A Stochastic Simulation of Optimized Access Strategies for a Distributed Database Design

Rajinder Singh, Gurvinder Singh, Varinder Pannu virk

Abstract—This paper highlights a design of a probabilistic solution to the operation allocation problem of Distributed Databases. Most of the present day commercial vendors of Distributed DBMS use deterministic procedures along with certain heuristics on exhaustive enumeration procedures like Dynamic Programming, Greedy Techniques, Randomized strategies etc. These procedures have a lot of scope for improvements when problem domain is increased from the point of view of 'number of sites' or 'number of joins' involved in a distributed query. Recently great interest has been shown by researchers to apply Genetic Algorithms to achieve this. This paper highlights design and implementation of one such model, Genetic Algorithm for Subquery Allocation (GA_SA), which is a modest effort to stochastically simulate optimization of retrieval transactions for a distributed query.

Index Terms— Stochastic Simulation, Genetic Algorithms, Distributed Database, Access Strategies, Query Optimization, Fragments, sub-query operation, Computer Network, Random Number Generation, Query Tree.

1 Introduction

ONE of the important key components of a distributed database query processing & optimization process is the allocation of the sites of a network to various operations or sub-queries involved in a particular Query. A distributed query is first broken into various sub operations like Selections/Projection or Joins/Semi-joins, then these operations may be performed at many different sites of the network in many different sequences. These two components of the distributed query optimization process are popularly referred as Operation Order Sequence Problem (OSP) & Operation allocation Problem (OAP). OSP involves finding the optimal ordering of operations e.g. Join order sequence.OAP involves finding the optimal placement of these operations to different permutations of network sites. Both of them are proven to be NP Complete and NP Hard problems while trying to find an optimal solution for a combination of large number of sites and operations [1]. For this reason one prefers a stochastic solution over deterministic ones. Because enumerative and deterministic procedures go intractable quite quickly as soon as the number of sites or number of complex operations like joins are increased. So this has been an active area of research for database community since last few decades and the hunt for better techniques or heuristics is still actively pursued [2].

In this paper focus is on sub-query operation allocation problem. An innovative Genetic Algorithm (GA_SA) is proposed as a solution and a stochastic simulator is designed based on this algorithm. Paper is organized as follows. Section 2 discusses various prevalent techniques and earlier research in the area. Section 3-5 describe step by step, all the details of the genetic algorithm and simulator's design like Query representation by Query Tree, mathematical formulation of the objective function, Data File Design, Flowchart of GA_SA, Genetic Algorithm and Graphical Analysis. Finally in section 6 we analyze and validate the solution by taking a set of Queries on a Winconsin Benchmark Database and comparing the GA_SA costs and execution times with other widely followed methods like Exhaustive Enumeration, Dynamic Programming, Branch & Bound and Simulated Annealing etc. Due to the NP-Hard nature of the problem, there are no standard benchmarks for comparing efficiency of distributed database design and query optimization algorithm variants. So, to validate the results of the proposed genetic solution GA_SA, some of above stated approaches have been compared by using benchmark analysis & results from the classic works of Martin & Lam [4], other good comparisons are available in [5], [6], and [7].

- *Rajinder Singh, Gurvinder Singh are working as Associate Professor at Guru Nanak Dev University, Amritsar, India. E-mail: tovirk@yahoo.com.*
- *Varinder Pannu Virk is a Sr. Lecturer at Govt. Polytechnic Amritsar,India.E-mail: viki_virk@yahoo.com*

2 Previous Research On Site Allocation Techhniques :

2.1 Introduction:

Research activities in this field can be broadly divided into following three categories, which have been extensively elaborated and compared in earlier works of [8],[9],[10],and [11].

a) Deterministic techniques: Exhaustive Enumeration with heuristics (Dynamic Programming), Branch & Bound, Greedy, Iterative Dynamic Programming.

b) Randomized Techniques: Iterative Improvement, Simulated Annealing, 2PO (Two phase Optimization).

c) Evolutionary Techniques: Genetic Algorithms, Multi objective Genetic Programming.

