Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 6, June 2015, pg.544 – 551

RESEARCH ARTICLE

A Framework for Optimizing Distributed Database Queries Based on Stochastic Fractal Search

Manreet Sohal[1], Atinderpal Singh[2], Dr. Rajinder Singh Virk[3]
[1]*Department of Computer Science and Technology, GNDU, Amritsar, India*
[2]*Department of Computer Science and Technology, GNDU, Amritsar, India*
[3]*Associate Professor, Department of Computer Science and Technology, GNDU, Amritsar, India*
[1] reetsohal@yahoo.com, [2] saini_amrit@live.com, [3]tovirk@yahoo.com

Abstract- ***In this paper the problem of query optimization in distributed databases have been discussed. Query Optimization in large distributed databases is a NP-Hard natured problem and is quite difficult to solve. Communication costs for transferring data across various sited is the major cost that affects the performance of the query. Lots of research has been done on this area and number of algorithms like ant colony optimization, genetic algorithms, swarm optimization has been used to optimize the queries in distributed databases. In this paper we have provided new framework for distributed query optimization based on stochastic fractal search algorithm. In this approach two process have been used diffusion and update. In the first process new particles are created from existing particles and in the second process the position of particles are updated in order to reduce the overall communication costs of the queries.***

Keywords- query optimization, distributed databases, evolutionary methods, stochastic fractal search, communication costs.

I. INTRODUCTION

The aim of query processing in distributed database system is to translate the query from high level query (user query in non procedural languages like SQL) in distributed database to low level query (queries in procedural language like relational algebra) in which low level details are hidden from the user.[11] In a relational database, queries are solved by joining various tables(relations), but in Distributed databases, these tables are located on different dispersed sites of a computer network, therefore in order to solve these queries data has to be moved between the sites. Therefore distributed query costs consists of processing costs(i/o costs, cpu costs) and a transmission cost [3].As the size of database increases considerably the requirement for a distributed DBMS increases constantly. Communication costs required for the data transfer are the major costs of query processing in a DDBMS[2].

Query Optimization is a major aspect of query processing. Query optimization refers to process of finding the best execution plan for a given query which represents the execution strategy for the query. This QEP minimizes an objective cost function. The main objective of the query optimization is to decide the most efficient query execution plan which has minimum execution cost, among many possible plans by determining execution sequential order of relational operators. Query Optimization in distributed databases is very difficult task due to number of factors like data allocation, communication channel's speed, memory availability, database size , storage of intermediate result, pipelining and size of data transmission [11],[12]. In a distributed query processing first of all initial query is decomposed into fragment queries which operate on fragments rather than on global relation(data localization).Then in the second phase joins/semi joins are applied to reduce the size of data that has to be sent across the network to different sites. In final phase all the processed files are sent to assembly site for generation of final output. A Query Optimizer is used to generate Query Execution Plans (QEP) which represents an execution strategy of the query with minimum cost. [12],[5].The performance of distributed query depends upon the query optimizers ability to obtain efficient strategies of query processing. Its ability to find out issues of joins, their ordering, methods, reduction of query data size and cost reduction is difficult task. A good query execution strategy generated by query optimizer involves three phases. In first phase search space is found which is a set of alternative QEPs. In second phase cost model is built that compares costs of different QEPs. Third phase is developing a search strategy which finds the best execution plan using cost model.

Optimization is needed to minimize, time, cost, risk and to maximize profit, quality or efficiency. Several real life complex optimization problems have come forward in many scientific fields like engineering, business etc that cannot be solved in reasonable amount of time. There are several optimization algorithms and approaches to deal with various optimization problems.[6]

Exhaustive approach: This strategy selects the most efficient or best solution ,but the overall cost of optimization is high. This is due to the reason that the search space can increase considerably while solving high dimensional optimization problems.[13]

Deterministic approach: Heuristics based Exhaustive Enumeration(Dynamic Programming), Branch & Bound, Greedy, Iterative Dynamic Programming.[13]

Randomized Techniques: Iterative Improvement, Simulated Annealing, 2PO (Two phase Optimization). [13]

Evolutionary Techniques: Genetic Algorithms, Multi objective Genetic Programming, Stochastic Fractal Search, Particle Swarm Optimization, Ant colony Optimization. [13]

In this report we have mentioned stochastic fractal search algorithm for solving query optimization problems in distributed databases. Hence the proposed algorithm is named as SFS_DBQ. The algorithm proposed is based on fractal properties. It provides an insight into solving optimization problems based upon diffusion property turned up into fractals. This proposed algorithm can obtain solution that has the least error compared with the globally optimum solution within a least number of iterations, thus provides an improvement in terms of accuracy, convergence time and simplicity of operations. Two main processes occurred in the SFS algorithm are: The diffusing process and the updating process.

