IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 2, Ver. III (Mar-Apr. 2016), PP 125-133
www.iosrjournals.org

A Method of View Materialization Using Genetic Algorithm

Debabrata Datta[1], Kashi Nath Dey[2], Payel Saha[3], Sayan Mukhopadhyay[4], Sourav Kumar Mitra[5]

[1]*(Department of Computer Science, St. Xavier's College (Kolkata), India)*
[2]*(Department of Computer Science and Engineering, University of Calcutta (Kolkata), India)*
[3]*(Department of Computer Science, St. Xavier's College (Kolkata), India)*
[4]*(Department of Computer Science, St. Xavier's College (Kolkata), India)*
[5]*(Department of Computer Science, St. Xavier's College (Kolkata), India)*

Abstract: *A data warehouse is a very large database system that collects, summarizes and stores data from multiple remote and heterogeneous information sources. Data warehouses are used for supporting decision making operation on an intelligent system. The decision making queries are complex in nature and take a large amount of time when they are run against a large data warehouse. To reduce the response time, materialized views can be used. Since, all possible views cannot be materialized due to space constraints, an optimal subset of views must be selected for materialization. An approach of selecting such subsets of views using Genetic Algorithms is proposed in this paper. This approach computes the top-T views from a multidimensional lattice and calculates the fitness values for each of them.*

Keywords: *Data Warehouse; Genetic Algorithms; Fitness Function; Materialized View; Multidimensional Lattice*

I. Introduction

Data warehouses are very large database systems that collect, summarize, and store data from multiple remote and heterogeneous information sources, for the purpose of supporting decision making. The queries for decision making are usually analytical and complex in nature and their response time is high when processed against a large data warehouse. This query response time can be reduced by materializing views over a data warehouse [1]. Since all views cannot be materialized, due to space constraints, an optimal subset of views should be selected for materialization. Materialized views are used in data warehouses instead of views because they store results of a query in a separate schema object which can be used to answer future queries without the requirement of calculating the result again, unless the base tables have been updated, and use of materialized views decreases the response time of the end user application. In this paper a Genetic algorithm has been proposed for the selection of views for materialization.

II. Related Work

Materialized view selection is the problem of selecting appropriate sets of views for materialization in the data warehouse such that the cost of evaluating queries is minimized subject to given space constraints. According to the definition given in [2] view selection is defined as "given a database schema R, storage space B, and a workload of queries Q, choose a set of views V over R to materialize, whose combined size is at most B". It is not feasible to materialize all possible views as the number of possible views is exponential in the number of dimensions, and for higher dimensions it is not possible to store all possible views within the available storage space. So, an optimal subset of views should be selected for materialization. There are many different algorithms to select views for materialization. Genetic algorithm (GA) is one of them. The majority of research in the area of materialized view selection is focused around greedy heuristics based techniques. These techniques are unable to select good quality views for higher dimensional data sets because their total view evaluation cost is high. GA is n widely used evolutionary technique suitable for solving complex problems involving the identification of a good set of solutions from within a large search space [3]. Several GA based view selection algorithms have been proposed in literature [5, 6, 7, 8, 9, 10]. These algorithms aim to select views for higher dimensional data sets with the key challenge of selecting views of high quality i.e. low total view evaluation cost (TVEC). In this paper a GA based algorithm is proposed that selects views from a multidimensional lattice. Each chromosome is represented as a string of views selected for materialization. The length of each chromosome is T for selecting top-T views for materialization. GA is applied with a pre-defined crossover and mutation probability and a set of top-T views are generated after a pre-specified number of generations.

III. Basics Of Genetic Algorithm

Genetic algorithms (GAs) are adaptive heuristic search methods based on the evolutionary ideas of natural selection and genetics. They are inspired by Darwin's theory about evolution – "Survival of the fittest." They represent an intelligent exploitation of random search used to solve optimization problems. GAs, although randomized, exploit historical information to direct the search into the region of better performance within the search space. In solving problems, some solution(s) will be the better than others. The set of all possible solutions is called search space or state space. Each point in the search space represents one possible solution, which can be marked by its value. This is known as the fitness value for the problem. GA looks for the best solution among a number of possible solutions.

