WEIGHTED CONSTRAINT SATISFACTION AND GENETIC ALGORITHM TO SOLVE THE VIEW SELECTION PROBLEM

Mohammed El Alaoui[1], Karim El moutaouakil[2] and Mohamed Ettaouil[1]

[1]Modelling and Scientific Computing Laboratory University Sidi Mohammed Ben Abdellah, Fez, MOROCCO
[2]National school of applied sciences Al-Hoceima (ENSAH) BP 03, Ajdir Al-Hoceima

ABSTRACT

A Data warehouse is a tool that is used by big companies, and it gathers data coming from different sources. The main goal of a data warehouse is not only to store data, but also to help companies to make decisions. The huge volume of data makes processing queries complex and time-consuming. In order to solve this problem, the materialization of views is suggested as a solution to improve the processing of the queries. The materialization of views aims to optimize an objective function which is a compromise between the cost of processing queries and the cost of maintenance under a storage space constraint. In this work, we modelled the view selection problem as a weight constraint satisfaction problem. In addition, we use the used the multiple view processing plans (MVPP) Framework as a search space, and we call genetic algorithm to select views to be materialized. According to experimental result, the proposed algorithm has been used to show the quality of the appropriate materialized views selection.

KEYWORDS

Data warehouse, multiple view processing plans, genetic algorithm, weighted constraint satisfaction problem, storage space constraint.

1. INTRODUCTION

Data Warehouse DW can be defined by different ways, but the definition that is most commonly used consists of considering it as being subject-oriented, integrated, non-volatile and time-variant collection of data that supports management's decision of any given entity. It helps store a vast volume of data to be used across the enterprise for decision making. Some intermediate results in the query processing are stored in the DW in order to increase the efficiency of the queries requested to a DW and as well as to avoid accessing the original data sources continuously. These intermediate results are known as materialized views, which help providing faster access to data, thus enhance the query performance. Undoubtedly, materialization of views minimizes query response time. However, materializing all possible views or leaving them all virtual can lead to two extreme and opposite outcomes. In fact, if all views are materialized in a DW, that may give the best performance but at the highest cost of view maintenance. Whereas, if all views are left virtual (i.e. no intermediate result saved in the DW), that will have the lowest view maintenance cost but the poorest query performance. Hence, the optimal choice between the two options is to materialize some of the views while leaving others virtual. At this point, we should pose the following question: what are all views that must be materialized to get the most optimal (or almost optimal) solution? One of the solutions consists of selecting a set of derived views to materialize that simultaneously minimizes the sum of total query response time as well as the maintenance cost of the selected views. This is known as View Selection Problem VSP. The

selection of views to be materialized is one of the crucial decisions in the design of data warehouse. That is in fact NP-hard problem because of the fact that the solution space grows exponentially as the problem size increases [1],[2]. Four search space representations have been proposed, for the selection of views to materialize in data warehouse, such as lattice representations of views[3],[4],[5], [6],[7],[8],[9]. AND-OR view graph representations of views [10],[11],[12]. Multiple View Processing Plan representation [13],[14] and data mining representation [15],[16]. Multidimensional Lattice representation of views is used to express dependencies among views; [3] used a lattice representation to express dependencies among different views of the data cube to handle the problem of view selection for materializing in data warehouse. Using this framework a greedy algorithm is used to pick a set of views with the maximum benefit. [17] proposed Particle Swarm optimization algorithm for materialized cube selection. AND-OR view graphs representations were introduced to represent all the possible ways to generate warehouse views such that the best query path can be utilized to optimize query. [10] have chosen this framework for the selection of materialized views. Multiple views processing plan (MVPP) representation is defined by [18] for the same problem but their works ignored storage costs. [14] Proposed simulated annealing for materialized view selection in data warehousing environment. Data mining representation, this approach is based on detection of common sub-expressions, and represented workload as a binary matrix, in this matrix, each row represents a query and each column is an attribute. This data mining techniques is used for view selection problem. [15] have used data mining techniques are applied to this matrix to obtain a set of candidate views for materializing. In this work, we used MVPP as a search space to obtain an optimal materialized view selection, such that the storage cost is addressed in this work. Multiple views processing plan representation is used to exploit the common sub-expressions that can be detected among the queries. The leaf nodes correspond to the base relations and the root nodes corresponds to warehouse queries. In this step of building MVPP search space we generate the global plan of all queries to find an appropriate set of views a set of views to materialize. The basic idea behind a multi-query graph is to create a single connection graph which represents a set of queries. A multi-query graph facilitates the detection of common sub-expressions among the queries represented by the graph[19]. The selection-view problem can be divided into two phases. The first phase is the identification of common tasks among a set of queries and prepares a small set of alternative plans for each query. The second phase generates a global execution plan that will produce the answers for all the queries when it is executed. In this paper we introduce a new approach for materialized view selection in conjunction with the use of a MVPP. The rest of the paper is organized as follows, In Section 2, we encoding view selection problem into weighted constraint satisfaction problem. In Section 3, we built our search space to solve materialized view selection. Section 4, describes our experimental results, and Section 5, concludes the paper.

