Approaches and Issues in View Selection for Materializing in Data Warehouse

Rajib Goswami, D.K Bhattacharyya, Malayananda Dutta

Department of Computer Science and Engineering, Tezpur University, Tezpur, India
E-mail: {rgos, dkb, malay}@tezu.ernet.in

Jugal K. Kalita

Department of Computer Science, University of Colorado Colorado Springs, Colorado Springs, CO USA
E-mail: kalita@eas.uccs.edu

Abstract: To facilitate efficient query processing in data warehouses and develop On Line Analytical Processing (OLAP) for decision support systems, intermediate data derived in the middle of complex processing may be stored as a set of materialized views. It is not possible to save each and every intermediate query result due to limitation of space and updating cost. Therefore, an optimum set of views need to be selected for materialization and, this requires a good optimization technique. Many approaches have been presented in the literature to achieve good solutions to this problem. In this paper we attempt to provide a comprehensive survey of the approaches and algorithms introduced to address the issue. We also attempt to identify the key issues and research challenges in this area.

Keywords: Data Warehouse; View selection; View materializing; Query response cost; Materialized view maintenance; OLAP; HRU Algorithm; Randomized Algorithm; AND-OR view graph; Optimal query plan.


Biographical notes: Rajib Goswami is currently working as Systems Analyst at Tezpur University, Tezpur, India. He did his post-graduation from Assam Engineering College, Guwahati University, Guwahati, India, in 1993. He has been involved in development projects of various financial applications, Decision Support Systems, Data Warehousing and ERP solutions.

Copyright © 2009 Inderscience Enterprises Ltd.
1 Introduction

In a data warehouse, it is necessary to perform OLAP operations fast to find different aggregations for decision making. Therefore, instead of computing aggregations on-the-fly every time, it is efficient if appropriate high level views are created and saved in the database as materialized views. To answer queries, the materialized views may be used without recomposing the views. On the one hand materializing every view requires a large amount of memory, and on the other hand, not materializing any view requires lots of redundant on-the-fly computations. Business excels by efficient quality driven data warehousing technologies (Smith 2011). Applications, technology and business users are constantly changing, and due to this there is a need of continuous updating or modification in database technologies (Dahanayake & Thalheim 2013) and thereby materialized view involves maintenance cost. Thus, there are trade-offs to be made.

In the context of conventional database management systems, a view is defined as a derived relation on top of some base relations. A view defines a function from a set of base tables to a derived table. In data warehousing, historical data are kept in terms of facts and dimension tables whereas aggregated values are kept in schemas like Star, Snowflake and Fact Constellation (Chaudhuri & Dayal 1997). The derived relations from the base tables for responding to decision support queries are called data warehouse views. Other than utilizing materialized views for making query response cycles of database applications faster, there are some other applications of materialized view as well, such as replication servers,
chronicles or data recording systems, data visualization, and mobile systems (Gupta & Mumick 1995). As we have entered an era of Big Data, and presently Big Data warehousing tool like Hive does not support materialized views, there is scope of more efficient query processing in Big Data environment by view materialization (Philip Chen & Zhang 2014). On massive collections of data, integrity checking is a critical problem (Feras Hanandeh & Alsmadi 2012). Materialized views which have already passed efficiency and integrity tests save a large amount of costs involved in these checks.

1.1 Selection of views for materializing in data warehouse

View materialization for reducing query processing costs in data warehouse applications requires a large amount of space. Materialized views are updated or maintained in response to changes in the base data. Therefore, it is necessary to select an appropriate set of views to materialize to increase performance, with optimized query processing and view maintenance costs. This is known as the materialized view selection problem (Harinarayan et al. 1996, Gupta et al. 1997, Gupta & Mumick 1999). Thus, the materialized view selection problem entails the following: Given a set of data warehouse queries, select a set of views to materialize so that the total query processing cost and view maintenance cost is minimized. Formally the problem may be defined as Definition 1.

**Definition 1:** Given a set of \( n \) frequent queries \( Q = \{q_1, q_2, \ldots, q_n\} \) on a data warehouse, and the set of \( m \) views \( V = \{v_1, v_2, \ldots, v_m\} \) generated while responding the queries \( Q \), a set of views \( V' \subseteq V \) are to be selected for materializing, such that, if \( \text{space}(V') \) is the space requirement for materializing \( V' \), \( C(Q, V') \) is the total cost of responding queries \( Q \) when \( V' \) is materialized and \( U(V') \) is the maintenance cost of materialized views \( V' \), then the selection \( V' \) optimizes \( C(Q, V'), U(V') \) and \( \text{space}(V') \).

Research on this problem started in the early nineties when several heuristic greedy algorithms were proposed (Harinarayan et al. 1996, Gupta et al. 1997, Gupta & Mumick 1999, Nadeua & Teorey 2002, Serna-Encinas & Hoya-Montano 2007). It has been observed that when the dimension of the data warehouse grows, the solution space grows exponentially and therefore it becomes NP-hard problem (Gupta & Mumick 1995, Harinarayan et al. 1996, Gupta et al. 1997, Gupta & Mumick 1999, Gupta 1999) and as a result, various stochastic, evolutionary, data mining and clustering based optimizing approaches have been proposed with different data structures and notions (as presented by a pie chart in Figure 1) to handle this problem. In this paper, we present different approaches developed so far to handle this problem and analyse issues involved.