Working Principle:

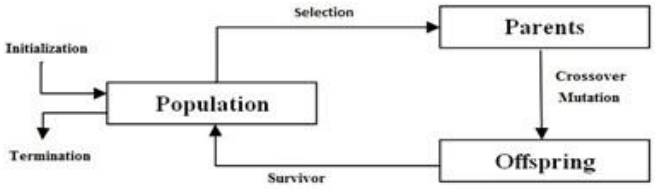

Fig.1. Working principle of Genetic algorithm

1. Initialization: An initial population of all chromosomes is usually generated randomly within the search space.
2. Evaluation: Once the population is initialized or an offspring population is created, the fitness values of all individuals within that population are evaluated using a proper fitness function.
3. Selection: Selection of two parents with higher fitness values for recombination. The main idea is to prefer better solutions to worse ones.
4. Recombination (or Crossover): Recombination combines parts of two or more parental solutions (or individuals) to create new, possibly better solutions (offspring).
5. Mutation: Mutation basically alters one or more gene values in a chromosome to maintain genetic diversity from one generation of a population to the next.
6. Replacement: The offspring population created by selection, recombination and mutation replaces the original parental population.
7. Repeat steps 2–6 until a terminating condition is met.

IV. Proposed Method

The proposed GA based method selects the top-T views from a multidimensional lattice as discussed below:

A. The Multidimensional Lattice

Views involved in On-Line Analytical Processing (OLAP) queries can be represented as nodes of a multidimensional lattice [4, 11]. The multidimensional lattice is a graph with no self-loops or parallel edges. The root node of the lattice represents the base facts table. All the views in the lattice either directly or indirectly depend on the root view. A view V_A is said to be dependent on another view V_B when the queries on V_A can be answered using V_B. In the graph, direct dependencies are represented by an edge between two views & indirect dependencies are discovered transitively.

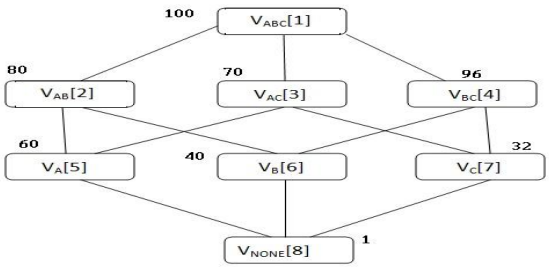

Fig.2. A 3-Dimensional lattice

In Fig. 2, a 3-dimensional lattice is shown. The index is shown in brackets inside the node & the frequency of the variables is shown on the top left corner of each node. The root view contains all the variables A, B & C.

B. Proposed Algorithm

1. [Start] Generate random population of n chromosomes.
2. [Fitness] Use the fitness function f(x) to calculate the fitness value of each chromosome x in the population.
3. [Test] If the end condition is satisfied, i.e. , a chromosome with the desired fitness value is found in the population or a value close to that value is found (& further repetition does not generate a chromosome with a better fitness value) then stop & return the best solution that is found.
4. [New Population] Create a new population by repeating the following steps until the new population is complete.

a. [Selection] Select two parent chromosomes from a population according to their fitness value. The higher the fitness value, the better the chances for it to be selected.
b. [Crossover] With a crossover probability p_c, combine the parent chromosomes to generate a new offspring.
c. [Mutation] With a mutation probability p_m, mutate the offspring at each locus.
d. [Accepting] Place the offspring in the new population.

5. [Replace] Replace the new population with the previous population & go to step 2.

C. Chromosome Representation

Each chromosome is represented as a string of distinct views. Each distinct gene corresponds to a view. The chromosome string contains only the index values of those views that are materialized. The chromosome representation for selecting the top-5 views for materialization is shown below:

3	5	1	4	8
1	7	2	3	5
8	6	2	1	4

Fig.3. Chromosome representation for selecting top-5 views

D. Fitness Function

The fitness function is used to give an estimate of how much fit a chromosome is for being selected for crossover. The quality of solution obtained by the GA based algorithm depends on the correctness & the appropriateness of the fitness function. Here, the fitness function is based on the frequency of attributes used in a view. The higher the total frequency of a chromosome, the better is its chances to be selected for crossover. The fitness function is given below:

$$TFV = \sum_{i=1}^{x} Mat(\mathrm{V_i}) + \sum_{j=1}^{y} AncNMat(\mathrm{V_j}) \text{ , where } i \neq j$$