In this paper we proposed an algorithm for DDB query optimization based on Stochastic Fractal Search Algorithm. In Section 2 related work concerning query optimization is discussed. In section 3 various gaps have been described. In section 4 algorithms are proposed that solve the gap. Finally Section 5 concludes the paper.

II. RELATED WORK

M.Gregory [3] has developed a GA for optimizing queries and its performance is compared with other alternative random optimization techniques like random search, multistart etc. All the tables are fully reduced in a tree query by optimizing a semijoin using this GA. For this problem, evaluation of the fitness function is a costly task. The Genetic algorithm proposed for this uses a tree-structured data model with customized crossover and mutation operators that avoid the need for full re-evaluation of the fitness function for new solutions. To meet the real time nature of query optimization task the genetic algorithm proposed here uses a local search phase to provide the required real-time performance. The GA proposed in this paper is robust, performs well at the beginning of a search, overcomes the problem of premature convergence and makes persistent progress to better solutions.

P.Tiwari et al. [7] have analyzed Various Optimization Strategies. It has been proved that when Ant Colony Optimization is integrated with other optimization algorithms the results are more effective and viable. Research has revealed that the implementations of these probabilistic algorithms generate feasible solutions in distributed as well as relational database management system when the query size and the number of joins in the query becomes large.

Virk et al. [10] have analyzed Decision support system queries in this paper. The exhaustive enumerative technique and genetic algorithms are used to simulate the selected set of DSS queries under serial and parallel environments. Then the results of these simulations have been evaluated that shows that exhaustive technique leads to more optimized solutions but consume large amount of time for complex DSS queries and is thus infeasible to implement it. The genetic approach on other side is very fast but is less accurate as compared to exhaustive approach. In addition to that the paper shows that by executing different suboperations of a DSS query in parallel the total cost of system resources can be reduced.

Rajinder Virk et al. [13] have highlighted a design of a probabilistic solution to the allocation problem of distributed database. They have implemented a model named (GA_SA) Genetic Algorithm for Subquery allocation. It has been used to simulate optimization of retrieval transaction from a distributed query.

X.Li et al. [1] proposed query optimization algorithms on multi-relation semijoin to apply to the condition that takes buffer zone of distributed database system as the final assembly station of query's intermediate result. Query processing and optimization of distributed database are the major factors that affect the performance, service efficiency and reliability of system. The experiment have shown that query optimization algorithm based on multi-relation semijoin leads to the reduction of volume of

the data of intermediate result and effectively decreases the overall network communications costs. Comparing with common multi-relation semijoin algorithm, the optimization advantage of the algorithm in this paper is higher.
J.Wang et al [2] have presented genetic algorithms to minimize the data transmission costs required for a distributed set query processing. This approach is based on appropriate data structures and in this the constraints are incorporated into chromosomes. The performance is evaluated and various algorithms are compared. It has been proved by the experimental results that the genetic approach presented in this paper performed well both on the computational effort and the quality of solution.
Y.Kwok et al. [14] have build up a site independent fragment dependency graph representation to represent the dependencies among the fragments accessed by a query. This graph representation has been used to deal with data allocation problems in distributed database systems based on query-site and move-small query execution strategies. In this paper four data allocation heuristics, a genetic algorithm, a simulated evolution algorithm ,a mean field annealing algorithm and a random search algorithm has been designed, evaluated and compared on the basis of query response time and algorithm running time. The conclusion drawn is that no single algorithm is better than all others in both aspects. The SE and GA can produce better solutions than RS and MFA but solution quality is not of constant magnitude. RS has lowest time complexity and is fastest whereas SE is quite slow. When efficiency and solution are equally important ,then GA is the best choice.
Hongxing Li et al. [4] have presented a new coding method with tree structure of the genetic algorithm based on the position and the value, for query optimization of distributed database. Further, the genetic operators i.e reproduction, crossover and mutation have been improved in terms of this coding. In dynamic stochastic coding rule of the genetic algorithm some conventional optimization algorithms, as halflink query have been incorporated. The improved crossover requires a two step implementation and the improved mutation comprises of the value mutation and the position mutation. The query of distributed database is implemented using this proposed algorithm and the experimental results have proved that the improved genetic algorithm is very effective for optimization of queries in distributed databases.
H.Salimi [6] proposed a new algorithm based on random fractals to solve both constrained and unconstrained global optimization problems. The two main processes : diffusion process and update process have been employed by this approach. Some constrained and unconstrained benchmarks have been used to validate the performance of the this algorithm. It has been verified by experiments that the algorithm proposed in this report performs appreciably better than other previous well-known metaheuristic algorithms in terms of preventing getting caught in local minimums, and finding the global minimum.

