduction is stopped when predefined maximum number of generations is reached or any further populations show no fitness improvement for a long number of generations.

2. Related Research Work
Distributed database systems design and query optimization has been and will remain an active area of research for a lot times to come, due to complex and intractable nature of the problem[10-17]. Most of the work has concentrated on two aspects: Fragmentation & Data Allocation(The plan of allocating Fragments to various sites) and Operation Allocation(How to generate a sequence of subqueries on various sites). Apers and P.M have discussed in detail the data allocation problem and their fragmentation in [18]. An integrated solution to problems of Data Fragmentation, allocation, replication in Distributed Databases, has been proposed in Tamhankar & Ram[19]. Simulation experiments for comparison of Branch & Bound, Simulated Annealing, Greedy approaches for operation allocation problem have been describes in detail by Martin & Lam[20]. Frieder and Baru [21] propose dynamic site selection strategies for distributed database design on a microcomputer. March & Rho in [10] have proposed an excellent cost model for reducing local I/O costs, CPU Costs and Communication Costs in operation allocation strategy. Johansson & Noumann in [22] extended their work by considering parallel processing and Load Balancing in Data and Operation Allocation.
Many different algorithms have been proposed for various aspects of Query Optimization, few e.g. fragmentation, data and operation allocation etc., may be divided into three major categories
1. Deterministic Search Algorithms
2. Randomized Algorithms
3. Genetic Algorithms

The most prevalent technique in first category is Dynamic Programming, as used in System R[3]. The Algorithm exhaustively searches through all plans for the query and principle of optimality prunes away bad sub-plans as early as possible[4]. Algorithm is suitable for a Database Design involving lesser no. of fragments and joins as it guarantees the optimal solution, but it is not time efficient for large no. of joins and relations as its exponential in nature[5]. Many variations of this classical version have come, most of which involve heuristic pruning [6,7].

In second category Randomized Algorithms make random choices as they walk thru the state space to find a local minima. Simulated Annealing (SA) was originally developed to model the annealing process of crystals. Difference from Iterative Improvement is that it can accept uphill moves with probability determined by a variable called Temperature as in Annealing process. The most successful of these Algorithms called 2PO (Two Phase Optimization) combines iterative improvement ( a variant of hill climbing) with Simulated Annealing[6]. This approach also becomes exponentially interactive soon with increase in number of relation, joins and sites[4].

The Algorithm presented in this paper (PGA) falls under the category of Genetic Algorithms. These algorithms simulate natural evolution process. These encode a potential solution to a specific problem on a simple chromosome (Encoded String of solution data) like structure. These strings are evaluated for Fitness (Objective Function) and relatively more fit members get more chance to reproduce their features to next population. Fitter chromosomes are propagated to next population until a predetermined no of population generations are produced or no further improvement in solution is taking place [2].

3. Genetic Fragmentation Model

3.1 Assumptions and experimental setup:

We consider retrieval transactions over a relation consisting of selections, projections and join operations. Relation attributes are classified as Clustering(Primary Key), Non-Clustering(Optional Key) and Non Key ones. Restrict attribute selects according to selection predicate, Scan attribute is the attribute which actually scans the relation.

It is assumed that following database transaction statistics are available apriori
(i) Transaction Frequency
(ii) Set of attributes needed for the transaction along with their selectivities
(iii) Set of restrict attributes and indexes.

Different access paths are evaluated for different disk accesses required for the transaction and multiplied by its frequency.

Initially a Relation is partitioned into two fragments labelled as Primary Fragment and Secondary Fragment. The total no of disk accesses are equal to the sum of disk accesses required for the Primary and Secondary fragment components. We use formulation of A.F.Cardenas[22], S.B.Yao[23] for estimating the disk accesses required a shown in following equations.

3.2 Estimated Disk Accesses for Primary Fragment:

   a) For Clustered Index Scan:
   
   Disk Access Cost is given by
   
   \[ D_{ci} = \frac{S_{sa}C_{r}T_{p}}{B} \]  \hspace{1cm} (3.1)
   
   where
   
   \[ S_{sa} \] : Selectivity of scan attribute
   \[ C_{r} \] : Cardinality of the relation
   \[ T_{p} \] : Tuple Size of Primary Fragment
   \[ B \] : Block Size (4k for a Page)

   b) For unclustered index
   
   Number of Disk accesses are estimates as
   
   \[ D_{ui} = P_{pf}(1-(1-1/P_{pf})^{1} + S_{sa}N_{op}) \]  \hspace{1cm} (3.2)
   
   \[ P_{pf} = C_{r}T_{p}/B \]
   \[ T = S_{sa}C_{r} \]
   \[ P_{pf} = \text{Primary Fragment Page Size} \]
   \[ T = \text{No. Of tuples after applying selection Predicate} \]
   \[ S_{sa} \] : Selectivity of scan attribute
   \[ C_{r} \] : Cardinality of the relation
   \[ T_{p} \] : Tuple Size of Primary Fragment
   \[ B \] : Block Size (4k for a Page)

   c) For Sequential Scan
   
   \[ D_{ss} = \frac{C_{r}T_{p}^{2}}{B*F_{b}} \]  \hspace{1cm} (3.3)
   
   \[ F_{b} = \text{Prefetch Blocking Factor} \]
   
   For different access strategies we look for minimum cost out of \{ \[ D_{ci}, D_{ui}, D_{ss} \].