1.2 Existing approaches

Several heuristic greedy approaches have been proposed by defining different cost and benefit parameters to deal with the view selection for materializing problem (Harinarayan et al. 1996, Gupta et al. 1997, Gupta & Mumick 1999, Nadeua & Teorey 2002, Serna-Encinas & Hoya-Montano 2007). Most of these approaches use multidimensional lattice structures to select views for materialization, based on the original greedy algorithm proposed by Harinarayan et al. (1996), and popularly referred to as the HRU-Greedy algorithm. In Gupta & Mumick (1999) and Gupta (1999), a competitive heuristic for selection of views to optimize total query response time is proposed using the notion of an AND-OR view graph given as an input. Vijay Kumar (2013) proposes a query based view
R. Goswami, D.K. Bhattacharyya, M. Dutta and J.K. Kalita

**Figure 1:** Research shares in data warehouse view selection for materialization

A selection approach (AQVSA) considering both the size and the query frequency of each view to greedily select top-\(k\) views for materialization. In Yang et al. (1997) a framework for analysing the issues in selecting views to materialize for achieving the best combination of good query performance and low view maintenance is proposed using a global query access plan by merging local access plans for individual queries based on shared operations on common data sets. This framework is known as the Multiple View Processing Plan (MVPP). Most of the recent approaches on view selection for materialization problem are based on exploitation of recent statistical data access in data warehouse (Derrar & Boussaid 2013). Yang et al. (1997) presented a heuristic algorithm for the materialized view design problem using MVPP. Some randomized algorithm based approaches have also been developed using MVPP (Derakhshan et al. 2006, 2008, Zhang & Yang 1999, Zhang et al. 2001). In Vijay Kumar & Devi (2012), an algorithm to construct a single materialized view by a heuristic that maximally merges previously posed optimal user queries on the data warehouse is proposed.

As noted earlier, the view selection problem is NP-hard, and therefore, most recent approaches use randomized algorithms such as simulated annealing (SA), parallel simulated annealing (PSA), Genetic Algorithms (GA) and Memetic Algorithms (MA), and Particle Swarm Optimization (PSO). Most of these approaches use AND-OR view graphs generated from the query workload as input or MVPP graphs. Wagner & Agrawal (2013) designed an Evolutionary Algorithm for view selection problem, by considering the amount and importance of data retrieved by data warehouse queries. Data mining techniques also have been used effectively on workloads (sets of queries) representative of data warehouse usages in order to deduce quasi-optimal configurations of materialized views and/or indexes (Aouiche et al. 2006, Aouiche & Darmont 2009, Das & Bhattacharyya 2005, Kumar et al. 2012).

### 1.3 Research objective

A major challenge to handle the view selection problem for materialization in data warehouses is to reduce the complexity of the view selection algorithms and to improve...
scalability. The objective of this paper is to analyse various approaches proposed to address the NP-hard problem of materialized view selection problem in data warehousing, by introducing respective data representations, and identifying various research challenges and associated issues.

1.4 Organization of the paper

In Section 2, a detailed review of literature surveyed on techniques proposed for selecting views to materialize in data warehouses are presented. Different solutions on this problem with their validations and limitations are presented in Section 3. Section 4 presents a discussion on performances of solution models suggested so far. In Section 5, concluding remarks about contribution from this survey, limitations of the study, implications to theory and practices on different techniques, and future research directions are presented.

2 Literature Review

Based on our survey of literature on selecting views for materializing in data warehouses, it has been observed that the distribution of research activities on this problem are going on (during the period 1997 till 2014) as illustrated in Figure 1. The surveyed literature on different representations and approaches are reviewed in following sub-sections.

2.1 Heuristic approach by Multidimensional Lattice representation of views

Typically, data in data warehouses are conceptualized as multi-dimensional data cubes where each cell of the data cube is a view consisting of an aggregation of interest (Harinarayan et al. 1996). Early approaches to the view selection problem for materializing investigated the issue of which cells of the data cube are to be materialized when it is too costly to save all the cells or views. Harinarayan et al. (1996) used a lattice framework to express dependencies among different cells or views of the data cube to handle this problem. This is pioneering work in the view selection for materializing problem. They use a multidimensional lattice representation consisting of nodes representing the possible views that may be candidates for materializing, and edges representing dependencies between the connected views (Harinarayan et al. 1996, Nadeua & Teorey 2002, Mohania et al. 1999). Each node of the lattice structure represents a view labelled with the set of dimensions of the GROUP-BY list for the respective view with the number of rows in the view. Thus lattices are the hyper cubes, in which the views are vertices of an n-dimensional cube for some n. An example of lattice structure is shown in Figure 2, where label on the top node, \{c1, c2, c3\} 6M, means GROUP-BY is used for c1, c2 and c3 and it returns 6 million rows.

A multidimensional lattice consists of nodes, depicting the possible views that can be materialized, and edges representing dependencies among these views. The greedy algorithm popularly known as the HRU-Greedy algorithm (Harinarayan et al. 1996) calculates the benefit of each possible view in successive iterations and selects the view which is most beneficial for materialization and adds it to the set of selected views. This process is continued till a pre-specified number (k) of materialized views have been selected and added to the list. To compute benefits, a cost model must be defined. The linear cost model defined in HRU-Greedy is, \( T = m \times S + C \), where \( T \) is the running time of the
query on a view of size $S$. $C$ gives the fixed cost, i.e., the overhead of running this query on a view of negligible size and $m$ is the ratio of the query time to the size of the view, after accounting for the fixed cost.

The advantage of this representation and technique is that the most beneficial views can be found directly from the base relations or schema of the data warehouse without considering query log files and query access frequency. However, the basic disadvantage of the lattice representation is that the number of nodes in the lattice structure grows exponentially with the dimension of the data warehouse. Since only query-response generation cost and space cost are considered for optimizing the selection of views for materializing without considering query frequency and view maintenance cost, this data structure is not applicable for frequent query access and frequent base table updating.

2.2 **AND-OR view graph representation of queries and views**

In Gupta & Mumick (1999) and Gupta (1999), a graph termed as AND-OR view graph is suggested as one of the inputs to the view selection problem. The queries and views are expressed in terms of **directed acyclic graphs** (DAGs). The AND-OR view graph is defined using the concepts of AND arcs and ANDOR-DAGs.

An AND-DAG for a query or a view is a directed acyclic graph having the base relations as ‘sinks’ with no outgoing edges and the view (node) $V$ as a ‘source’ with no incoming edge. If a node or view $u$ has outgoing edges to nodes $v_1, v_2, \ldots, v_k$, all of the views $v_1, v_2, \ldots, v_k$ are required to compute the cost of $u$. This dependence is indicated by drawing a semicircle, called an AND arc, through the edges $(u, v_1), (u, v_2), \ldots, (u, v_k)$. Such an AND arc has an operator and a cost associated with it, which is the cost incurred during the computation of $u$ from $v_1, v_2, \ldots, v_k$.

An ANDOR-DAG for a view or a query $V$ is a DAG with $V$ as a source and the base relations as sinks. Each non-sink node has associated with it one or more AND arcs. The Definition 2 provides a formal way of defining the AND-OR View Graph.