III. GAPS

By conducting the review it has been found that the most of the existing literature in this field have certain flaws like:

- Fast convergence along with accuracy is not guaranteed.
- Lack in simplicity of operations.
- Get stuck in local minima

IV. METHODOLOGY

SFS_DBQ consists of two main processes: Diffusion process and Update process. In the first process, to assure exploitation property each particle diffuses around to current position. With this process the chances of achieving global minimum increases and it prevents being trapped in local minima. For SFS_DBQ static diffusion is considered i.e only the best particle generated from diffusion process is considered. Further SFS_DBQ uses random update processes that leads to exploration properties. Two generate new particles, a statistical method called Gaussian Walk is used that leads to global minima. A series of Gaussian walks taking place in diffusion process is listed in following equations:

$$GW_1=\text{Gaussian}(\mu_{pbest},\infty)+(\Theta \times P_{best}- \Theta'-P_i) \quad (1)$$
$$GW_2=\text{Gaussian}(\mu_p, \infty) \quad (2)$$
Where Θ and Θ' are uniformly distributed random numbers restricted to [0,1] and P_{best} is position of best point and P_i the ith point in the group or population.
$_{pbest}$, μ_p and ∞ are Gaussian parameters where $\mu_{pbest}=|P_{best}|$ and $\mu_p=|P_i|$

Standard deviation ∞ is given as

$$\infty= \left| \frac{\text{Log}(g)}{g} \quad (P_i-P_{best}) \right| \quad (3)$$

Suppose our optimization problem is D dimensional problem, therefore each individual particle is based on D dimensional Vector. During initialization process each particle is initialized randomly based on problem constraints. In our problem of distributed databases each individual particle is a vector whose length is equal to number of operations in the given query that is distributed among various sites. Initialization equations of kth point, P_k is addressed as follows:
$$P_k=LB + \Theta \times (UB - LB) \quad (4)$$
Where LB and UB are lower and upper problem trained vector

After initialization, the fitness function of each point is calculated to get the best point(P_{best}). In accordance with exploitation property in the diffusion process ,all the points roam around their current position to utilize problem search space. Two statistical exploratory update procedures are used that increase better space exploration. The first update process is applied to each individual vector index and second update then acts upon on all points.

