Optimal Genetic View Selection Algorithm Under Space Constraint

Ziqiang Wang, Dexian Zhang

Department of Computer Science, Henan University of Technology, Zheng Zhou 450052,China

wzqagent@xinhuanet.com

Abstract

Materialized view selection problem is one of the most important decisions in designing a data warehouse. In order to efficiently solve the problem, a modified genetic algorithm for how to select a set of views to be materialized so as to achieve both good query performance and low view maintenance cost under a storage space constraint is proposed. The essential idea of the algorithm is as follows. First, a greedy algorithm based on benefit value and storage space constraint is used to produce initial solution. Then, the initial solutions are improved by genetic algorithm. In addition, the invalid solutions during the evolution search are repaired by loss function. The experimental results show that the proposed algorithm is superior to heuristic algorithm and classical evolutionary algorithm.

Keyword: Data Warehouse, Genetic Algorithm, View Selection, Space Constraint.

I. Introduction

A data warehouse contains multiple views accessed by queries. Since queries may share some intermediate results, materialized view[1], which is to pre-compute frequently asked queries and store them for future use, is a useful technique for query answering in a data warehouse. But materialized views occupy the disk space and we must maintain the consistence of them. Selecting views to be materialized is one of the most important decisions in designing a warehouse. The materialized view selection problem can be described as follows: Given a set of queries *Q* and a quantity S (available storage space), the view selection problem is to select a set of views *M* to be materialized, that minimize total query response time and the total maintenance cost under the constraint that the total space occupied by *M* is less than *S*. Materialized view selection problem has to been proved to be NP-hard[2].

The materialized view selection problem has attracted substantial attention in the research community. Gupta proposed a systematical methodology for the selection of materialized views in which he extended the aggregated queries into a more general form that could be represented by AND-OR graph and presented several heuristic algorithms for view selection[1,3]. However, in this way, there are some disadvantages that the search time is very long and only near-optimal solution can be achieved. Ref.[4] modeled the view selection problem as a state space optimization problem

and provided exhaustive and heuristic algorithms, but they did not concern for the storage constraint. Ref.[5] presented and analyzed algorithms for selection of views in the special case of ”data cubes”, but their works ignored maintenance costs. Ref.[6] modeled the view selection problem based on MVPP(Multiple View Processing Plan) and provided heuristic algorithm and 0-1 integer programming algorithm to generate MVPP. Recently, genetic algorithm(GA) based on the mechanics of natural selection and survival of fitness has been applied to view selection problem[7-10].But the drawbacks of the canonical genetic algorithm lie in that it is hard to acquire good initial solutions, and those genetic view selection algorithms use a penalty function to punish the fitness of the infeasible solutions. Research has shown that the repair scheme is better in dealing with invalid solutions than penalty function[11]. In this paper, the GA method is further refined by modifying genetic operators and the repairing scheme of infeasible solutions . The experiment results show that the proposed algorithm is superior to heuristic algorithm and conventional genetic algorithm in finding optimal solutions.

The rest of this paper is organized as follows. Section 2 explains and formulates the model of materialized view selection. In section 3, we apply genetic algorithm to solve the model. Section 4 gives the experimental results using our genetic algorithm. Conclusions are summarized in Section 5.

II. Materialized View Selection Problem Formulation

A. *AND-OR View Graph Definition*

Definition 1 Expression A-DAG. An expression A-DAG(AND-DAG) for a query or a view is a directed acyclic graph having the base relations as ”sinks”(no outgoing edges) and the node *V* as a ”source”(no incoming edges).If a node/view *u* has outgoing edges to nodes $v_1, v_2, \cdots, v_k$,then all of the views $v_1, v_2, \cdots, v_k$ are required to compute *u* and this dependence is indicated by drawing a semicircle, called an AND arc, through the edges $(u, v_1), (u, v_2), \cdots, (u, v_k)$ .Such an AND arc has an operator and a cost associated with it, which is the cost incurred during the computation of *u* from $v_1, v_2, \cdots, v_k$ .And the evaluation cost of a node *u* in an expression A-DAG is the sum of the costs associated with each of its descendant’s AND arc.

Definition 2 Expression AO-DAG. An expression AO-DAG(AND-OR-DAG) for a view or a query *V* is a directed acyclic graph with *V* as a source and the base relations as sinks. Each non-sink node has associated with it one or more AND arcs, each binding a subset of its outgoing edges.

Definition 3 AND-OR View Graph. A graph *G* is called AND-OR view graph for the views (or queries) $V_1, V_2, \cdots, V_k$ if for each V_i there is a sub-graph G_i in *G* which is an expression AO-DAG for V_i .Each node *u* in an AND-OR view graph has the following parameters associated with it: f_u (frequency of the queries on *u*), S_u (space occupied by *u*), and g_u (frequency of updates on *u*).