**Definition 2:** A DAG $G$, with base relations as the sink is called an AND-OR view graph for a set of views and query responses $V_1, V_2, \ldots, V_k$, if for each $V_i$, there is a sub-graph $G_i$ in $G$ that is an expression ANDOR-DAG for $V_i$. Each node $V$ in an AND-OR view graph has the following parameters associated with it: space $S_v$, query frequency $f_v$ (frequency of the queries on $V$), update-frequency $g_v$ (frequency of updates on $V$), and reading cost $R_v$ (cost incurred in reading the materialized view $V$).
The view selection for materialization problem using AND-OR View graph is defined as Definition 3.

**Definition 3:** Given an AND-OR view graph $G$ and a quantity $S$ (available space), the view-selection problem is to select a set of views $M$ which constitute a subset of the nodes in $G$, that minimizes the total query response time, under the constraint that the total space occupied by $M$ is less than $S$ under a maintenance-cost constraint.

In Gupta & Mumick (2005) a heuristic model based on this representation was used to handle view selection problem and found that a fairly close optimal solution was obtained. Stochastic, evolutionary and other bio-inspired algorithm based models are presented on this problem in Zhang & Yang (1999), Zhang et al. (2001), Lee & Hammer (2001), Qingzhou et al. (2009) and Sun & Wang (2009) using AND-OR graph representation of views.

This representation is widely used for the general problem of selection of views in a data warehouse. The AND-OR view graph represents the general data warehouse scenario in an easily understandable manner for analysing the queries and their component views. Therefore, it is suitable for computing the cost of answering queries (using the sets of materialized views in the view graph) and the maintenance cost. Each query and its attached views and base tables are considered individually and therefore, sharing of materialized views by multiple queries is ignored.

### 2.3 Optimal query plan based graphical representation in heuristic as well as randomized algorithmic models

Another approach used in view selection for materializing in data warehouses uses a directed acyclic graph (DAG) representing all frequently asked queries or a specific number of queries by a query processing strategy of warehouse views (Yang et al. 1997). Here, the leaf nodes correspond to the base relations in the member databases and the root nodes correspond to warehouse queries. The graph is called a Multiple View Processing Plan (MVPP). Analogous to a query execution plan, different MVPPs for the same view specification may be appropriate under different query update characteristics of the applications. The idea is that for different types of analysis, a data warehouse may contain multiple views that are shared by a number of queries. Therefore, it may be more efficient not to materialize all of the views, but to materialize certain commonly shared views or portions of the base data, from which the warehouse views can be derived.

An example MVPP graph is illustrated here by five base relations: Employee(ecode, name, deptid), Dept(deptid, name, location), Paybill(ecode, account_head_code, amount), Account_head(account_head_code, details), Transaction(tid, narration, ecode, date) and by following four (SQL) queries and an MVPP graph in the Figure 3.

- **Query 1:**
  ```sql
  SQL> select employee.name from employee, 
            dept where dept.location='Tezpur' and 
            employee.deptid=dept.deptid;
  ```

- **Query 2:**
  ```sql
  SQL> select transaction.narration
  ```
from employee, transaction, dept
where dept.location='Tezpur' and
employee.deptid=dept.deptid and
transaction.ecode=employee.ecode;

• Query 3:

SQL> select account_head.details,
employee.name, paybill.amount from
employee, dept, paybill,
account_head where
department.location='Tezpur'
and employee.deptid=dept.deptid
and employee.ecode=paybill.ecode
and paybill.account_head_code=
account_head.account_head_code
and paybill.amount>40000;

• Query 4:

SQL> select account_head.details,
paybill.amount
from paybill, account_head where
paybill.amount>40000
and paybill.account_head_code
=account_head.account_head_code;

The number of rows in each view is given by the side of each node or view in the MVPP graph depicted in Figure 3. For example, the node ‘Result 1’ in the MVPP graph means, it has 35.35 thousand rows. The unit ‘k’ denotes a thousand and ‘m’ denotes million. Query frequencies are marked on top of each query. In Figure 3, the query frequency of query 1 is 10, query 2 is 0.5 and so on.

The problem for materialized view design in terms of MVPP can be described as: If $V$ is the set of vertices in an MVPP, and for $\forall v \in V$, $R(v)$ is the result relation generated by corresponding vertex $v$, then to determine a set of vertices in $V$, such that $\forall v \in V, R(v)$ is materialized, the cost of query processing and view maintaining is minimal.

Yang et al. (1997) designed a heuristic algorithm to select views for materializing by using MVPP DAG. Derakhshan et al. (2006, 2008) applied Simulated Annealing algorithm using this representation in view selection for materializing problem. In Goswami et al. (2012), MVPP DAG representation is used for defining the problem as multi-objective optimization problem and applied Multi-Objective Simulated Annealing techniques.

The MVPP representation is suitable for depicting relationships among queries to the base relations through intermediate and shared temporary views. From the MVPP graph, the size of intermediate views can be found or computed easily and provided as input to the view selection for materialization algorithm. But the cost involved in generation of an MVPP graph from the query workload of a data warehouse is high when the query processing plan changes and input workload is very large.
2.4 Data mining based techniques in view materialization problem

This approach is based on detection of common sub-expressions within workload queries and finding the underlying views (Aouiche et al. 2006, Aouiche & Darmont 2009, Das & Bhattacharyya 2005, Rizzi & Saltarelli 2003, Kumar et al. 2012). A workload is syntactically analysed to enumerate relevant candidate views. The warehouse’s transaction logs are first analysed over a certain time period and the most appropriate workload is considered for anticipating future workload of the system by the warehouse administrator. In Aouiche et al. (2006), Aouiche & Darmont (2009), Kumar et al. (2012), all the queries and the attributes in them are identified and then by analysing the workload queries and their sub expressions, a query vs. attribute binary matrix is formed. In this matrix, each row represents a query and each column is an attribute. A cell is marked as one if a particular attribute is present in a particular query, and zero otherwise. Data mining techniques are applied to this matrix to obtain a set of candidate views for materializing.