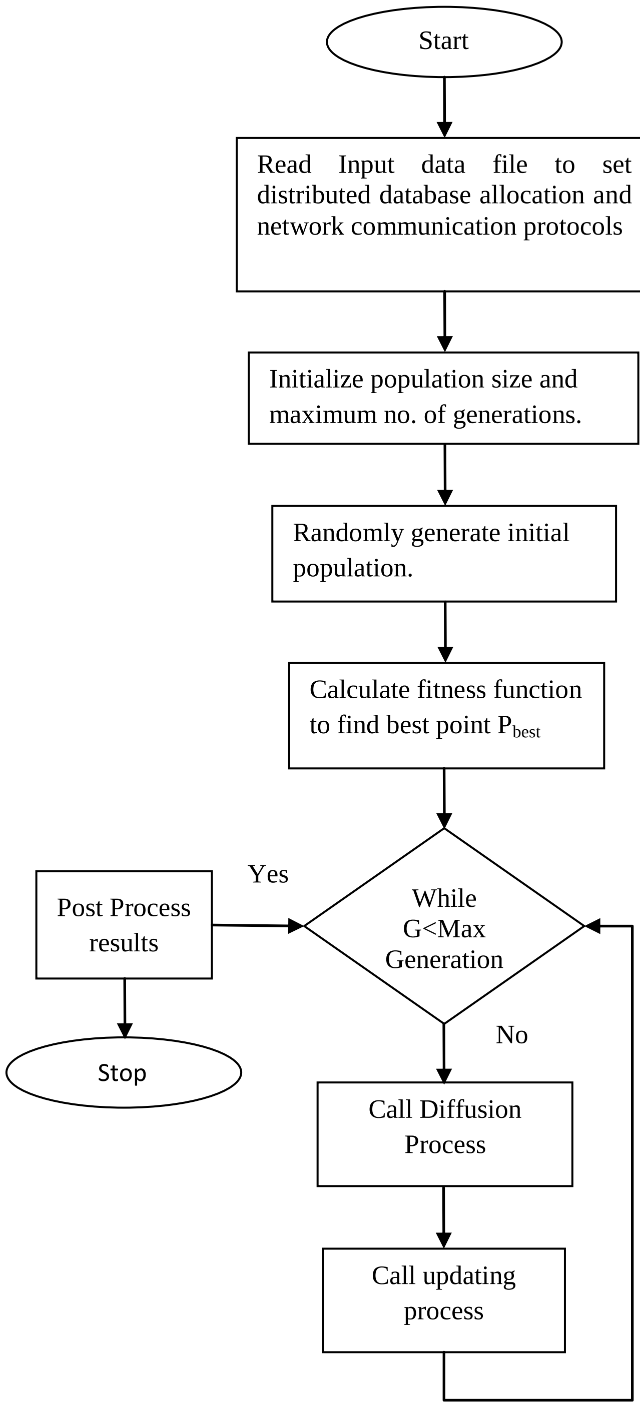
Start
Read Input data file to set distributed database allocation and network communication protocols
Initialize population size and maximum no. of generations.
Randomly generate initial population.
Calculate fitness function to find best point P_{best}
While G<Max Generation
Yes
No
Post Process results
Stop
Call Diffusion Process
Call updating process

Fig. 1 Flowchart for SFS_DBQ

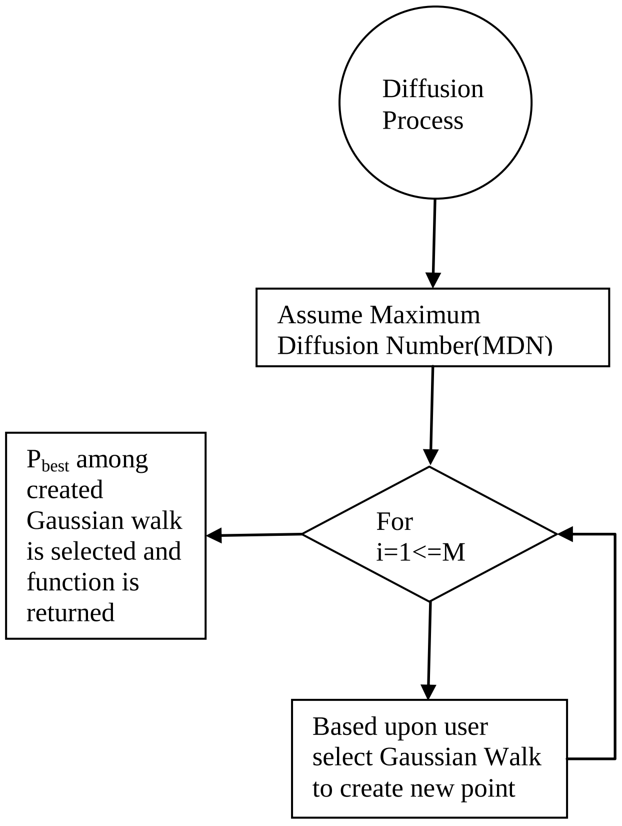
Diffusion Process
Assume Maximum Diffusion Number(MDN)
P_{best} among created Gaussian walk is selected and function is returned
For i=1<=M
Based upon user select Gaussian Walk to create new point

Fig. 2 Flowchart for Diffusion Process

In first process, first of all the points are ranked based on the fitness value evaluation which is total(i/o , processing and communication costs) cost of processing query in case of our problem.