TFV = Total Frequency Value
n = Total number of genes or views
x = Number of materialized views
y = Number of non-materialized views
n = x + y
$Mat(V_i)$ = Frequency of attributes for view V_i where V_i is materialized
$AncNMat(V_j)$ = Frequency of a materialized ancestor with maximum frequency among other ancestors of a non-materialized view V_j
For example, if views V_{AC}, V_{BC}, V_A, V_B & V_C are selected for materialization and, x=5 & y=3 then the TFV computation is as follows:

$$\sum_{i=1}^{5} Mat(V_i)$$

$= Mat(V_{AC}) + Mat(V_{BC}) + Mat(V_A) + Mat(V_B) + Mat(V_C)$
$= 70 + 96 + 60 + 40 + 32$
$= 298$

$$\sum_{j=1}^{3} Mat(V_j)$$

$= Mat(V_{ABC}) + Mat(V_{AB}) + Mat(V_{NONE})$
$= 0 + 0 + 60$
$= 60$

$$TFV = \sum_{i=1}^{5} Mat(V_i) + \sum_{j=1}^{3} Mat(V_j)$$
$$= 298 + 60$$
$$= 358$$

E. Fitness Function Algorithm

The fitness function takes the following parameters as its inputs:

1. The adjacency matrix adj[][] for the entire multidimensional lattice,
2. freq[] array which contains the frequencies of attributes for each view,
3. **level[]** which contains the level of each view in the graph(lattice),
4. The total number of distinct genes (tot_gene),
5. The number of genes to be materialized (mat_gene), and
6. The size of the population popl.

- The procedure RAND (1, tot_gene) generates a random number between 1 & n where n is the total number of distinct genes.
- The procedure LINEAR (topt, k, i, mat_gene) checks if the item k is present in the i^{th} chromosome or not. If it is, it returns true. Otherwise, it returns false.
- The procedure MAX (anc) finds out the maximum number in the anc [] array, i.e. the materialized ancestor of a non-materialized gene which has the maximum frequency.

The algorithm proceeds in the following way:

- First it generates the initial population and stores the chromosomes in array **topt [][].**
- Calculates total frequency value for each chromosome in the population. This value is termed asTFV1.
- Finds out the non-materialized views in each chromosome and finds the maximum frequency materialized ancestor for each of those non-materialized views.
- Calculates the sum of attribute frequencies of these ancestors for each chromosome. This value is termed as TFV2.
- Add TFV1 and TFV2 for each chromosome to generate the Total Frequency Value (TFV).

Algorithm FITNESS (adj[][], freq[], level[], tot_gene, mat_gene, popl)

```
begin
        for i = 1 to popl do
          for j = 1 to mat_gene do
            while (true) do                    //Loop for removing duplicate genes
                flag = false;
                rand = RAND(1, tot_gene);
                        for k = 1 to j do
                                if (topt[i][k] == rand)  then      //If the gene is already present in the chromosome then generate a new number
                                        flag = true;
                                        break;
                        end for
                        if (flag == false) then
                           topt [i][j] = rand;
                           break;
                end while
         end for
end for
        for i = 1 to popl do
                tv1=0;
                        for j = 1 to mat_gene do
                                k = topt[i][j];
                                tv1 = tv1 + freq[k];
                        end for
```

```
                tfv1 [i] = tv1;
        end for
        flag = false;
        for i = 1 to popl do
                k = 1; l = 0;
          while (k <= tot_gene  and  l < n_mat_gene) do
                flag = LINEAR(topt, k ,i, mat_gene);
                if(flag == false) then
                        mma [i][l]=k;
                        l++;
            k++;
          end while
    end for
        for i = 1 to popl do
                tv2 = 0 ;
                for j=1 to n_mat_gene do
                k = mma[i][j];
                q=0;
                anc[]={0,0,0,0,0};
                for  p= 1 to tot_gene do
                        if(level[k]==1)
                                break;
                        else if ((adj[k][p] == 1) and (level[p] == (level[k]-1))) then          //If entry in
adjacency matrix is 1 & the node is in the                      //previous level then it is an ancestor
                        anc[q++] = freq[p];                     //Add the ancestor to anc[]
                end for
                m =MAX(anc);    //Calculate the freq that is max because we need max freq ancestor
                 tv2 = tv2 + m;
        end for
tfv2[i]=tv2;
end for
for i=1 to popl do
                TFV[i]=tfv1[i]+tfv2[i];
        end for
end
```