2.2 Exhaustive Enumeration

It is the most primitive technique, when used along some heuristics to prune suboptimal plans is then called Dynamic Programming Technique. This was the first approach in System R project of IBM and was later used for query optimization by most of DBMS vendors. Working in a bottom-up fashion it builds more complex sub-plans from earlier simple sub-plans until the completion of Plan. Access plans are build for every relation in the first phase and then algorithm enumerates all two way join plans using access plans as building blocks in the second phase. Third phase (*finalizePlans*) finalizes the plans by attaching the operations to make a complete plan. The function *prunePlans* helps in discarding suboptimal plans as early as possible. A brief outline of the general procedure is given below in figure 2.1.Distributed version does table scans at different sites and plan pruning is postponed till covering all sites.

```
Input: SPJ query q on Relations $R_1, R_2, \ldots, R_n$.
Output: A query plan for q
for i=1 to n do
        optPlan ({$R_I$}) = accessPlans($R_i$)
        prunePlan(optPlan ({$R_I$}))
endfor
for i=2 to n do
        for all S $\in$ { $R_1, R_2, \ldots, R_n$ } and $|S|$= I do
                optPlan(S) = $\emptyset$
        for all O $\in$ $S$ do
                optPlan(S) = opPlan(S) $\cup$ joinPlans
                (optPlan(O), optPlan(S-O))
                prunePlans (optPlan(S))
        endfor
        endfor
endfor
finalizePlans(optPlan( { $R_1, R_2, \ldots, R_n$ })
prunePlans(optPlan( { $R_1, R_2, \ldots, R_n$ })
return(optPlan( { $R_1, R_2, \ldots, R_n$ })
```

Figure 2.1DynamicProgramming Algorithm

Dynamic programming outperforms other techniques for a small number of relations and fragments on different sites,up to 2-5 relations having 7-12 fragments on 3-5 sites. Beyond this when we suddenly increase number of fragments and sites, its computing time rises exponentially and soon goes practically intractable.

2.3 Branch & Bound

It represents allocation program by considering a search tree. Sub queries are represented by nodes and arcs give allocation of sites to these subqueries. Branch & Bound operates in depth first search of search tree and terminates a search path if it finds objective cost function exceeding the current minimum cost for a complete function [13].This algorithm is highlighted in Figure.2.2.

```
Generate initial random allocation $A_0$
$U \leftarrow$ Cost($A_0$)
$C \leftarrow 0$
        $A \leftarrow \emptyset$ and $M \leftarrow \emptyset$
        Randomly choose first $i \in N$
        Call Branch&Bound (A, C, U, i, M).
        Branch&Bound (A, C, U, i, M)
        for each j $\in S$ do
        if CapUsed + CapRequired(i) $\leq$ MaxCap(j) and
        J $\in$ SiteSet(i) then do
                $C' \leftarrow C + \Delta c(i,j)$.
        If $C' \leq U$ then do
                $C' \leftarrow C$
                $A \leftarrow A + Allocate(i,j)$.
                 If all nodes allocated then do
                        $U \leftarrow C$
                        $M \leftarrow A$
                return.
                end.
                else do
                Randomly choose new $i' \in N$,
                Call Branch&Bound (A, C, U, i', M).
                end.
                end.
          end.
        end.
        end Branch&Bound.
```

Figure 2.2: Branch&Bound Algorithm

The variable A_0 corresponds to an initial allocation of the subqueries uses to denote initial upper bound for the cost given by U.C & C' denote costs, A denotes current partial allocation and M denotes current minimum cost allocation, both are empty initially.i&i' denote nodes and j denotes a site whereas *SiteSet(i)* .The function CapRequired(i) returns the amount of capacity required by a site to be able to solve that query. The function $\Delta c(i,j)$ returns cange of cost effected by alternating site i and j. The function *Allocate(i,j)* does allocation of site j for subquery i . Cost associated with an allocation plan A is given by *Cost(A)* [4].

B&B is a recursive procedure which allocates various different sites to all possible allocation enumerations, while keeping watch over restriction that cost does not increase over the current minimum. Then all possible enumerations from that point onwards are discarded. Its search space is shortened very effectively by this bounding function. It takes quite less time to find the minimal cost allocation as compared to Exhaustive Enumeration

2.3 Simulated Annealing

Simulated Annealing is a process analogous to the annealing of crystals, in which liquids are cooled very slowly at temperatures nearing freezing point. Physical Annealing searches the space of all possible atomic arrangements for the state with a minimum possible energy. In general Optimization each state is replaced by a possible solution to the problem and energy is replaced by the cost of a suggested solution [4].