The objective function is to minimize the sum of local processing costs and communication costs given by following formula:
Min(local processing costs for selection and projections+ i/o costs for storing intermediate results of previous operations current join operation's site+ cpu and i/o costs for performing current join operations on a particular site+ communication costs)

Each point is then assigned a probability given by formula :

$$Pb_i = \frac{\text{Rank}(P)}{N} \qquad (5)$$

With this equation the chance of changing the position of points which have not yielded good solution is increased. On the other hand chances of passing good solution in next generation will increase. For each point P_i in the population, if $Pb_i < \Theta$, then the kth component of P_i is updated according to equation (6) , otherwise it remains unchanged.

$$P_i'(k)=P_s(k) - \Theta \times (P_t(k) - P_i(k)) \qquad (6)$$

P_i' is the new modified position of $P_{i.}$ P_s and P_t are random selected points in the population.

After the first update procedure, which is carried out on the components of the points, a second update function is applied in which the position of the point is changed considering position of other points in the group. With this property quality of exploration is improved and diversification property is assured. Before the second procedure starts all points attained from the first update process are ranked according to equation 5.In the same way as in the case of first statistical procedure, if $Pb_i < \Theta$ for a new point P_i', the present position of P_i' is changed according to equation (7) and (8) otherwise there is no updated.

$P_i'' = P_i' - \hat{\Theta} \times (P_t' - P_{best}) \quad | \hat{\Theta} <= 0.5$ (7)

$P_i'' = P_i' + \hat{\Theta} \times (P_t' - P_s') \quad | \hat{\Theta} >= 0.5$ (8)

P_t and P_s' are random selected points attained from first procedure and $\hat{\Theta}$ is created by Gaussian normal distribution.

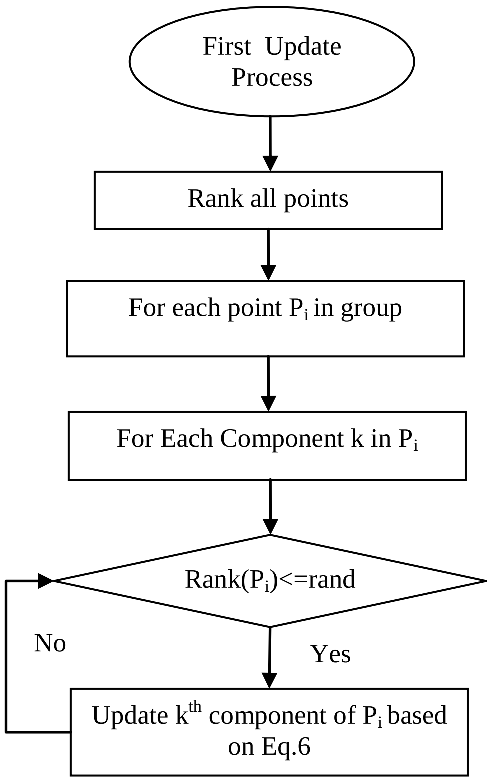
First Update Process
Rank all points
For each point P_i in group
For Each Component k in P_i
Rank(P_i)<=rand
No
Yes
Update k^{th} component of P_i based on Eq.6

Fig. 3 Flowchart for First Update Process

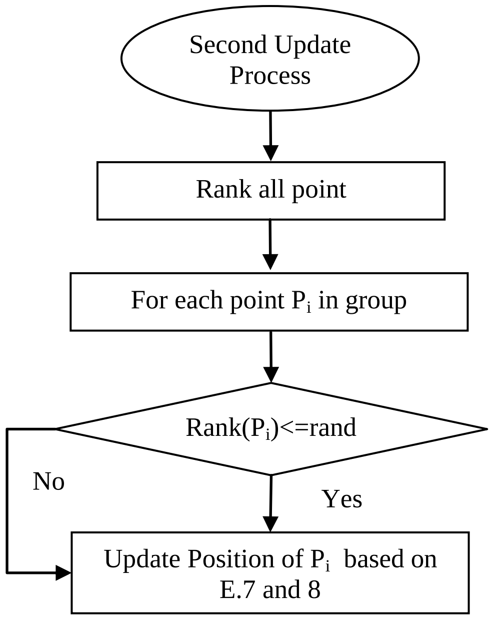
Second Update Process
Rank all point
For each point P_i in group
Rank(P_i)<=rand
No
Yes
Update Position of P_i based on E.7 and 8