F. Selection

Selection is a process of choosing individuals from a population for performing crossover. The fitter individuals are more likely to produce fitter offsprings in the subsequent generations. Selection of only fitter individuals might hinder exploration of the search space thereby leading to early convergence. Therefore, there is a need to randomly select individuals where individuals, having higher fitness value, have greater likelihood of being selected for crossover. The approach uses binary tournament selection method where two individuals are randomly selected from the population and a tournament is conducted among them. A value r is randomly generated between 0 and 1. If r is less than the pre-defined value of k (say k=0.75), taken between 0 and 1, the individual having higher TFV is selected else the individual with lower TFV is selected.

Chromosomes (top-T Views)	Fitness value(TFV)
[3 6 7 8 2]	303
[2 5 4 1 8]	629
[2 3 1 8 6]	541
[7 4 8 5 1]	595
[7 3 8 1 6]	513
[6 2 1 7 4]	568

Fig.4. TFV of top-T views in V_{TopT}

Selection Algorithm:

1. randomly two individuals from the population.
2. Given: parameter k = 0.75.
3. Choose Choose: a Random number r between 0 and 1.
4. If r < k, Select the fitter individual among the two individuals.

5. Else, Select the less fitter individual among the two individuals.

Randomly generated Indexes[i] [j]	Tournament between individuals [P(i)] [P(j)]	Fitness [TFV(P(i))] [TFV(P(j))]	Random Number (r)	Individual selected
[2] [3]	[2 3 1 8 6] [7 4 8 5 1]	541 585	0.9765590312995504	[2 3 1 8 6]
[5] [0]	[6 2 1 7 4] [3 6 7 8 2]	303 568	0.7209919519618984	[3 6 7 8 2]

Fig.5. Selection of top-T views using Binary Tournament Selection

G. Crossover

In Crossover, two individuals are randomly selected and are recombined using single point crossover method with crossover probability, p_c= 0.5. A uniform random number, r, is generated and if $r \leq p_c$, the two randomly selected individuals undergo crossover. Otherwise, the two offspring are simply copies of their parents [12].

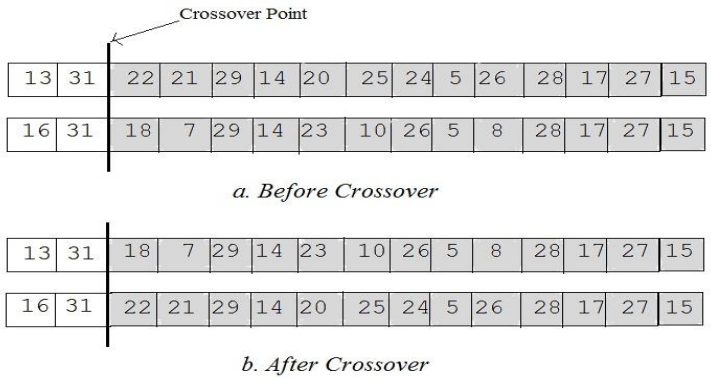

Fig.6. Single Point Crossover

Crossover Algorithm:

1. Select two parents randomly for crossover from the population
2. Generate random number r
3. If r < = pc then do

a. Randomly generate a point for crossover
b. Perform single point crossover between the parents corresponding to the randomly generated point
c. If there are duplicate genes within either of the children after crossover, then repeat Step 1-3
d. Else If any of the children have already been generated before to be placed in the new population then repeat steps 1-3

Else Do not perform crossover. Simply copy the parents to their corresponding children.