The query vs. attribute binary matrix is well exploited by data mining techniques to obtain a candidate set of views and indexes for materializing (Aouiche et al. 2006, Aouiche & Darmont 2009). Although the matrix representation is easy to implement and directly usable by data mining and clustering algorithms, the main difficulty is syntactic analysis of the query workload. This is because scanning through numerous sub-queries and intermediate results for generating the binary matrix requires a fool proof algorithm.

2.5 Discussion

The pioneering view selection for materialization algorithms such as the HRU-Greedy algorithm and PGA (Harinarayan et al. 1996, Nadeua & Teorey 2002) use the lattice representation of views in data warehouses. Though this representation is suitable and easy to implement in low dimensional deterministic cases, the main disadvantage of this representation is that the number of nodes in the lattice structure is exponential relative to the number of dimensions. The AND-OR view graph and the MVPP representation are
Table 1: Representations used in view selection algorithms and associated issues

<table>
<thead>
<tr>
<th>Notions</th>
<th>View selection algorithms used</th>
<th>Associated issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lattices</td>
<td>HRU-Greedy, PGA</td>
<td>Exponential growth with dimension of data warehouse. Only query-response generation cost and space cost are considered, query frequencies and view maintenance frequencies are not considered.</td>
</tr>
<tr>
<td>AND-OR graphs</td>
<td>Heuristic, GA, MA, PSO</td>
<td>Plan for multiple query processing is not considered and therefore sharing of materialized views by multiple queries are ignored.</td>
</tr>
<tr>
<td>MVPP graphs</td>
<td>Heuristic algorithm, Simulated Annealing(SA), Parallel Simulated Annealing(PSA)</td>
<td>Cost of building view graphs when the query processing plan changes and input workload is large.</td>
</tr>
<tr>
<td>Wavelet structure-Dwarf</td>
<td>Heuristic-greedy algorithm by physical re-designing of Data warehouse</td>
<td>To change physical design of data warehouse.</td>
</tr>
<tr>
<td>Query vs. attribute binary matrix</td>
<td>Data mining and clustering</td>
<td>Requirement of scanning through numerous sub-queries and intermediary results.</td>
</tr>
</tbody>
</table>

mostly used in randomized algorithms for view selection for materialization. However, the graph generation process becomes costly for complex and huge query workloads. The matrix representation of view attributes and base relations is directly usable by data mining and clustering algorithms. However the need of syntactic analysis of large query workload is an issue to be handled.

Other approaches such as wavelet framework (Smith et al. 2004) represent multidimensional data cubes by decomposing the cubes into an indexed hierarchy of wavelet view elements that correspond to partial aggregations of data cubes. Although keeping aggregated values in data warehouses is in the spirit of view materialization, it is all about changing the physical design of the data cubes. Similarly, Sismanis et al. (2002) propose a concept called dwarf structure to compress data cubes which impacts on the physical design of data warehouses.

Almost all the approaches we have seen, analyse queries to find sub-expressions inside to find components or intermediate views that may be beneficial if materialized. Semantic analysis of sub-expressions is used either to generate some kind of graphs or to generate matrices which are used as input to the view selection algorithm for materialization. Scanning through numerous intermediate results is very costly and these methods are not scalable with respect to the number of queries (Aouiche & Darment 2009). The various data structures and concepts used in different view selection algorithms and associated issues are presented in Table 1.
3 Theoretical Models for Selecting Views to Materialize in Data Warehouse

In following sub-sections, various models and algorithms used for selecting views to materialize in data warehouses are presented with their validations and limitations.

3.1 Models using heuristic approaches

Most heuristic approaches are descendants of the view selection algorithm for materializing in data warehousing called the HRU-Greedy algorithm (Harinarayan et al. 1996). It searches the hypercube lattice structure to select an optimum set of views in terms of space utilization and the number of views. The algorithm suffers from the problem of exponential explosion with dimensionality. Therefore, Nadeua & Teorey (2002) propose an algorithm called the Polynomial Greedy Algorithm, PGA, for a scalable solution. The execution time for the PGA algorithm is lower than that for the HRU algorithm theoretically as well as experimentally, though the trend is the same. In Gupta & Mumick (1995, 1999), Gupta (1999), Gupta & Mumick (2005) a greedy algorithm framework for the view selection problem using the AND-OR view graph is used. Yang et al. (1997) presents a heuristic algorithm which can provide a feasible solution based on individual optimal query plans. In Vijay Kumar (2013), a query based view selection approach is proposed considering both the size and the query frequency of each view to greedily select the top-\(k\) views for materialization.

3.1.1 The HRU algorithm

To solve the optimization problem, the HRU greedy algorithm, first tries to minimize the average time taken to derive views under the constraint of materializing a fixed number of views (Harinarayan et al. 1996). It uses the hypercube lattice notion to represent the various views or GROUP-BY statements in queries as discussed in Section 2. Suppose we have a data cube lattice with known associated space cost for each view. Let \(C(v)\) be the cost of view \(v\). Let us assume that we can select a maximum of \(k\) views in addition to the top-view. If a view \(w\) can be answered by \(v\), it is said that the view \(w\) is covered by \(v\). For each view \(w\) that \(v\) covers, this algorithm compares the cost of answering \(w\) using \(v\) and using another view from \(S\) which is the cheapest so far for answering or deriving \(w\). If the cost of \(v\) is less than the cost of its competitor, the difference is part of the benefit of selecting \(v\) as a materialized view. The total benefit is the summation of benefits over all views. The HRU Greedy algorithm for selecting \(k\) views to materialize is given in Algorithm 1 where \(B(v, S)\) is the total benefit using \(v\) to evaluate \(w\). In HRU-Greedy, the number of views to be materialized is first fixed. This number of views to be materialized is the number of

---

**Algorithm 1** The HRU Greedy algorithm

**Require:** \(k\) number of candidate views \(v_1, v_2, \ldots, v_k\) and the top-view

**Ensure:** The selected set of views \(S\) for materializing

1: \(S \leftarrow \{\text{top-view}\}\)
2: for \(i = 1\) to \(k\) do
3: \(S \leftarrow S \cup \{v_i\}\)
4: end for
5: return \(S\)
iterations of the algorithm. In different iterations, each node or view other than the top-view is evaluated in terms of benefits (if it is materialized) and the highest benefit node or view is selected for that iteration.