It is refinement of an earlier randomized algorithm Iterative Improvement which suffered from premature findings of local cost minima, instead of finding a global minimum. This is achieved by allowing both downhill and uphill state changes, i.e. it allows the cost function to decrease as well as increase. The probability is set to $e^{-\left(\frac{\Delta c}{t}\right)}$, where Δc is the difference in cost in the source and destination. It is highlighted in Figure.2.3.

```
        Calculate        $T_0$ and $T\infty$
        Generate a Random Initial Allocation $A_0$
```

```
Calculate        C ← $Cost\ (A_0)$
$T \leftarrow T\infty$      and      $A \leftarrow A_0$
while $T > T_0$
while (P < $k_1$ x |N|)      and (R < $k_2$ x |N|)
do
        Generate a new Allocation $A'$
        Calculate
        $c \leftarrow Cost(A') - Cost(A)$
        if
$(\Delta c \le 0)$ or $random(0,1) < e^{-(\frac{\Delta c}{t})}$
        then    $A \leftarrow A', P \leftarrow P' + 1, C \leftarrow C' + 1$
        else $R \leftarrow R + 1$
        end.
        P ← 0, R ← 0, T ← k3 x T
end.
```

Figure 2.3: Simulated Annealing Algorithm

3 Genetic Algorithm Design

3.1 Objective function formulation Decision Variables and cost model

Simulation starts with designing of a distributed database environment by assuming a set 'S' of data distribution sites, a set 'R' of relations/fragments stored on those sites and a Set 'Q' representing a set of transactions.

We have assumed for simplicity of design that only retrieval queries are there. The model can be easily expanded to incorporate update queries.

Let a query transaction 'q' for retrieval, be broken into a set of 'j' sub queries on the 'R' set of relations.

(i) Data Allocation Variable Ars :

A_{rs} = 1 (if site 's' holds copy of fragment 'r').
A_{rs} = 0 (if 'r' copy is not available at site's').

(ii) Variables for site selection for sub query execution S_{qys} :
S_{qys} : (represents sequence of sub query execution at various sites in the life time of query).
S_{qys} = 1 (subquery 'y' of Query 'q' is done at site' s').
S_{qys} = 0 (otherwise).

(iii) For Join operations a notation is proposed to handle left previous operation of a join operation (LPO) & right previous operation of a join(RPO) as following:

$S_{yv[p]}S$ = 1 (for [p] = 1 for left previous operation of a Join).
$S_{yv[p]}S$ = 1 (for [p] = 2 for right previous operation of a Join
$S_{yv[p]}S$ = 0 otherwise.

(iv) I_{qry}: represents whether the sub query 'y' of query 'q' references the intermediate relation/fragment 'r'.
I_{qry} = 1 (if the base relation 'r' or intermediate fragment 'r' is used by sub query 'y' of 'q' query).
I_{qry} = 0 otherwise.

v) For use of intermediate Relations by Join Operation

$I_{qryv[p]}$ = 1 (for LPO of join 'y').
$I_{qryv[p]}$ = 1 (for RPO of join 'y').
Iqryv[p] = 0 otherwise.

By making use of above decision variables Operation Allocation Problem formulation is represented as :

Given (Ars & Iqry) :

A Transaction Profile, a Data Allocation Scheme represented by variable **Ars,** and given **Iqry** intermediate relations/fragments is used by sub query y of query q .

To find (Sqys):

We have an objective Function to calculate as to **find Sqys** which minimizes the objective function, which is cost of query**.**

3.2 COST MODEL FORMULATION

Given a set of fragments	R = {r1,r2,…,r_n}
Given a network of sites	S = {s1,s2,…,s_m}
Given a set of sub queries	Q = {q1,q2,…,q_q}

Sub Query Allocation optimization problem involves finding the "minimum query cost" possible distribution of R to S.