H. Mutation

Mutation is performed using random resetting method with predefined mutation rate, p_m. A random number rand is generated & if it is less than the mutation rate only then we perform mutation. Otherwise, we do not perform mutation. So, the actual number of genes to be mutated is not fixed. For a chromosome of length L, on average $L*p_m$ genes will be mutated [13]. The method ensures that there are no duplicate genes in the chromosome after mutation.

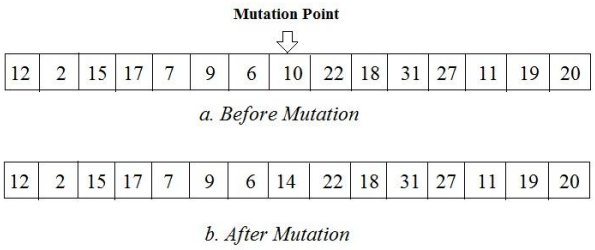

Fig.7. Mutation

Mutation Algorithm:

1. for i=1 to L do
    a. Generate rand
    b. if rand < pm then
        i. Generate a random gene value
        ii. Mutate gene[i] of the selected chromosome with the random gene value generated in the previous step
        iii. if the newly added gene is a duplicate then repeat steps i and ii
2. Repeat Step 1 for all the chromosomes until the end of the population.

I. Replacement

After the generation of the new population we replace the old population with the new population. Then we perform the entire cycle of fitness value calculation, selection, crossover, mutation, and replacement for a pre specified number of generations.

V. Result & Analysis

The GA based view selection algorithm was implemented using jdk 1.7.0_55 in Windows 7(x64) environment. The algorithm was successfully tested for a 3 & a 4 dimensional lattice of views. The result for selecting top-5 views for materialization from a 3 dimensional lattice is shown in Fig. 4. From the figure it can be observed that the matrix for top-T views contains distinct views in each row of the matrix. Duplicate views are eliminated since the data warehouse does not require redundant data for recovery. TFV1 is the first part of the fitness function that calculates the total frequency value for only materialized views. TFV2 is the second part of the fitness function that calculates the total frequency value for non-materialized views. These two are added to give the TFV for each chromosome.

```
The population matrix is:
 6 8 2 7 3
 5 3 2 6 8
 8 7 6 4 1
 5 8 4 6 7
 2 1 5 8 4
 6 2 3 1 7

TFV1: 223
TFV1: 251
TFV1: 269
TFV1: 229
TFV1: 337
TFV1: 322

The TFV1 array is:
 223 251 269 229 337 322
The MMA matrix is:
 1 4 5
 1 4 7
 2 3 5
 1 2 3
 3 6 7
 4 5 8

Ancestor :
 0 0 0 0 0

Ancestor :
 0 0 0 0 0

Ancestor :
 80 70 0 0 0
TFV2: 80

Ancestor :
 0 0 0 0 0

Ancestor :
 0 0 0 0 0

Ancestor :
 70 0 0 0 0
TFV2: 70

Ancestor :
 100 0 0 0 0

Ancestor :
 100 0 0 0 0

Ancestor :
 0 0 0 0 0
TFV2: 200

Ancestor :
 0 0 0 0 0

Ancestor :
 0 0 0 0 0

Ancestor :
 0 0 0 0 0
TFV2: 0

Ancestor :
 100 0 0 0 0

Ancestor :
 80 96 0 0 0

Ancestor :
 96 0 0 0 0
TFV2: 292

Ancestor :
 100 0 0 0 0

Ancestor :
 80 70 0 0 0

Ancestor :
 40 32 0 0 0
TFV2: 220

The TFV2 array is:
 80 70 200 0 292 220

The fitness values of the chromosomes are:
Chromosome 0 :: 303
Chromosome 1 :: 321
Chromosome 2 :: 469
Chromosome 3 :: 229
Chromosome 4 :: 629
Chromosome 5 :: 542
```

Fig.8. Result for selecting top-5 Views from a 3 Dimensional Lattice

The size of the adjacency matrix increases exponentially with the increase in lattice dimensions. Hence, with the increase in dimensions of the lattice, the complexity increases. When we calculate the attribute frequencies for a non-materialized view, we use the frequency of attributes of the maximum materialized ancestor of that non-materialized view because the non-materialized view is dependent on the ancestor. This means that all the queries for the non-materialized view can be answered using its materialized ancestor, without requiring any other views to be materialized. But, if the root view itself was not materialized then we use a frequency value of zero in the calculation because there are no other views in the lattice which can give answers

to all the queries that can be made to the root view as no other view but the base or root view contains all the attributes.