2. ENCODING VIEW SELECTION PROBLEM INTO WEIGHTED CONSTRAINT SATISFACTION PROBLEM.

In this part, we propose original weighted constraint satisfaction problems (WCSP) that encode the view selection problem (VSP). WCSP is a constraint satisfaction problem (CSP) in which preferences among solutions can be expressed. In the last few years, the CSP framework has been extended to the soft constraints permits to express preferences among solutions[20]. Soft constraint frameworks associate costs to tuples and the goal is to find a complete assignment with minimum aggregated cost. Costs from different constraints are aggregated with a domain dependent operator. This case is known as the weighted constraint satisfaction problems WCSP.

2.1. Weighted constraint satisfaction problem WCSP

A weighted CSP instance is a quadruplet $P = (X, D, C, W)$ where:

- $X = \{x_1, x_2, \ldots, x_n\}$: is a finite set of n variables,

- D : is a function that maps each variable x_i to the set $D(xi)$ of possible values it can take (domain of x_i),
- $C = \{C_1, C_2, \ldots, C_m\}$: is a finite set of constraints.

Each constraint C_j is a relation over an ordered set $Var(C_j)$ of variables from X , i.e., for $Var(C_j) = \{y_1, y_2, \ldots, y_k\}$, $C_j \subseteq D(y_1) \times D(y_2) \times \ldots \times D(y_k)$.

The elements of the set C_j are called satisfying tuples of C_j and k is called the arity of the constraint C_j.

- $w : \{C_j \mid j \in [1,2,\cdots\cdots,m]\} \rightarrow R^+$: is a function that assigns a positive real value to each constraint C_j of P. $w(C_j)$: is called the weight of constraint C_j.

2.2. Proposed WCSP for the VSP

A number of parameters such as query cost, query frequencies, maintenance cost, frequency of updating, frequency query are used to select an appropriate set of views to be materialized. First, to encode the SVP in terms of a WCSP, we define the following concepts:

- R : is the set of base relations.
- $Q = \{q_1, q_2, \cdots\cdots, q_n\}$: is the set of queries on a given data warehouse;
- f_q : is query frequency associated with the query q;
- M : is the set of intermediate nodes to be materialized whose the elements are obtained by applying the operations: select or join or project or aggregation;
- S : is the cost of storage;
- S_v : is the cost of storage corresponding to view v.
- f_u : is update frequency associated with the view v.

Variables: the variables of the weighted constraint satisfaction problem considered in this paper are given by:

- $Mat(v)$: is a decision variable that equals to 1 if the view v is materialized, 0 else;
- $Cost_q(v)$: is the cost of the query q corresponding to the view v;
- $Cost_m(v)$: is the cost of maintenance when v is materialized.