With the above definitions, an AND-OR view graph for a data warehouse could be defined and constructed by a given set of queries $Q_1, Q_2, \cdots, Q_k$ to be supported at a warehouse. We constructed an AND-OR view graph for the queries in the following steps. First, we construct an expression AO-DAG D_i for each query Q_i in the set. Then, all the expressions AO-DAGs $D_1, D_2, \cdots, D_k$ are merged to get an AND-OR view graph G for the set of queries. Note that,

each node in the AND-OR view graph G will represent a view that could be selected for materialization, and these are the only views considered for materialization.

B. *Mathematical Model of Materialized View Selection Problem*

Let $Q(u,M)$ denotes the cost of answering a query u (a node of G) using the set M of materialized views in the given view graph G. $Q(u,M)$ is the evaluation cost of the cheapest embedded expression A-DAG for u in G whose sinks belong to the set $M \bigcup L$,where L is the set of sinks in G. Let $U(u,M)$ denotes the maintenance cost for the view u in the presence of the set of materialized views M and the set of sinks, L. So, given an AND-OR view graph G and the size of storages S, the view selection problem is to select a set of views/nodes $M = \{V_1, V_2, \cdots, V_m\}$ that minimize $\tau(G,M)$.

$$\tau(G,M) = \sum_{i=1}^{k} f_{Q_i} \cdot Q(Q_i,M) + \sum_{i=1}^{m} g_{V_i} \cdot U(V_i,M) \quad (1)$$

under the constraint that $\sum_{v \in M} S_v \le S$ .Where $\sum_{i=1}^{k} f_{Q_i} \cdot Q(Q_i,M)$ is the sum of query response time, and $\sum_{i=1}^{m} g_{V_i} \cdot U(V_i,M)$ is the sum of view maintenance cost.

III. Modified Genetic Algorithm for View Selection

A. *Initial Solution Construction*

Before starting the genetic algorithm, we first present pre-process algorithm to produce initial solution (view set M) under the constraint that the total space occupied by M is less than S. Then, the solutions are improved by genetic algorithm. In the following, we give greedy algorithm description based on benefit definition and storage space constraint.

Definition 4 Benefit of Selected View Set. Let C be an arbitrary set of views in a view graph G. The benefit of C with respect to M, an already selected view set, is denoted by $B(C,M)$ and is defined as $\tau(G,M) - \tau(G,M \cup C)$,where $\tau(,)$ is the function defined by Equation (1).The benefit of C per unit space with respect to M is $B(C,M)/S(C)$,where $S(C)$ is the space occupied by the views in C.

Algorithm 1 : Initial solution construction algorithm.
Input: An AND-OR view graph G, the space constraint S.
Output: The set of materialized views M.
Begin
$M = \varnothing$;
While $(S(M) < S)$
For every view C in view graph G, calculate its benefit per unit space with respect to M according to Definition 4. Let C be the view which has the maximum benefit per unit space with respect to M.
If $\left(S(M) + S(C) < S\right)$ Then
$M = M \cup C$; $G = G - C$;
End While;
Return M;

End.

B. *Chromosome Encoding Representation*

Each chromosome is consisted of constant number of binary bit string (0 or 1), each AND-OR view graph is encode as a binary string. Algorithm 2 shows how to map an AND-OR view graph into a binary string. The main reason of using binary strings, rather than graphs directly is to simplify the implementation of genetic algorithm (including selection, crossover and mutation).

Algorithm 2: Mapping AND-OR view graph into binary strings.
Begin
Input a set of queries represented by a AND-OR view graph;
Use breadth-first graph traversal strategy to traverse through all nodes in the AND-OR view graph and produce an ordered list of nodes;
Create a binary string according to this order, where 0 denotes that the corresponding node is not materialized and 1 indicates that the corresponding node is materialized;
Return binary string;
End.

C. *Fitness Function and Evolutionary Operator Design*

A fitness function evaluates the degree of fitness of each chromosome. Since the fitness function of GA is usually defined as maximization while the objective in view selection mathematical model is to minimize the sum of query and maintenance cost, we adopted the following simple transformation to define the fitness function from the cost function in Equation (1).

$$F(G,M)=\begin{cases}\tau_{\max}-\tau(G,M): & if\ \tau_{\max}>\tau(G,M)\\ 0: & otherwise\end{cases} \tag{2}$$

where $F(G,M)$ is the fitness function, $\tau(G,M)$ denotes the cost function defined in Equation(1),coefficient $\tau_{\max}$ denotes the largest $\tau(G,M)$ value in the current population.