Validation: Let us consider Figure 2. If \{c2, c3\} is selected, the total benefit will be \((6 - 0.8)M \times 4 = 20.8M\), because 4 nodes or views, viz., \{c2, c3\}, \{c3\}, \{c2\} and \{\} are dependent on it. Similarly for \{c3\}, the benefit is \(5.99M \times 2\). Thus, we compute the benefit for each node and the most beneficial node is added to the list of selected views for materialization. Then again in the next iteration, the whole process is repeated assuming that one view is already materialized. In the first iteration if \{c2, c3\} is selected, then in next iteration, the benefit of \{c3\} will be \((0.8 - 0.01)M \times 2 = 0.79M \times 2\) and the benefit of \{c1\} is \((6 - 0.1)M \times 2\). Thus after computing the benefits of all the remaining nodes, the most beneficial node is selected for materialization in this iteration. The process continues for the fixed number of iterations and in each of the iterations one view is selected and added to the list of views that are to be selected for materializing.

In each of the iterations, the algorithm evaluates every unselected node, and in each evaluation, it considers the effect on every descendant. Thus we find that, if \(k\) views are to be selected and there are a total of \(n\) nodes in the lattice structure, the complexity of this algorithm is \(O(kn^2)\). If \(d\) is the number of dimensions in the data cube, the number of nodes in the lattice structure equals to \(2^d\), i.e., \(n = 2^d\). Therefore, complexity becomes \(O(n^2) = O(2^{2d})\). Thus the algorithm results in exponential bursts when number of dimensions is high. The HRU-Greedy algorithm needs to know the size of each of the views beforehand and then it computes the benefit of each and every view if materialized, and selects the most beneficial ones for materializing. Therefore, the quality of the views selected for materialization is good.

Limitations: The main problem with this technique is that the algorithm results in exponential bursts when number of dimensions grows. It also does not take into account query access frequency and view maintenance cost due to updating of base tables.

3.1.2 The Polynomial Greedy Algorithm (PGA) for materialized view selection

In PGA model (Nadeua & Teorey 2002), each iteration of the HRU-Greedy algorithm is divided into a nomination phase and a selection phase to tame the exponential explosion of HRU-Greedy algorithm. From the top-view, it first selects the most beneficial node in the lattice structure of views which is connected to the top view. This node is added to the list of nominations. Then from this nominated node, it selects the most beneficial node from the next layer of connected nodes. This is again added to the list of nominated nodes. The process goes on till it traverses to the bottom. Out of this set of nominated nodes, the most benefitting node is selected for materializing and put into the list of selected views. Again in the second iteration, from the top node, out of all nodes connected to the top view but not already nominated, the most beneficial node is selected for inclusion in the nomination list. From this node, the most beneficial node from the connected nodes is selected for adding to the nomination list and so on. From this second list of nominations, the most beneficial node is selected and added to the list of selected views for materializing. This process continues for some iterations and a list of views or nodes from the lattice is selected for materializing.

Validation: To overcome the problem of evaluating an exponential number of nodes, as in the case of HRU-Greedy algorithm, it considers only the promising nodes of the lattice and thereby the PGA model controls the complexity of the HRU model.
Limitations: Though the PGA model can control the complexity of the HRU-Greedy algorithm, the HRU algorithm is better than the PGA algorithm in terms of the quality of the views selected for materialization (Nadeua & Teorey 2002).

3.1.3 AND-OR View Graph based greedy algorithm

Gupta & Mumick (1999) and Gupta & Mumick (2005) present a heuristic greedy algorithm using AND-OR view graph to optimize selection of views for materializing considering the total query response time under constraints of disk-space and view maintenance costs. An AND-OR view graph for a set of queries can be represented by integrating or merging ANDOR-DAGs. The nodes in the final AND-OR view graph represent a candidate view for materialization. Two other parameters are also used to compute the cost of views. They are query frequencies \( f_v \) of views of the query workload of the data warehouse, and update frequencies \( g_v \), which is the sum of the updating frequencies of all the base relations used for derivation of the view. For an AND-OR view graph \( G \), the view selection problem is to select a set of views \( M \), which is a subset of the nodes in \( G \), that minimizes the total query response time and maintenance cost of \( M \) under the constraint that the total space occupied by \( M \) is less than \( S \). It is formally explained below.

Let \( Q(v, M) \) denote the cost of answering the query \( v \) using the set of materialized views \( M \) in the view graph \( G \) and \( UC(v, M) \) be the maintenance cost for the view \( v \) when the set of views \( M \) is materialized. Given an AND-OR view graph \( G \) for queries \( q_1, q_2, \ldots, q_k \) and a quantity \( S \), the view selection problem is to select a set of views or nodes \( M = \{v_1, v_2, \ldots, v_m\} \), that minimizes \( \tau(G, M) \) in Equation 1, where under the constraint \( \sum_{v \in M} S_v \leq S \), \( S_v \) is the space occupied by the view \( v \), \( f_q \) is query frequency and \( g_v \) is update frequency of view \( v \).

\[
\tau(G, M) = \sum_{i=1}^{k} f_{q_i} \cdot Q(q_i, M) + \sum_{i=1}^{m} g_{v_i} \cdot UC(v_i, M)
\]  

Any AND-OR view graph can be converted into an equivalent query view graph. A query view graphs \( G \) is a bipartite graph \((Q, v, \zeta, E)\), where \( Q \) is the set of queries to be supported and \( \zeta \) is a subset of all views \( V \). An edge \((q, \sigma)\) is in the set of edges \( E \) iff the query \( q \) can be answered using the views in the set \( \sigma \) and the cost associated with the edge is the cost of answering \( q \) using \( \sigma \).

The AND-OR Greedy algorithm for query-view graph for view selection works as follows. At every stage a connected sub-graph \( H \) of \( F_v \) is picked such that its corresponding set of views \( V_H \) offers the maximum benefit per unit space at that stage. The sets of views \( V_H \) is then added to the set of views \( M \) already selected in the previous stage. The algorithm stops and returns \( M \) when the constraint value of \( M \) exceeds \( S \).

Validation: In Derrar & Boussaid (2013), an approach based on exploitation of recent statistical data access like query frequencies, in data warehouse for dynamic fragmentation in data warehouse is presented that significantly reduces query response time. In Gupta & Mumick (2005), proofs are presented to show that this algorithm is guaranteed to provide a solution that is fairly close to the optimal solution. The heuristic (in Gupta & Mumick (2005)) is extended to the general AND-OR view graphs. But evaluation of the algorithm in terms of the quality of solutions is not provided.