Domains: the domain of a given variable x , denoted by $D(x)$, is the set of the values that this variable can take.

As the variable $Mat(v)$ is a binary variable, we have $D(Mat(v)) = \{0,1\}$.

The other two variables $Cost_q(v)$ and $Cost_m(v)$ have the two domains: $D(Cost_q(v)) \subset N$ and $D(Cost_m(v)) \subset N$.

Constraints: constraints are divided into two groups:

Binary constraints: The constraint between the variables is presented as follows:

Query cost:

$$Cost_q(v) = \begin{cases} f_q * (\text{the cost of the path from q to v}) & \text{if } Mat(v)=1 \\ f_q * (\text{the cost of the tree of root q}) & \text{otherwise} \end{cases}$$

If v is materialized, the cost query of q is sum cost paths from q to views v. If v is not materialized, the cost query of q is sum cost path from q to base relations.

The query processing cost of an MVPP DAG can be obtained by multiplying query frequency by query processing cost.

The total query cost: $Cost_Q = \sum_{q \in Q} Cost_q(v)$

Maintenance cost:

$$Cost_m(v) = \begin{cases} f_u * (\text{the cost of the path from v to relations R}) & \text{if } Mat(v)=1 \\ 0 & \text{otherwise} \end{cases}$$

If v is not materialized, the cost maintenance is null, else maintenance cost is sum cost from v materialized to base relations.

Materialized view maintenance cost of an MVPP representation can be obtained by multiplying update frequency by maintenance cost.

The total maintenance cost: $Cost_M = \sum_{m \in M} Cost_m(v)$

Unary constraints:

The unary constraint corresponds to the minimization of cost maintenance query. The variable $, Cost_m(v),$ value is minimized by a unary constraint with weight β: $(\text{value of variable maintenace}) \times \beta$ with $0 \le \beta \le 1$.

Minimize the total processing query cost is query processing cost add to maintenance cost. $\text{minimize}(Cost_T = Cost_Q + Cost_M)$ under the constraint that $\sum_{v \in M} S_v \le S$

3. MULTIPLE VIEW PROCESSING PLAN

Multiple views processing plan MVPP is constructed from the workloads queries supported in a data warehouse system. Each node in the MVPP represents a view that could be selected for materialization. The cost metrics for selecting materialized views are based on the following costs: Cost of a query access is the number of rows in the table used to answer this query. Cost of query processing is the frequency of the query multiplied by cost of query access from materialized views. Cost of view maintenance is equal to the cost of constructing the view in response to the changes in the base relation. Total cost is equal to the sum of the cost of query processing and the cost of view maintenance. In this sense, MVPP is a graphic representation of a given set of queries. The root (source) nodes are the queries themselves, the leaf (sink) nodes are the base relations, and all other intermediate nodes are selection, projection, join and aggregation function views that contribute to the construction of a given query in a bottom-up manner.

The view selection process is based on two steeps:

- Construct the best MVPP
- Select views to materialize from the best MVPP

In the first step, MVPP global execution plan is obtained by selection of exactly one plan for each query and identify sub-expressions in which sharing is possible. The selected plans should minimize the cost of global execution plan[21].The second step, so we have to select an appropriate set of materialized view from the best MVPP. The algorithm for generating the MVPP graph is as follows. First, create all alternate execution plans for each query; and then, individual query processing plans are merged for generate MVPP graphs, ultimately the goal is to find the best MVPP, such as the cost of global execution plan is minimized.

Example:

In this part, an application example is provided to illustrate the process of generating multiple views processing plan by considering two queries Q1 and Q2. These queries were selected from Star Schema Benchmark(SSB) [22].