We use a mixture of selection methods for reproducing the chromosomes. The first selection method is an elitism that the best chromosome with the highest fitness value is passed in the new population. The second selection method is a modified *k*-tournament method[11].In this method, a chromosome having the best fitness value among the *k* chromosomes randomly selected from the upper class of chromosomes according to their fitness values is chosen for the reproduction. Two chromosomes C' and C'' obtained by repeating the above procedure consecutively create a new chromosome *C* by applying the one-point crossover and bit-flipping mutation operations. New chromosome *C* is replaced by a certain chromosome $\tilde{C}$ that has the worst fitness value among the *k* chromosomes randomly selected from the lower class of chromosomes according to their fitness values. The above reproduction procedure is repeated as many times as $p_c \times Pop$, where p_c is the crossover probability, and Pop is the population size. Finally, the remaining portion of the population set is filled by copying the population set in the order of magnitude of fitness values.

We adopted crossover operator was half-uniform crossover where the bits at each location of the selected pair of strings are compared. If the bit at a particular location is the same in both strings, it is kept intact. If the bit differs, a new bit is taken from either string with equal probability. To maintain diversity during the initial generations of the GA, the Hamming

Distance[12] between pairs selected from the breeding population is used to ensure very similar bit strings do not undergo crossover. This helps avoid the situation of a fit bit string becoming dominant before a reasonable amount of the search space has been investigated. So, when two strings are selected to undergo crossover the Hamming Distance between the strings is determined and if it is less than the threshold they are returned. If it is greater than the threshold, they are allowed to undergo crossover.

Mutation is used to create new genes that may not be present in any member of a population and enabled the genetic algorithm to reach all possible solutions in the search space. Mutation is implemented as a bit-flipping operator in this study: Each locus of an individual is subjected to mutation with probability $p_m = 0.02$ and the bit value at that locus is changed from 0 to 1 or vice versa.

D. *Dealing with Invalid Solutions*

Because GA does not embed the constraint into its encoding scheme and the storage space of data warehouse is not enough to materialize all views, infeasible solutions may be generated during the evolutionary search, e.g., by crossover and/or mutation. To avoid the infeasible solutions, we proposed a repair algorithm based on loss function to do this. We correct the genes of an infeasible chromosome each time by choosing the views with the least loss function value, that is, the elimination of this view will increase the least cost per space unit. In the following, we first define loss function, then the repair algorithm of infeasible solutions is proposed.

Definition 5 Loss function. The total cost increase caused by the elimination of a view *c* relative to some selected set of views *M* is denoted loss function $L(c,M)$, its formal definition is as follows:

$$L(c,M) = \frac{1}{|c|}\left(L_1(c,M) + L_2(c,M)\right) \tag{3}$$

$$L_1(c,M) = \sum_{c_i \in Q} f_{c_i} \cdot \left[Q(c_i,M) - Q(c_i,M-\{c\})\right] \tag{4}$$

$$L_2(c,M) = \sum_{c_i \in M-\{c\}} g_{c_i} \cdot \left[U(c_i,M) - U(c_i,M-\{c\})\right] \tag{5}$$

where *Q* denotes the set of queries, function $Q(,)$ and $U(,)$ definitions are illustrated above section.

Algorithm 3: Infeasible solutions repair algorithm.

Begin
Let an individual *c* that needs to be repaired be $(x_1, x_2, \cdots, x_n)$, and the corresponding set of selected views be $M = \{c_i \mid x_i = 1, 1 \le i \le n\}$;
While $\sum_{c \in M} S_c > S$ do
For each $c_i \in M$ do
Compute the loss function $L(c_i, M)$;
Let c_j be the view with minimal loss function value;

$x_j = 0$; $M = M - \{c_j\}$;
End While;
End.

E. Algorithm Description for Materialized View Selection

Based on analysis the primary issues involved in applying GA to the view selection problem, the modified genetic algorithm for view selection is outlined as follows.

Algorithm 4: A modified GA for materialized view selection

Begin
Initialize the system parameters;
Construct initial solution(population) *P* according to Algorithm 1;
$t = 1$;
While $t \le Max\_Iteration$ do
Clear the new population P';
For each individual $c \in P$ do;
If *c* is not feasible solution Then Call for Algorithm 3 to repair *c*;
Calculate the fitness of *c*;
End For;
While $|P'| \le Pop\_Size$ do
Select two parents from population *P* and perform evolutionary operation according to operator design in Section C.
Place the offspring into population P';
End While;
$P' \to P$;
$t = t + 1$;
End While;
End.

IV. Experimental Results Comparison

To evaluate the effectiveness of the modified genetic algorithm for materialized views selection, we compared it with the heuristic algorithm[1] and canonical genetic algorithm[8]. All our experiments were performed on Intel Pentium 4(CPU 2.2G,RAM 512M) processor under Windows Server 2000. We used the test data sets in TPC-D benchmark[13], which is a database generator following the proposal of the transaction processing performance council in 1995, and is dedicated to decision support application. We used 1GB database.