Limitations: The AND-OR View Graph based greedy algorithm considers few frequent queries with some shared views. In case of a large number of complex queries with large
number of shared views and queries, with different query processing plan may result in different optimum configurations. Therefore, instead of computing costs and benefits of materializing the views of different segments of the bigraph, a common view processing plan may be more suitable.

3.1.4 Optimal query plan and heuristic algorithm for selecting views to materialize

This approach presents an algorithm for constructing Multiple View Processing Plans (MVPP) graph and an algorithm to select views for materializing using the MVPP graph (Yang et al. 1997). To generate an MVPP graph, individual optimal query processing plans are merged. The algorithm for generating the MVPP graph is as follows. First, for every individual optimal plan, if there is a join operation involved, push the select and project operations up along the tree; and then, for two such modified optimal query plans, first find the common sub expressions for the join operations if they share the same source relations, and then merge them. Ultimately the goal is to push down all the select, project and aggregate operations as deep as possible in the tree.

If view $v$ is materialized, the total cost involved in a query plan is defined as in Equation 2. Here $q \in R$ is the set of queries, $r \in L$ is the set of base relations, $f_q$ is the frequency of executing queries and $f_u$ is the frequency of updating base relations. For each $v \in M$, $C_{aq}^u(v)$ and $C_{am}^u(v)$ are the cost of access for query $q$ using view $v$ and cost of maintenance of view $v$ based on changes to base relation $r$, respectively. The problem is to find a set $M$ so that if the members of $M$ are materialized, the value of $C_{total}$ will be the smallest among all the feasible sets of materialized views.

$$C_{total}(v) = \sum_{q \in R} f_q C_{aq}^u(v) + \sum_{r \in L} f_u C_{am}^u(v)$$  \hspace{1cm} (2)

Let $M$ be a set for keeping views selected for materialization, initialized as empty. $D(v)$ returns the set of ancestors of view or node $v$ and weight of a node $w(v)$ is defined by Equation 3. Here $O_v$ denotes the set of global queries which use view $v$, and $I_v$ denotes the base relations which are used to produce $v$. $S_v$ is the set of nodes (both leaf and intermediate) which are used to produce $v$ and $LV$ is the list of nodes based on descending order of $w(v)$. Whenever a new node is considered for materialization, the saving it brings in is calculated after accessing all the queries involved, subtracting the cost for maintaining this node as expressed in Equation 4.

$$w(v) = \sum_{q \in O_v} f_q C_{aq}^u(v) - \sum_{r \in I_v} f_u(r).C_{am}^u(v)$$  \hspace{1cm} (3)

$$C_s = \sum_{q \in O_v} \{ f_q(C_{aq}^u(v) - \sum_{u \in S_v \cap M} C_{au}^u(u)) \} - \sum_{r \in I_v} \{ f_u(r).C_{am}^u(v) \}$$  \hspace{1cm} (4)

The algorithm for selecting views to materialize is given in Algorithm 2. The algorithm is used to determine a set of views ($M$) for materialization where the sum cost of processing all the queries and maintaining all the views is the smallest possible.

Validation: A query can have multiple execution plans. In this algorithm, for a set of query execution plans the sharing of different views are mapped into MVPP graphs providing a clear and simple representation. This heuristic algorithm provides a near optimal
Algorithm 2 View selection using optimal query plan

Require: An MVPP graph
Ensure: The selected set of views $M$ for materializing

1: Compute the weights of nodes
2: Create list $LV$ for all the nodes (with positive value of weights) based on the descending order of their weights.
3: repeat
4: Pick up one view $v$ from $LV$
5: Generate $O_v$, $I_v$, and $S_v$
6: Compute $C_s$
7: if $C_s > 0$ then
8: Insert $v$ into $M$ and remove $v$ from $LV$
9: else
10: $v$ and all the nodes are removed that are listed after $v$ and are in the subtree rooted at $v$
11: end if
12: until $LV$ is empty
13: for $v \in M$ do
14: if $D(v) \subset M$ then
15: remove $v$ from $M$
16: end if
17: end for
18: return $M$

solution using 0-1 integer programming. Yang et al. (1997) presented that the heuristic algorithm for generating multiple MVPP is just of complexity $O(n)$. Therefore, for finding any reasonable solution of selecting views, this model may be used.

Limitations: Though this model is just good for selecting reasonable solutions, but for optimal MVPP selection and thereby to select a set of views with optimum costs, the complexity of 0-1 integer programming approach is of $O(2^n)$. Therefore, when there is a huge query-workload, the MVPP graph becomes very complicated and the cost of generating the MVPP graph becomes very high. In fact, all heuristic methods are effective for this problem when the number of views is relatively small (Derakhshan et al. 2008).

3.2 Randomized algorithmic approaches

Randomized algorithms are based on the logic that it is sometimes beneficial if randomness is deliberately introduced into a search process as a mean for speeding convergence and making the algorithm less sensitive to modelling errors. As the problem at hand is NP-hard, several randomized stochastic optimization methods have been proposed (Derakhshan et al. 2006, 2008, Horng et al. 1999, Loureiro & Belo 2006, Zhang & Yang 1999, Lee & Hammer 2001, Sun & Wang 2009, Qingzhou et al. 2009).