Q1. *select c_nation, s_nation, d_year,*
sum(lo_revenue)
from customer, lineorder, supplier, dates
where lo_custkey = c_customerkey
and lo_suppkey = s_suppkey
and lo_orderdatekey = d_datekey

and c_region = 'ASIA'
and s_region = 'ASIA'
and d_year >= 1992 and d_year <= 1997
group by c_nation, s_nation, d_year ;

Q2. *select c_city, s_city, d_year,*
sum(lo_revenue)
from customer, lineorder, supplier, dates
where lo_custkey = c_customerkey
and lo_suppkey = s_suppkey
and lo_orderdatekey = d_datekey
and c_nation = 'UNITED STATES'
and s_region = 'ASIA'
and d_year >= 1992 and d_year <= 1997
group by c_city, s_city, d_year ;

In Figure 1, the global MVPP is constructed by combining the local query plan the first query with second query. Each query has multiple execution plans. One execution plan is choose for each query to obtain the global MVPP. The best global MVPP is corresponding to best total query costs. In each graph, the query access frequencies are labelled on the top of each query node. And for each node except the root (query node) and leaf (base relation node) nodes, there are two data associated with it. The left stands for the query operator and the right stands for the cost to generate the nodes from base relations.

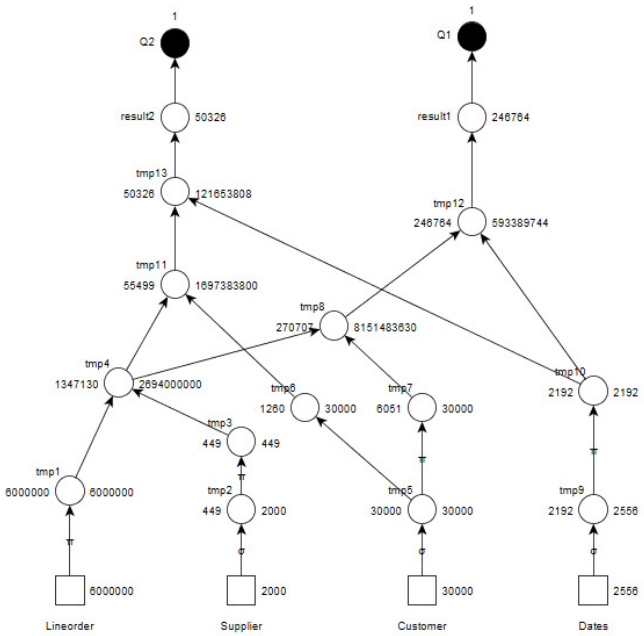

Figure 1: MVPP after all Query Q1-Q2 are merged

The view selection problem, based on framework MVPP, is to choose a set of views to materialize such that the sum cost for query processing and view maintenance is minimal. The sum cost is calculated from the possible combination of nodes. A framework MVPP of n nodes, then we have to try 2^n combinations of nodes to use as a view to materialize. Suppose that there

are some intermediate nodes to be materialized. For each query, the cost of query processing is query frequency multiplied by the cost of query access from the materialized node(s). In Figure 1, if node tmp4 is chosen to be materialized, the query processing cost for Q1 is 1*(246764+ 593389744+ 2192+ 2556+ 8151483630+ 30000+ 30000+ 1347130). The maintenance cost for the materialized view is the cost for the process of updating a materialized view in response to the change in the base relations. The view maintenance cost of temp4 is 2*(2694000000+ 449+ 2000+ 6000000). The goal of solving view selection problem is to find an appropriate of nodes to be materialized in data warehouse.

4. EXPERIMENTAL RESULTS

In this section, we present some computational results. The numerical experiments are carried out using TPC-H database with scale factors of 1GB and the TPC-H queries as our workload. The components of the TPC-H database are defined to consist of eight tables including Region, Nation, Customer, Supplier, Part, Partsupp, Lineitem and Order. TPC-H is the benchmark of the Transaction Processing Performance Council (TPC) for decision support[22]. In order to determine a suitable set of views that minimizes the total cost associated with the materialized views, in conjunction with MVPP framework. In order to evaluate the performance of the proposed algorithm, we compared the result obtained with another algorithm. The proposed algorithm is called genetic algorithm (GA) and the second called hill-climbing (HC) algorithm. The basic idea of the comparison test is to compare the obtained results for materialized views. Based on the analysis of Figure 3, it can be seen, that the result obtained by the proposed approach is better than the HC algorithm.