3.3 Access Costs for secondary Fragment

   a) Sequential Scan
   
   Using tuple identifiers Estimated disk accesses are given by
   
   \[ D_{2ss} = P_{pf}(1-(1-1/P_{pf})^{T}) \]  \hspace{1cm} (3.4)
\[ P_{pf} = C_T T^s / B \]
\[ T = P_S C_T \]
Where \( P_S \) is selectivity from primary segment due to multiple restrict attributes and is given by
\[ P_S = S_{sa} \prod_{i=2}^{k} S_i \]
\( S_i \) is the selectivity of attribute \( i \) of primary fragment or 1 otherwise.
\( T^s \) is tuple size of secondary fragment

Using sequential scan
\[ D_{2is} = C_T T^p / B^* F_b (3.X5) \]

Disk Access Costs are taken from minimum of \( \{D_{2is}, D_{2ss}\} \), same is true for unclustered index scan.

b) Clustered Index Scan

Estimated disk accesses are given by
\[ D_{2ci} = S_{sa} C_T T^p / B (3.X7) \]


Input: Transaction Profile
output: Different Fragmentation schemes along with their costs

1. Initialize gene with randomly generated binary strings to initialize \( N \) initial Parent Chromosomes.
Repeat steps 2-6 until \( \text{No of Gen's} = M \) (Max no of gens) or previous five generations have not improved any further on Avgfit.
2. Read Input Data File containing various transactions and their frequencies, attributes required and their selectivities.
3. Read clustering attribute, restrict attributes and indexes available.
4. Calculate fitness of each chromosome by calculating the Objective Function by estimating disk accesses for each case of access strategy, as given above by set of equations 3.X and choosing one access strategy which minimizes the disk access costs.

3. MaxFit = Maximum Fitness of a chromosome from present generation, Avgfit=Average fitness of the generation.
4. Elitism: Pass the MaxFit Chromosome to next generation.
5. Generate the mating pool with various chromosomes entering it with copies proportional to the fitness. Exchange parts of chromosomes with Crossover Operator with probability=7/10 and apply Mutation Operator with probability=2/100.
6. Generate the remaining \( N-1 \) chromosomes of the new population. Generate the population Report.
7. Print Final Fragmentation scheme table.

5. Experimental Results:

Algorithm was coded into a Genetic Program in PASCAL Language by simulating a Distributed Database Environment. Experiments were conducted on a PC Pentium Core 2 CPU 2.13GHz with 1 GB RAM on WINDOWS-XP Platform.

Evaluation of the GA is done by taking the classic example problem of Cornell & Yu[25]. A transaction profile for this is shown below

Comparing results with that of Cornell and Yu[25] and Rho & March[14] and unpartitioned one. we find PGA showing 30% improvement over unpartitioned technique. It also outscores other two techniques by 8% an 19% of the disk access costs respectively.

<table>
<thead>
<tr>
<th>Transaction Profile</th>
<th>DISK ACCESS COSTS</th>
<th>SOLUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGA</td>
<td>70</td>
<td>{1 2 5 7 8} {3 4 6 9 10}</td>
</tr>
<tr>
<td>Rho &amp; March</td>
<td>78</td>
<td>{1 2 3 5 7} {3 4 6 8 9 10}</td>
</tr>
<tr>
<td>Cornell &amp; Yu</td>
<td>89</td>
<td>{4,6,10} {1 2 3 5 7 8 9}</td>
</tr>
<tr>
<td>Unpartitioned</td>
<td>100</td>
<td>{1 2 3 4 5 6 7 8 9 10}</td>
</tr>
</tbody>
</table>

### Experimental Results Table

- **PGA**: 70 {1 2 5 7 8} {3 4 6 9 10}
- **Rho & March**: 78 {1 2 3 5 7} {3 4 6 8 9 10}
- **Cornell & Yu**: 89 {4,6,10} {1 2 3 5 7 8 9}
- **Unpartitioned**: 100 {1 2 3 4 5 6 7 8 9 10}
6. Conclusion:
The goal of this research paper is to present a probabilistic Genetic Solution to the fragmentation and access strategies problem of a Distributed Database Design. Significant improvements over other prevalent methods have been highlighted as shown by Results table and graphs above.

References:
[8] www.tpc.org