Ceri[12] gives a model for Total cost as Total Cost Function having two components: query processing and storage cost as

TOC = ∑ QPCi + ∑vs€S∑ vfj€F STCjk

Where QPCi is query processing cost of application qi and STCjk is the cost of storing fragment Fj at site Sk.

We follow and modify this model of query cost as function of sum of local processing costs and transmission costs .We simplify it further by ignoring update costs and ignoring concurrency control costs as we are giving model for retrieval transactions(queries) only. Further concurrent retrievals don't impose any more integrity control costs. Ceri's formulation gives

QPCi = LPCi + TCi (LPC: Local Processing Cost)
(TC : Communication Cost)

3.3 Local Processing Costs

For Simple selection & projections
LPCqy = ∑s Sqys(I OCs ∑r IqryMqry + CPCs ∑r IqryMqry) (1)

Where Mqry = No. of memory blocks of relations 'r' accessed by sub query 'y' of q.
IOCs = Input Output Cost Coefficient of site s in msec per 8k bytes
CPCs = CPU Cost coefficient of site s.

So equation (1) represents local processing costs of transforming input relation from disk to memory and CPU time processing a selection or projection at sites 's'.

Ceri's model didn't take care of join costs separately in detail. For that we have extended his model to add join costs as following.

Local processing costs for a join
LPCqy = ∑sSqysIOCs∑p∑rpsIqryv[p]Mq ryv[p]

2(a)

+

$\sum sSqys(IOCs\prod rIqryMqry + CPCs\prod rIqryMqry)$ 2(b)

Where 'p_s' is Selectivity Factor & is referred as the ratio of possible different values of a field to the domain of that field.(0<= ps <=1)
Mryv[p] is the size of intermediate relation
where v[p] represents p=1 for left previous operation of a join & p=2 for right previous operation of a join.
Equation 2(a) represents
Input Output costs in storing intermediate results of previous operations to the site of current join operation.
Equation 2(b) represents
CPU & I/O costs for performing join operations at site 's'.

3.4 Communication Costs:

These are involved mainly in case of join operations only as we have assured that selections & projections of retrievals are to be done only at sites which hold a copy of those base relations. Join may be performed at any of possible sites.

$COMMqy = \sum p \sum s \sum t\ Sq_y v[p]s\ *\ SqytCst\ (\ \sum_r\ I\ q_{ry} v[p]\ Mq_{ry} v[p]\ )$

Where C_{st} : is the communication cost coefficient bet -ween site t and s taken from input data matrix)

$C_{st} = 0$ if $(s = t)$ (i.e. previous operations and join operation on same site)

If the final operation is not done at the query destination site then a communication component is added separately for sending the final query result to that site.

4 DATABASE STATISTICS:

We consider a distributed database environment based on a Winconsin Database Benchmark Query (wq6). Structure of Tables and few example tuples are shown below:

Table: Bi (i varies from 1 to 7 for 7 Base Tables)
Cardinality: 10,000, Tuple Size: 10 bytes, Table Size: 100 kB Block Size: 1Kb , Table Size = 100 Blocks , Number_of_sites = 3
4.1 Relation Bi

Unique (0– 9999)	Twos (0 – 1)	Tens (0-10)	Hundreds (0 – 99)	FiveHunds (0 – 499)	Thousands (0 – 999)
7	0	1	11	4	999
111	1	4	2	3	4
9998	1	9	4	444	111
777	0	3	98	499	45
10,000 Tuples	....	......	....	......	......etc

Winconsin Database Benchmark Query (wq6).

4.2 Query: wq6:

$(\ \pi_{Unique}(\sigma_{2<Tens<9})B1):X:(\ \pi_{Unique}(\sigma_{2<Tens<9})B2):X:\ (\ \pi_{Unique}(\sigma_{2<Tens<9})B3)$
$:X:\ (\ \pi_{Unique}(\sigma_{2<Tens<9})B4)\ :X:(\ \pi_{Unique}(\sigma_{2<Tens<9})B5):X:(\ \pi_{Unique<Tens<9})B6)$
$:X:\ (\ \pi_{Unique}\ (\sigma_{2<Tens<9})B7\ ).$