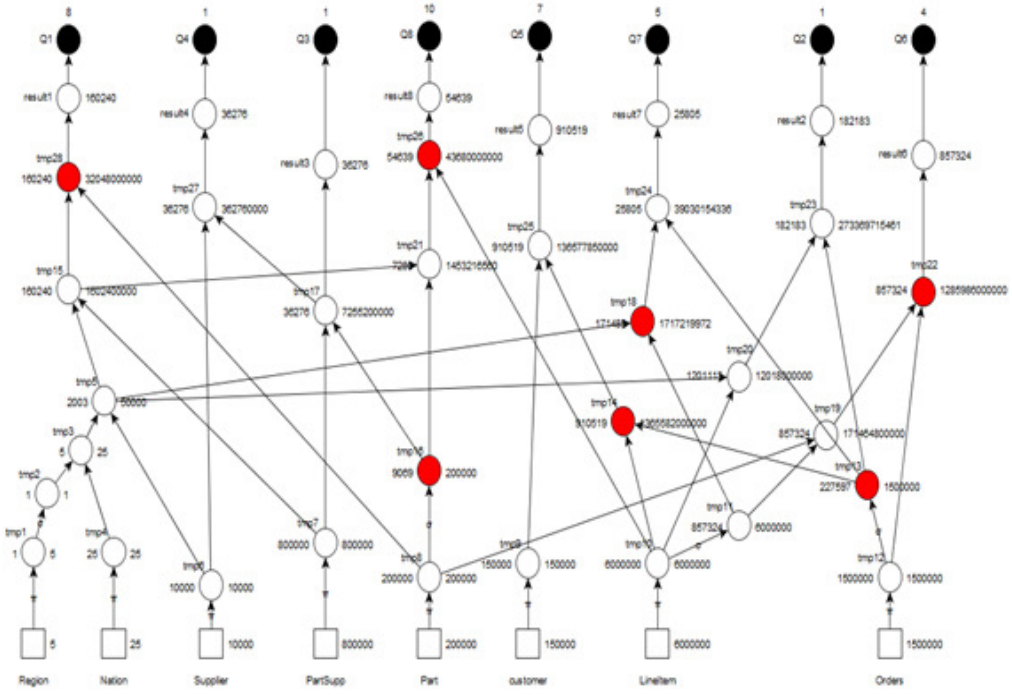

Figure 2 : MVPP graph using TPC-H benchmark data warehouse with materialized views

Table 1: MVPP, cost of query processing, cost of maintenance and total cost

	Cost of query processing	Cost of maintenance	Total cost
All virtual views	17585777978944	0	17585777978944
All materialized views	12014999	10995789073264	10995801088263
Optimal set materialized views obtained by (GA)	1451491204595	7559210580088	9010701784683
Optimal set materialized views obtained by (HC)	1451508620875	7902161180088	9353669800963

The result from Table 1 shows the total cost of optimal set materialized views, obtained by using proposed approach, is better than all virtual views, all materialized views and also of the HC algorithm. The solution of the proposed approach, based on MVPP framework, correspond to materialize the views tmp28, tmp26, tmp16, tmp18, tmp14, tmp22, tmp13 which have appeared in Figure 2. The total cost of the optimal set materialized views refers to the sum of total query processing & maintenance cost of the selected views.