A frequency value of zero is also used when a non-materialized view has no ancestors. This is because only an ancestor view that the child view is dependent on or the child view itself can give answers to all the queries that can be made to the child view.

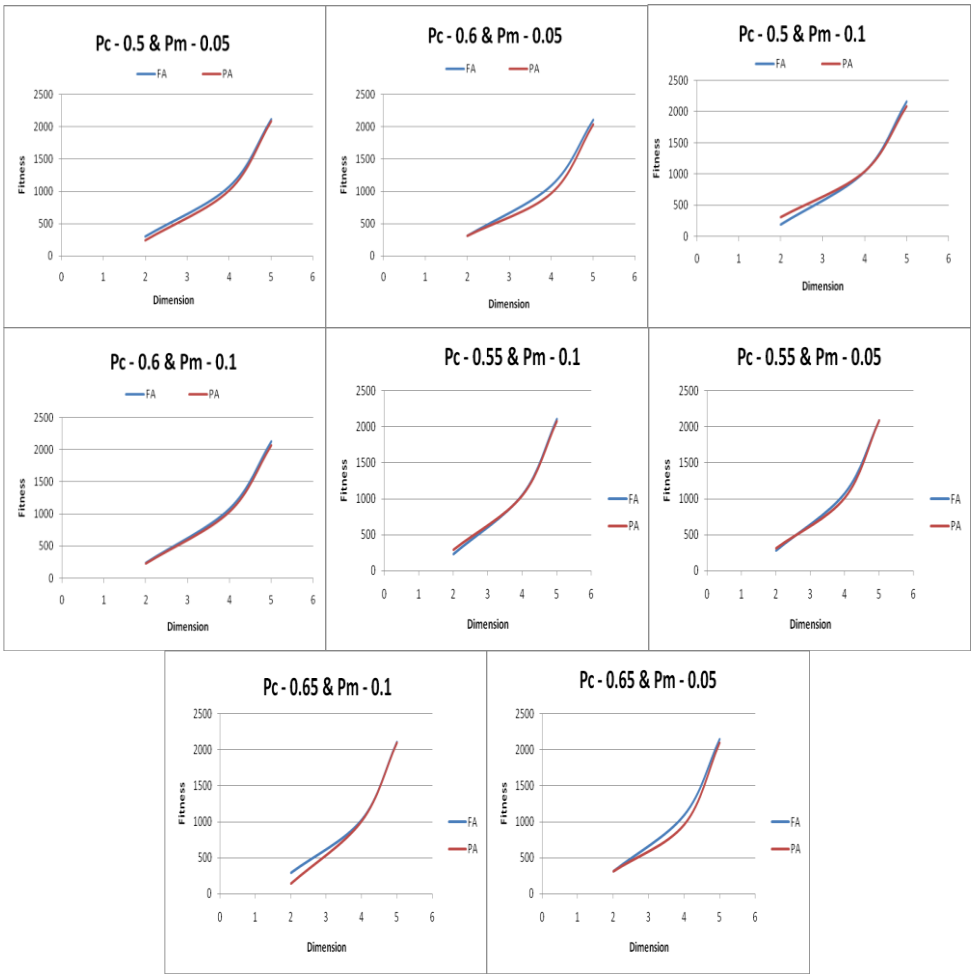

Fig.9. Comparison of FA and PA – Fitness Vs. Dimensions for different Pc's and Pm' s