3.2.1 Simulated Annealing (SA) algorithm based models

Derakhshan et al. (2006, 2008) introduce a set of approaches for materialized view selection based on Simulated Annealing (SA) in conjunction with the use of MVPP graph. Given an MVPP graph, they attempt to find the best set of intermediate nodes (views) that can
answer all queries with minimal cost. The set of views of the MVPP graph are labelled and represented as a binary string of 1s and 0s to represent views that will and will not be materialized, respectively. The nodes in the MVPP graph are numbered starting at the base relation moving left to right, and continued up to the rightmost node at the top of the graph. Nodes are thus numbered or labelled 0 to \( m - 1 \), (where \( m \) is the number of intermediate nodes). A mapping array of size \( m - 1 \) is used, where each index in the array corresponds to a graph node. An array element '1' denotes that the corresponding node in the graph is materialized and '0' if the node is not materialized. From this matrix, different strings of 0s and 1s are obtained by perturbing the initial string by changing every time one bit from '1' to '0' or '0' to '1'. The simulated annealing algorithm that is executed is given in Algorithm 3. The resultant \( s \) is the solution configuration.

\begin{algorithm}
\caption{Simulated annealing for selection of views to materialize}
\label{algorithm:sa}
\begin{algorithmic}[1]
\Require An MVPP graph with view labels and sizes, base relation sizes, base relation updating frequencies, query frequencies, query response sizes
\Ensure A solution string of bits \( s \)
\State Define: Initial temperature \( T \), terminating temperature \( T' \), space constraint \( C \), maximum number of iteration \( I_{max} \)
\State Initialize a candidate solution string \( s \) such that it satisfies space constraint \( C \)
\Repeat
\For \( I = 1 \) to \( I_{max} \)
\State \( s' \leftarrow \text{perturb}(s) \)
\State \( E = \text{cost}(s) \)
\State \( E' = \text{cost}(s') \)
\State \If \( (E' < E) \) or \( \text{(random()) < } e^{(E-E')/T} \)
\State \If \( s' \) satisfies the constraint \( C \)
\State \( s \leftarrow s' \)
\EndIf
\EndIf
\EndFor
\State \( T = \text{decrement}(T) \)
\Until \( T > T' \)
\Return \( s \)
\end{algorithmic}
\end{algorithm}

Simulated annealing (SA) is considered a good tool for nonlinear optimization problems, but a major disadvantage is that it is extremely slow at times and hence, parallel versions of the algorithm have been developed. Derakhshan et al. (2008) use Parallel Simulated Annealing (PSA) in the materialized view selection problem by using MVPP graph as input. In SA, the solution quality is affected by the numbers of time that the initial solution is perturbed. By performing simulated annealing with multiple inputs over multiple computer nodes, PSA is able to increase the quality of obtained sets of materialized views.

The view selection for materialization problem is usually formulated as a single objective optimization problem. But, in Goswami et al. (2012) an attempt also has been made to solve this problem using the Multi Objective Simulated Annealing (MOSA) and Archived Multi-Objective Simulated Annealing (AMOSA) algorithms (Bandyopadhyay et al. 2008).

Yuhang et al. (2010) present an algorithm that combines Clonal Selection Algorithm (CSA) with SA algorithm. In this technique, during clonal selection for mutation, it accepts
non-optimal solutions also on certain probability to avoid pre-mature convergence. Thus this version of SA based technique improves efficiency of the algorithm and the quality of solution. This algorithm represents candidate solution set as antigen of the antibody of CSA and first searches global optimal solution from the initial population and brings in new antibody population through perturbation of clones, variation and selection. According to antibody and antigen affinity function on the basis of the simulated annealing metropolis criterion in the variation process, the algorithm decides whether to accept the new antibody (candidate solution) for subsequent steps of simulated annealing or not. This process is repeated till it reaches the minimum temperature specified. Yuhang et al. (2010) claim that this hybrid algorithm has more chance of escaping from local optimum and reaching the global optimum, compared to Genetic Algorithm (GA) and CSA.

**Validation:** Experimental results as reported by Derakhshan et al. (2006) show that the cost of selected views is considerably better than ones obtained by the previously reported heuristics. By using simulated annealing, the cost of a selected set of materialized views comes down by up to 70% (Derakhshan et al. 2008) than the cost obtained by genetic and heuristic algorithms. Also, in Derakhshan et al. (2008) experimental studies show that parallel simulated annealing provides a significant improvement in the quality of the obtained set of materialized views over existing heuristic and sequential simulated annealing algorithms.

**Limitations:** In Yuhang et al. (2010), authors present that the hybrid algorithm combining CSA and the Metropolis rule of SA in view selection problem has quicker convergence rate than GA. But when the solution space is smooth (e.g. gradient descent), heuristic and simpler methods work much better than SA.

### 3.2.2 Genetic Algorithm (GA) based model

As the problem at hand is NP-hard, Evolutionary Algorithms (EA) such as GA is likely to provide efficient solutions (Zhang & Yang 1999). To obtain better solutions from a large number of views taking into account view maintenance and query processing costs, GAs have been used (Zhang & Yang 1999, Zhang et al. 2001, Lee & Hammer 2001). In this approach, the AND-OR view graph notion is used for generating a string of bits where the bit in position $i$ (starting from the leftmost bit as position 1) is 1, if the view $i$ is selected for materializing and else 0. These strings of bits are considered as a genome of the population (Lee & Hammer 2001). That is, the sets of candidate configurations (views and indexes) are referred to as genomes of the candidate population. The Genetic Algorithm (GA) uses a multi-directional search by maintaining a pool of candidate points in the search space. Information is exchanged among the candidate points to guide the search process using the evolutionary concept i.e. fit candidates survive while unfit candidates die. A fitness function, which evaluates the superiority of a genome, is used in this process. The fitness function is used to evaluate a genome with respect to query benefit, i.e., reduction in the query cost due to materialization of query. Whenever a view is selected, the benefit not only depends on the view itself but also on other views that are selected and corresponding materialized view maintenance cost. Therefore, a penalty value is used as a part of the fitness function. Penalty is applied in three different ways when calculating the fitness. (i) Subtract mode that Calculate the fitness by subtracting the penalty value from the query benefit. Since the fitness value cannot assume a negative value, fitness is set to 0 when the result of the calculation becomes negative, (ii) Divide mode that divides the query benefit by the penalty value in an effort to reduce the query benefit. When the penalty value is less than 1, the division is not performed to prevent the fitness from increasing and (iii)
Subtract and divide mode that combines the two methods (i) and (ii). If the query benefit is larger than the penalty value, subtract mode is used. If the penalty value is larger than the query benefit, divide mode is used. The penalty value is calculated using a penalty function. The cost model used is as defined in Equation 1. For crossover operation, each genome is selected with a probability and the selected genomes are paired. For each pair, a crossover point is randomly decided and information exchanged among genomes. For the mutation operation, for all genomes, for each bit in the genome, the bit is mutated (flipped) with a probability. The selection, crossover, mutation and evaluation processes are repeated in a loop until the termination condition is satisfied. Thus after several generations, it is expected that the resultant population is composed of superior genomes, i.e., superior combinations of views for materialization. An example GA-based approach applied to the Materialized View Selection problem is given by Lee & Hammer (2001). In Wagner & Agrawal (2013) an Evolutionary Algorithm (EA) is used by representing the view selection problem as weighted materialized view selection problem where both the amount and importance of data retrieved are considered.