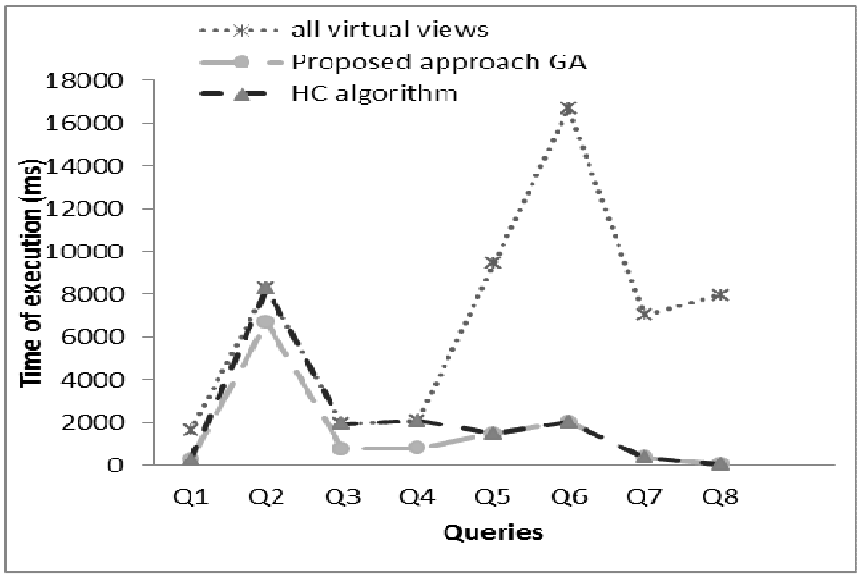

Figure 3: the execution times of the queries (in ms)

In Figure 3, we show how the execution times of the queries workload changes if we consider views materialized. As can be seen, there is a considerable difference in the time of execution for the all queries, mainly due to there are many materialized views that were created to help that the queries. The proposed approach gives better execution time compared with HC algorithm. Finally, something quite important to mention is that all queries, have benefited from materialized views either partially or fully.

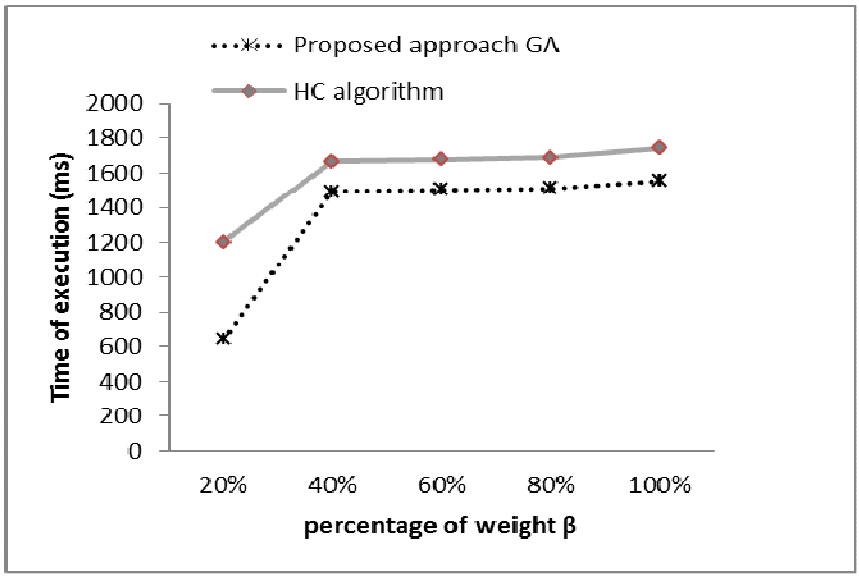

Figure 4: comparison of the average execution time of different weight β.

In Figure 4, shows the average execution time of proposed algorithm is better than HC algorithm on eight queries of different weight β. It is clear from figure 4, that there is a significant decrease in average execution time while changing the value of variable maintenance by weight β, with $\beta = \{20\%, 40\%, 60\%, 80\%, 100\%\}$. As it was expected, the final population produced by our method contains only a feasible solution of the weighted constraint satisfaction problem (proposed in the section 2).