Our proposed GA based view selection algorithm using frequency of attribute (FA) and another GA based view selection algorithm using size of attribute (PA) are compared. The comparisons are carried out on Fitness value due to views selected by the two algorithms. The experiments were performed for selecting the top-T views for materialization for dimensions 2 to 5 over 1000 generations. First, the graphs showing Fitness value for different crossover and mutation probabilities for selecting top-6 views, were plotted and compared with Fitness value of selecting top-6 views using PA. These graphs, plotted for pair of crossover and mutation probabilities (0.5, 0.05), (0.6, 0.05), (0.5, 0.1), (0.6, 0.1), (0.55, 0.1), (0.55, 0.05), (0.65, 0.1), (0.65, 0.05), are shown in Fig. 9. The graphs show that FA, in comparison to PA, is able to select views that are Fitter than that obtained by PA for different crossover and mutation probabilities. This difference in Fitness value becomes significant for higher dimensions. Further, it can be observed from the graph that, for crossover probability 0.6 and mutation probability 0.05, the best result is obtained by FA in the Fitness value across all dimensions.

VI. Conclusion

Here we have used Genetic Algorithms & used a fitness function to evaluate the fitness values of the possible solutions. The Genetic Algorithms are comparatively easy to understand & does not require too much mathematical knowledge. Analysis of state of the art of view selection has shown that there is a very few work on view selection in distributed databases and data warehouses. One of challenging directions of future work aims at addressing the view selection problem in a distributed setting. Randomized algorithms can be applied to complex problems dealing with large or even unlimited search spaces. Using GA for selecting views enables exploration and exploitation of the search space. As a result, the views so selected are likely to have a higher TFV. Further experimental results show that the views selected using the proposed GA-based algorithm have a

comparatively higher frequency value to those selected using PA for the observed crossover and mutation probabilities. That is, the GA based algorithm using frequency is able to select comparatively better quality views. This in turn results in reduced query response time enabling efficient decision making. Thus, the use of randomized algorithms such as GAs can be considered in solving large combinatorial problems such as the view selection problem.

References

[1]. Vijay Kumar, T.V., Kumar, S.: Materialized View Selection using Genetic Algorithm, Communications in Computer and Information Science (CCIS), Volume 306, Springer Verlag, pp. 225-237, 2012.
[2]. Chirkova, R., Halevy, A. Y., Suciu, D.: A formal perspective on the view selection problem, 27 VLDB Conference, Italy; 2001
[3]. Goldberg, D. E.: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.
[4]. Harinarayan, V., Rajaraman, A., Ullman, J. D.: Implementing data cubes efficiently, ACM SIGMOD International Conference on Management of Data, pages 205-227, 1996.
[5]. Horng, J.T., Chang, Y.J., Liu, B.J., Kao, C.Y.: Materialized view selection using genetic algorithms in a data warehouse system, in Proceedings of the World Congress on Evolutionary Computation, Washington, pp. 2221–2227, 1999.
[6]. Horng, J.T., Chang, Y.J., Liu, B.J.: Applying evolutionary algorithms to materialized view selection in a data warehouse, Soft Computing.2003, Vol.7, pp.574–581, 2003.
[7]. Lee, M., Hammer, J.: Speeding Up Materialized View Selection in Data Warehouses Using A Randomized Algorithm, in the International Journal of Cooperative Information Systems, Vol. 10, Nos. 3, pp. 327-353, 2001.
[8]. Lin, W.Y., Kuo, I.C.: A genetic selection algorithm for OLAP data cubes, Knowledge and Information Systems 6 (1) pp. 83–102, 2004.
[9]. Wang, Z., Zhang D.: Optimal Genetic View Selection Algorithm Under Space Constraint, In International Journal of Information Technology, vol. 11, no. 5, pp. 44 - 51, 2005.
[10]. Yu, J. X., Yao, Y. X., Choi, C., Gou, G.: Materialized view selection as constrained evolutionary optimization, in the Journal of IEEE Transactions on Systems, Man, and Cybernetics - TSMC, vol. 33, no. 4, pp. 458-467, 2003.
[11]. Mohania, M., Samtani, S., Roddick, J. and Kambayshi, Y.: Advances and Research Directions in Data Warehousing Technology, Australian Journal of Information Systems, Vol 7, No 1; 1999.
[12]. Sastry, K., Goldberg, D., Kendall, G.: Search Methodologies, Introductory Tutorials in Optimization and Decision Support Techniques, Springer US, 2005.
[13]. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing, Springer-Verlag, pages 42-43, 2003.