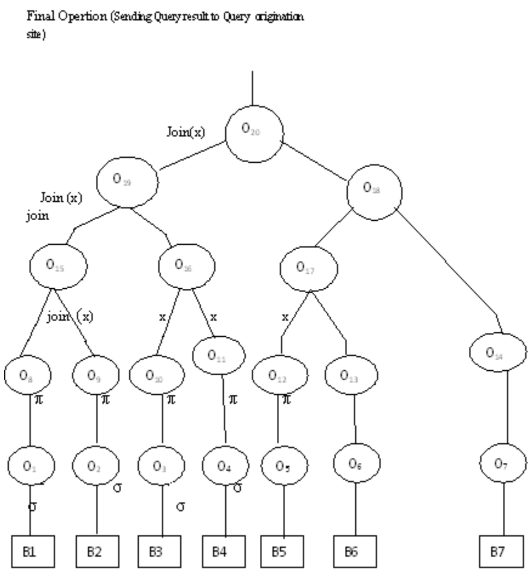

Fig: 4.1:Query Tree for WQ6 Query

4.3 The Query Tree Description:

No. Of Selections:	7 (O1 – O7)
No. Of Projections:	7 (O8-O14)
No of Joins:	6 (O15 – O20)
No. Of Operations:	10 (O1 – O21) (Tree Nodes)
No. Of Fragments:	27 (Tree Edges)

4.4. Designing the Input Data File: Wq6.dat

Here we explain some components of the input data file design to simulate a part of the problem domain.One may start by a text file line like

5 27 21	which represents	
Line1:	Query Site is site number :	5
	No of Base Relations :	27
	No. Of Operations :	21

1 1 1
Line2: I/O Coefficients of various sites
1 1 1
Line3: CPU Coefficients

Lines 4-6:Communication speeds between various
0 1 1 pairs of sites

```
1 0 1
1 1 0

1 1 0      Lines 7-13: Base Relation Placements on sites
0 0 1      Variable:  Ars :   Data Allocation Matrix
1 1 0 ──▶ e.g. ⎧  This line represents that Relation
0 0 1          ⎨  B3  is available at sites S1 & S2
1 1 0          ⎩
0 1 1
1 0 0
```

100 100 100 100 100 100 100 70 70 70 70 70 70 70 20 20 20 20 20 20 20 40 40 40 40 40 100
(Base & Intermediate Fragment's estimated Sizes in Kb(blocks))