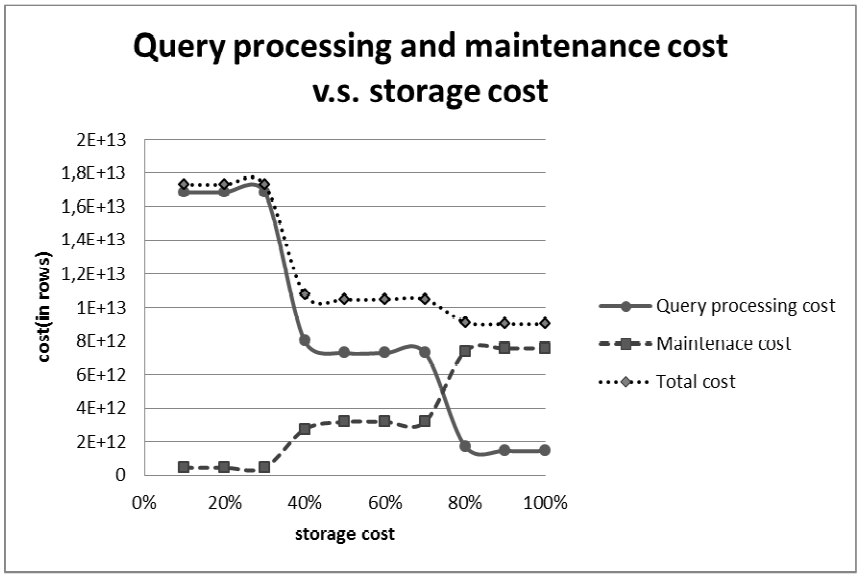

Figure 5: Query processing and maintenance cost vs. storage cost

The objective of the storage cost analysis is to minimize the total cost, which is the sum of two costs: the cost associated with query processing and the cost associated with maintenance. The cost of query processing is the cost to queries that have benefited or not from the materialization

of the views. The maintenance cost is the cost associated with updating the materialized views. The intimate objective of storage cost analysis is to find a compromise between the cost associated with the query processing and the maintenance cost. Note that when the cost of storage increases, the total cost decreases. Figure 5, illustrates this concept well.

5. CONCLUSION

The selection of views to materialize is one of the most discussed topics in data warehouse. View selection problem (VSP) can be solved by the construction of a multiple view processing plan (MVPP) search space based on multiple query optimizations (MQO) technique. In this work, we have extended the usage of VSP in conjunction with the MVPP framework. The first extension is to model the VSP as WCSP. A second extension is to incorporate storage cost into the objective function to select optimal set of views to materialization. In order to find most optimized solution, the proposed algorithm is better in term of selection of suitable set of materialized views as compared to HC algorithm. In future work, we will extend this work by applying the neural network approach and weighted constraint satisfaction problem to solve the view selection problem.

REFERENCES

[1] H. Gupta And I. S. Mumick, "Selection Of Views To Materialize Under A Maintenance Cost Constraint," Proc. 7th Int. Conf. Database Theory, Vol. 13, Pp. 453–470, 1999.

[2] H. Gupta And I. S. Mumick, "Selection Of Views To Materialize In A Data Warehouse," Ieee Trans. Knowl. Data Eng., Vol. 17, No. 1, Pp. 24–43, 2005.

[3] V.Harinarayan, "Implementing Data Cubes Efficiently," Proc.Acm Sigmod Int. Conf. Manag. Data, Pp. 205–216, 1996.

[4] D. Yang, M. Huang, And M. Hung, "Efficient Utilization Of Materialized Views In A Data Warehouse," Pakdd 2002 Adv. Knowl. Discov. Data Min., Pp. 393–404, 2002.