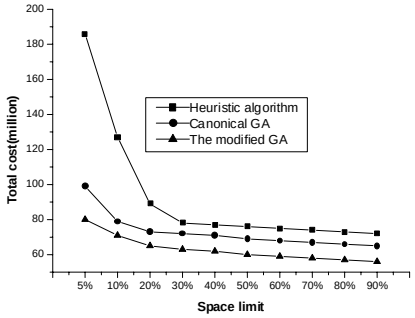

Fig. 1. Algorithm comparison with random frequencies

The parameter settings used in the modified genetic algorithm are as follows. Max generation is 200, population size 200,crossover probability $p_c = 0.80$, mutation probability $p_m = 0.02$. Without loss of generality, we consider view random invoking frequencies between 0 and 1,for the space constraint, we considered 10 different values set as 5%,10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, and 90% of the total size of all views respectively, the experimental results are depicted in Figure 1.

The experimental results show that the modified GA outperforms heuristic algorithm and conventional GA in all cases regardless of the storage space, which illustrated the advantage of having the mixture of optimal strategies to search a larger part of a huge space and thus find a better solution, these optimal strategies include pre-process algorithm for producing initial solutions, improved genetic operator design and repair algorithm of infeasible solutions.

V. Conclusion

Materialized view selection problem is one of the most challenging problems in data warehouse. In this paper, we have addressed how to select of views to be materialized so that the sum of processing a set of queries and maintaining the materialized views is minimized. A modified genetic algorithm having the mixture of optimal strategies and infeasible solutions repair algorithm is proposed to solve view selection problem. The experimental results show that the modified GA is superior to heuristic algorithm and canonical GA in finding better solutions.

References

[1] H. Gupta,Selection of views to materialize in a data warehouse. In Proc. 6th Int. Conf. on Database Theory (ICDT’97), Delphi, Greece,1997, pp.98–112.

[2] H. Gupta,I.S. Mumick,Selection of views to materialize under a maintenance cost constraint. In Proc. 7th Int. Conf. on Database Theory (ICDT'99), Jerusalem, Israel,1999, pp. 453–470.

[3] C.H.Choi, J.X. Yu, and G. Gou, What difference heuristics make: maintenance-cost view-selection revisited. In Proc. 3rd Int. Conf. on Advances in Web-Age Information Management (WAIM'02), Beijing,China,2002,pp.247–258.

[4] D.Theodoratos,T.K.Sellis,Data warehouse configuration. In Proc.23rd Int. Conf. on Very Large Data Bases (VLDB’97), Athens, Greece,1997, pp.126–135.

[5] V.Harinarayan, A.Rajaraman, and J.D.Ullman, Implementing data cubes efficiently. In Proc. the 1996 ACM SIGMOD Int. Conf. on Management of Data, Montreal, Canada, 1996, pp. 205–216.

[6] J.Yang,K. Karlapalem, and Q.Li, Algorithms for materialized view design in data warehousing environment. In Proc.23rd Int. Conf. on Very Large Data Bases(VLDB’97), Athens, Greece,1997,pp.136–145.

[7] C. Zhang, X. Yao, and J.Yang, An evolutionary approach to materialized views selection in a data warehouse environment. IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews. 2001,Vol.31, pp.282–294.

[8] J.T. Horng, Y.J. Chang, and B.J. Liu, Applying evolutionary algorithms to materialized view selection in a data warehouse. Soft Computing.2003, Vol.7, pp.574–581.

[9] W.Y.Lin, I.C.Kuo, A genetic selection algorithm for OLAP data cubes. Knowledge and Information Systems.2004,Vol.6, pp.83–102.

[10] C.Zhang, J.Yang, Genetic algorithm for materialized view selection in data warehouse environment. In Proc. 1st Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK'99),Florence, Italy,1999,pp.116–125.
[11] Z.Michalewicz,Genetic Algorithms+Data Structures=Evolution Programs. New York: Springer-Verlag,1994,pp.28-36.
[12] R.Johnsonbaugh, Discrete Mathematics. New York: Prentice Hall,1993,pp.38-67.
[13] Transaction Processing Performance Council(TPC).TPC Benchmark-D (decision support) proposed revision 1.0.Transaction Processing Performance Council, San Jose,USA,1995.

Ziqiang Wang was born in 1973.He received Ph.D. from Xi'an Jiaotong University in 2005. He is now an associate professor in Henan University of Technology. His main research interests is evolutionary computation.

Dexian Zhang was born in 1961.He received Ph.D. from Huazhong University of Science and Technology in 1995. He is now a professor in Henan University of Technology. His main research interests is machine learning.