Next 27 lines each representing whether a fragment f (1-27) is required by an operatio o(o1- o21):

```
   ------------ Operations --------------->
F   1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
r   0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
a   0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
g   0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
m   0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
e   0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
n   0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
t   0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
s   0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
-   0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
b   0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
e   0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
i   0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
n   0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
g   0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
-   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
r   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
e   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
f   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
f   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
e   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
r   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
e   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
d   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

5. Flowchart for GA_SA Design

Next shown is a flowchart for GA_SA working
In fig.5.1.

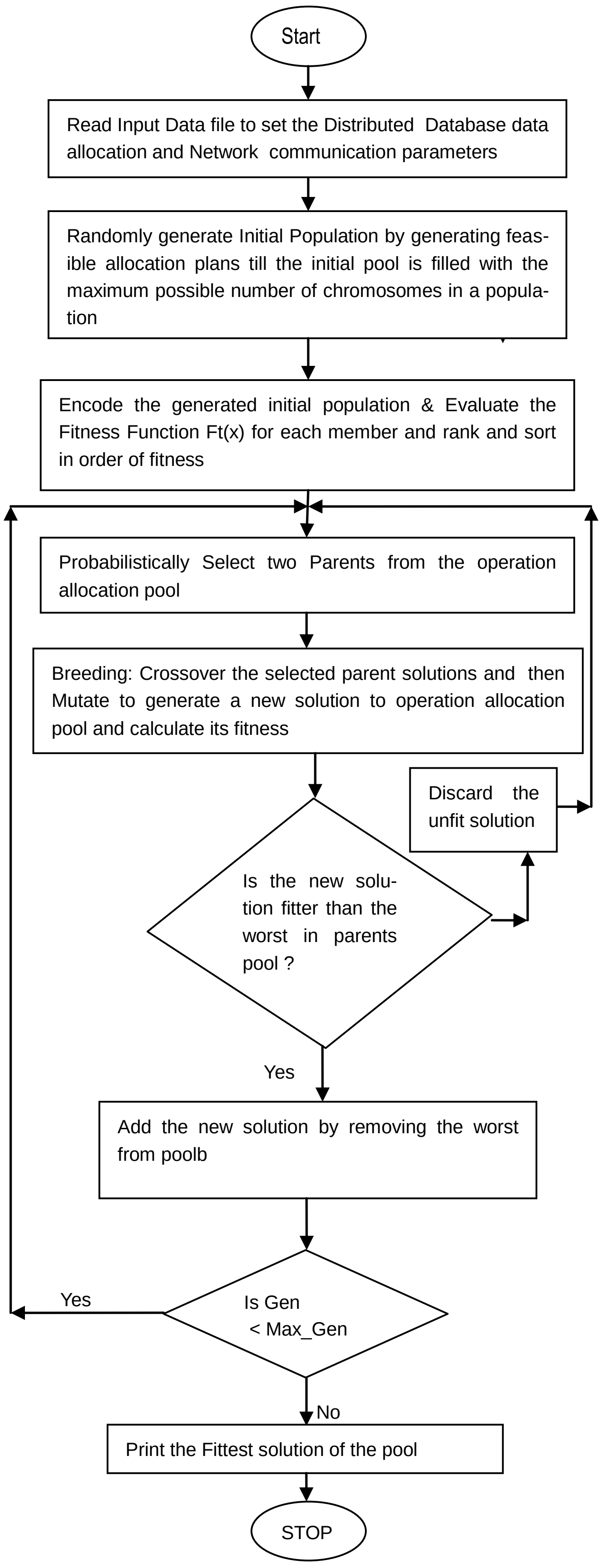
Start
Read Input Data file to set the Distributed Database data allocation and Network communication parameters
Randomly generate Initial Population by generating feasible allocation plans till the initial pool is filled with the maximum possible number of chromosomes in a population
Encode the generated initial population & Evaluate the Fitness Function Ft(x) for each member and rank and sort in order of fitness
Probabilistically Select two Parents from the operation allocation pool
Breeding: Crossover the selected parent solutions and then Mutate to generate a new solution to operation allocation pool and calculate its fitness
Discard the unfit solution
Is the new solution fitter than the worst in parents pool ?
Yes
Add the new solution by removing the worst from poolb
Yes
Is Gen < Max_Gen
No
Print the Fittest solution of the pool
STOP

Fig.5.1

6. Genetic Algorithm Design:

Pseudo code of the proposed Genetic Algorithm (GA_OA) is as following:-

Step 1:
(a) Generate 'n' random numbers between integers '1' and 's' where 's' is a integer denoting the maximum no. of sites and 'n' is the number of operations to be performed. Generating one random possible plan.
(b) Repeat step (a) for the total strength of the initial population.
(c) Evaluate cost of each operation allocation plan using objective functions.
Step 2:
(a) Iterate through number of generations to be generated.
(b) Probabilistically select using roulette wheel method without replacement , the parents , with higher than average fitness getting more that one offspring in next generation.
(c) Apply crossover and mutation to produce new child execution plan
(d) The maximum fit member of the previous generation replace the worst member of new generation.
(e) Evaluate the various fitness values for the entire generation.
(f) Repeat steps (2a) to (2e) for maximum number of generations.

6. Experimental Results:

We run Genetic Algorithm ***GA_OA*** based on optimizing the earlier described objective function.

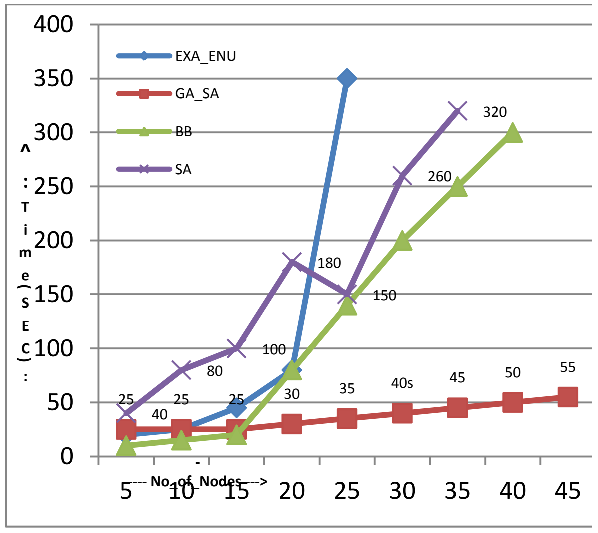

Fig. 6.1. (Simulator ouput for Query: wq6)

It highlights the fact that genetic algorithm takes much less time to find optimal solution for operation allocation of a distributed query as compared to deterministic exhaustive procedures or Branch&Bound, or Simulated Annealing Techniques.

A deterministic procedure ***EXA_ENU*** is also coded which enumerates all possible combinations for 21 operation of query ***WQ6*** to be done at various sites S_i . We plot a graph between varying numbers of sites on x axis to time taken by computer to find optimal solution in seconds as y axis. It is also compared with Branch & Bound and Simulated Annealing Algorithm run times as given in [4], ans is shown in fig 6.1 above.

7 CONCLUSION

In this paper an attempt has been made to highlight the main components of a stochastic simulation of a distributed query allocation, using Genetic Algorithms. Design of query representation and input data file to the simulator is explained. Run Time comparison of GA_SA with other popular Operation Allocation techniques is done as highlighted in a graph at Fig.6.1. GA_SA is the best choice possible when 'number of sites' or complex operations like 'joins' increase. Dynamic Programs or Randomized techniques like Simulated Annealing take far more time to find a good solution than a Genetic Solution like GA_SA.

REFERENCES

[1] Garey, M., D.Johnson Computers and Intractability: A Guide to the Theory of NP completeness, W.H.Freeman, 1979.
[2] M.Tamer,Ozsu,Patrick Valduriez: Principles of Distributed Database Systems, Dorling Kindersley, 2006.
[3] Douglas W, Cornell and Philip S Yu," On Optimal Site Assignment or Relations in the Distributed Database Environment", *IEEE Transactionson Software Engineering*, vol 15, no. -8, Aug-1989.
[4] Martin, Lam, Russel ," An Evaluation Of Site Selection Algorithm For Distributed Query Processing", The Computer Journal,vol33,1990.
[5] Ram NarsimhanNetwork Outputs, with Relationships to Statistical Pattern Recognition," *Neurocomputing – Algorithms, Architectures and Applications*, F. Fogelman-Soulie and J. Herault, eds., NATO ASI Series F68, Berlin: Springer-Verlag, pp. 227-236, 1989. (Book style with paper title and editor)
[6] March. S .T, Rho. "Allocating Data and Operations to Nodes in distributed Database Design" IEEE Transactions on Knowledge and Data Engineering:pp. 305- 317,7April,1995.
[7] Cosar & Sevinc," An Evolutionary Genetic Algorithm for optimization of Distributed Database Queries", *The Computer Journal*, vol.54, no.5,pp.717-725,2011.
[8] Martin, Lam, Russel ," An Evaluation Of Site Selection Algorithm For Distributed Query Processing", The Computer Journal,vol33,1990.
[9] March. S .T, Rho. " Allocating Data and Operations to Nodes in distributed Database Design" IEEE Transactions on Knowledge and Data Engineering :pp. 305- 317,7April,1995.
[10] Kossmann D, "The State of Art in Distributed Query Optimization, " ACM Computing Surveys, Sep 2000.
[11] Amol V. Deshpande and Joseph M.Hellerstein," Decoupled Query Optimizationfor Fedrated Database Systems," *A Project Report, Universityof California* Berkeley,April 2001.
[12] Stefano Ceri,Giuseppe Pelagatti,"Allocation of Operations in Distributed Database Access," IEEE Transactionson on Computer, vol.C- 31, no. -2, Feb-19892
[13] P.R.Ma, E.Y.S Lee and M.Tsuchiya, "A task allocation model for distributed computing systems," IEEE Transactions on Computers C-31(1),41-47(1982) .