[5] G. Gou, J. X. Yu, And H. Lu, "A* Search: An Efficient And Flexible Approach To Materialized View Selection," Ieee Trans. Syst. Man Cybern. Part C Appl. Rev., Vol. 36, No. 3, Pp. 411–425, 2006.

[6] T. V. V. Kumar And S. Kumar, "Materialized View Selection Using Simulat- Ed Annealing," Int. Conf. Big Data Anal., Pp. 168–179, 2012.

[7] C. S. Park, M. H. Kim, And Y. J. Lee, "Finding An Efficient Rewriting Of Olap Queries Using Materialized Views In Data Warehouses," Decis. Support Syst., Vol. 32, No. 4, Pp. 379–399, 2002.

[8] J. Chang And S. Lee, "Extended Conditions For Answering An Aggregate Query Using Materialized Views," Inf. Process. Lett., Vol. 72, Pp. 205–212, 1999.

[9] E. Soutyrina And F. Fotouhi, "Optimal View Selection For Multi Dimensi- Onal Database Systems," Ideas, Pp. 309–318, 1997.

[10] I. Mami, R. Coletta, And Z. Bellahsene, "Modeling View Selection As A Constraint Satisfaction Problem," Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), Vol. 6861 Lncs, No. Part 2, Pp. 396–410, 2011.

[11] D. Theodoratos, "Detecting Redundant Materialized Views In Data Warehouse Evolution," Inf. Syst., Vol. 26, No. 5, Pp. 363–381, 2001.

[12] T.V.V.Kumar And S.Kumar, "Materialised View Selection Using Differ- Ential Evolution," Int. J. Innov. Comput. Appl., Vol. 6, No. 2, Pp. 102–113, 2014.

[13] J. Yang, K. Karlapalem, And Q. Li, "Algorithms For Materialized View Design In Data Warehousing Environment," Vldb, No. 1, Pp. 136–145, 1997.

[14] R. Derakhshan And F. Dehne, "Simulated Annealing For Materialized View Selection In Data Warehousing Environment.," 24th Iasted Int. Conf. Database Appl., Pp. 89–94, 2006.

[15] K. Aouiche And J. Darmont, "Data Mining-Based Materialized View And Index Selection In Data Warehouses," J. Intell. Inf. Syst., Vol. 33, No. 1, Pp. 65–93, 2009.

[16] K. Aouiche, P.-E. Jouve, And J. Darmont, "Clustering-Based Materialized View Selection In Data Warehouses," Lect. Notes Comput. Sci. Incl. Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinforma., Vol. 4152 Lncs, No. 1, Pp. 81–95, 2007.

[17] A. Gosain And Heena, "Materialized Cube Selection Using Particle Swarm Optimization Algorithm," Procedia Comput. Sci., Vol. 79, Pp. 2–7, 2016.

[18] J. Yang, K. Karlapalem, And Q. Li, "A Framework For Designing Material- Ized Views In Data Warehousing Environment," Icdcs'97, Pp. 458–465, 1997.

[19] T. K. Sellis And S. Ghosh, "On The Multiple-Query Optimization Problem," {Ieee} Trans. Knowl. Data Eng., Vol. 2, No. 2, Pp. 262–266, 1990.

[20] K. Haddouch, K. Elmoutaoukil, And M. Ettaouil, "Solving The Weighted Constraint Satisfaction Problems Via The Neural Network Approach," Int. J. Interact. Multimed. Artif. Intell., Vol. 4, No. 1, P. 56, 2016.

[21] S. K. Mishra And S. Pattnaik, "Evaluation Of Gene Value And Heuristic Function Of Alternate Plans In Multi Database System Using Genetic Algorithm," Int. J. Database Manag. Syst., Vol. 3, No. 3, Pp. 78–86, 2011.

[22] P. O. Neil, B. O. Neil, And X. Chen, "Star Schema Benchmark - Revision 3," Tech. Rep., 2009.