Fig. 4 Flowchart for Second Update Process

SFS_DBQ Algorithm

```
Read Input Data file to set DDBMS data allocation and network communication parameters

Initialize a population of N points

While g < maximum generation Do
{
For each Point $P_i$ in the population Do
{
Call Diffusion Process with the following method:
{
md = (maximum considered number of
diffusion).
For j = 1 to md Do
{
If (the first Gaussian walk is applied to solve
the problem)
{
Create a new point based on Eq.(1).
}
Else (user sets the second Gaussian Walks to
solve the problem)
{
Create a new point based on Eq.(2).
}}}}
Call Update Process with the following procedure
{
First Update Process:
First, all points are ranked based on Eq.(5)
For each Point $P_i$ in the system Do
```

```
{
For each component k  in $P_i$ Do
{
If rand[0,1] >= $Pb_i$
{
Update the component of $P_i$ based on Eq. (6)
}
Else
{
Do nothing.
}}
Second Update Process:
Once again, rank all points obtained by the first update process based on Eq. (5)
For each new point $P'_i$ in the system Do
{
If rand[0,1] >= $Pb_i$
{
Update the position based on Eqs (7)and (8)
}
Else
{
Do nothing.
}}}
```

V. CONCLUSION

This paper has offered new framework for distributed query optimization based on stochastic fractal search algorithm. In this approach two process have been used diffusion and update. In the first process new particles are created from existing particles and in the second process the position of particles are updated in order to reduce the overall communication costs of the queries. In the near future we will design and implement this proposed technique in MATLAB tool with the help of MATLAB toolbox and the effectiveness of this proposed technique will be measured.

REFERENCES

[1] Xiaofeng Li, Dong Le ,Hong Zhi Gao and Lu Yao,"*Study of Query of Distributed Database Based on Relation Semi Join*", International Conference On Computer Design And Appliations (ICCDA),IEEE,2010.

[2] Jiunn-Chin Wang, Jorng-Tzong Horng, Yi-Ming Hsu, and Baw-Jhiune Liu , "*Genetic Algorithm for Set Query Optimization in Distributed Database Systems*",IEEE,1996.

[3] Michael Gregory , "*Genetic Algorithm Optimization of Distributed Database Queries*",IEEE,1998.

[4] Hongxing Li and Bingzhang Luo , "*A Tree-based Genetic Algorithm for Distributed Database*", International Conference on Automation and Logistics Qingdao, China September 2008.

[5] Alaa Aljanaby, Emad Abuelrub, and Mohammed Odeh, "*A Survey of Distributed Query Optimization*", The International Arab Journal of Information Technology, Vol. 2, No. 1, January 2005.

[6] Hamid Salimi, "*Stochastic Fractal Search: A powerful metaheuristic algorithm*", Elsevier,2014.

[7] Ms. Preeti Tiwari and Dr. Swati V.Chande,"*Optimization Of Distributed Database Queries Using Hybrids Of Ant Colony Optimization Algorithms*", International Journal of Advances in Engineering Sciences Vol.3 (3), June, 2013.

[8] Jiunn-Chin Wang, Jorng-Tzong Horng, Yi-Ming Hsu, and Baw-Jhiune Liu," *A Genetic Algorithm for Set Query Optimization in Distributed Database Systems",*IEEE,1996.

[9] Deepak Sukheja and Umesh Kumar Singh, "*A Novel Approach of Query Optimization for Distributed Database Systems*", International Journal of Computer Science Issues(IJCSI), Vol. 8, Issue 4, No 1, July 2011.

[10] Manik Sharma ,Rajinder Singh Virk, Gurvinder Singh and Gurdev Singh, "*Design and Comparative Analysis of DSS Queries in Distributed Environment*", IEEE,2013.

[11] M.Tamer Ozsu, Patrick Valduriez, "*Principles of Distributed Database Systems*", Third Edition,Springer,2011.

[12] Ms. Preeti Tiwari and Swati V. Chande "*Query Optimization Strategies in Distributed Databases*", International Journal of Advances in Engineering Sciences Vol.3 (3), July, 2013.

[13] Rajinder Singh, Gurvinder Singh and Varinder Pannu, "*A Stochastic Simulation of Optimized Access Strategies for a Distributed Database Design*", IJSER, Vol 2, Issue 11, November-2011.

[14] Ishfaq Ahmad, Kamalakar Karlapalem,Yu-Kwong Kwok and Siu Kai So," *Evolutionary Algorithms for Allocating Data in Distributed Database Systems*",2002.