**Validation:** The GA uses a multi-directional search over a pool of candidate solution points in the search space. The multi-directional evolutionary process allows the GA to efficiently search the space and find a point near the global optimum (Lee & Hammer 2001). Lee & Hammer (2001) presented that their solution, in speeding up materialized view selection, is better than the existing solutions, in terms of expected run-time behaviour as well as the warehouse configuration obtained. It is also claimed that this approach makes a dramatic improvement in time complexity over existing heuristic search based models. According to Lee & Hammer (2001), their algorithm yields solution that lies within 10% of the optimal query benefit, exhibiting only a linear increase in execution time.

**Limitations:** The drawback of GA is that mathematically there is no validity proof for the solutions obtained. It also needs more function evaluations than other linear methods. There is no guaranteed convergence to global minimum and the convergence is usually slow.

### 3.2.3 Memetic Algorithmic (MA) model

The memetic algorithm (MA), first proposed by Moscato (1989), is similar to GAs but the elements that form a chromosome are called memes, not genes. In MA, all chromosomes and offspring are allowed to gain some experience, through a local search, before being involved in the evolutionary process. In Qingzhou et al. (2009), the authors use MA in the materialized view selection problem. The AND-OR view graph representation is used for constructing the memes and the cost model is based on Equation 1. A local optimizer is applied to each offspring before it is inserted into the population. Thus a local search mechanism is used in addition to other parameters of GA, i.e., population size, number of generations, crossover rate, and mutation rate. To improve GAs by reducing slow convergence for each generation, the MA presents a new and enhanced EA.

**Validation:** With the model suggested by Qingzhou et al. (2009), by setting system parameter values as population size=20, maximum number of generations=50, selection rate=0.85, cross-over ratio=0.8 and mutation rate=0.5, if without loss of generality for the space constraint a random view invoking frequency in the range [0,1] with 10% to 90% of the total size of all views are considered, the MA outperforms heuristic algorithm and GA in all cases regardless of storage space.

**Limitations:** The basic difference between GA based model and MA based model is that, in MAs a local optimizer is applied to each offspring (of GA) before it is inserted into
the population to improve the performances of the GA. This reduces the slow convergence for each generation (Qingzhou et al. 2009). However the other drawbacks of GA remain in MA based models.

3.2.4 Particle Swarm Optimization (PSO) in selecting views for materializing

The PSO technique has also been used in the materialized view selection problem (Sun & Wang 2009). Sun & Wang (2009) show that PSO achieves much better performance than heuristic algorithms and GAs. The mathematical model of the materialized view selection problem is based on the AND-OR graph as in Equation 1. Like GA and MA, in PSO as presented in Sun & Wang (2009), each AND-OR view graph is encoded as a binary string where 0 indicates that the corresponding node (view or query) is not materialized and 1 indicates that it is materialized. The binary strings generated are considered the particles of the PSO algorithm. The fitness function used is the cost function $\tau(G, M)$ as defined in Equation 1. Each particle knows its fitness value and at a particular stage the best fitness value is taken as the personal best position. The particle with the best fitness value among all particles at a specific iteration is denoted the global best fit position. The velocity of each particle is modified according to the Equation 5, where, $t$ is the iteration number, $p_i$ is i\textsuperscript{th} particle’s personal best position, $p_{gb}$ is global best fit position, $x_i(t)$ is the position of $i$\textsuperscript{th} particle at iteration $t$, $i_{max}$ is the maximum number of iterations and $w_i = w_{max} - ((w_{max} - w_{min})/i_{max}) \times i$.

$$v_i(t + 1) = w_i v_i(t) + c_1 r_1 (p_i - x_i(t)) + c_2 r_2 (p_{gb} - x_i(t))$$

The position of each particle is modified according to the Equation 6.

$$x_i(t + 1) = x_i(t) + v_i(t + 1)$$

If the global best fit value $p_{gb}$ does not improve or the iteration number has not reached the limit, the process is repeated. The particle with the best fitness value $p_{gb}$ at the end is the best binary string that gives the best set of views for materializing.

**Validation:** The PSO with system parameters set as population size=50, maximum iteration number=100, $c_1$=$c_2$=2, $r_1$ and $r_2$ as two random functions in the range [0,1], with maximum velocity $v_{max}$=20 and minimum velocity $v_{min}$=2, considering view random invoking frequencies in range [0,1] (for space constraints), Sun & Wang (2009) presented that regardless of the storage space constraint, the total maintenance cost of PSO based model is much lower than those of heuristic algorithm and GA based models.

**Limitations:** Though the experimental results reported in Sun & Wang (2009) demonstrate that the PSO algorithm to solve the materialized view selection problem in designing data warehouse achieves much better performance than other heuristic algorithms and GAs, the major drawback of PSO is premature convergence and getting trapped in local optima (Sedighizadeh & Ellips 2009).

3.2.5 Ant Colony Algorithm (ACA) for optimizing view selection for materialization

In Maniezzo et al. (2001), Gu et al. (2007), Song & Gao (2010), Ant Colony Algorithm (ACA) is used for optimal selection of views for materializing in Data warehouse. In this approach for view selection problem, an ant is defined as a set of views representing a
solution to the problem. In ACA based view selection optimization, for a specified number of iterations, each ant moves in the solution space to find the local optimum. In the traversal along the solution space, the numbers of time the solutions are visited by the ants are used as parameter to a function to update a value representing the pheromone updating process (or the pheromone evaporation controlling process) in ACA. The route of subsequent ants is guided by the value of the pheromone function. This function uses several parameters like pheromone level at a state, relative effect of paths, expected effects of path and number of paths available for each of the ants while updating the pheromone level in a path. This pheromone updating function thus guides the ants to different solution search paths to avoid trapping in local optimum. The mostly visited solution by the ants in an iteration is selected as the best solution for that iteration. At the end, the global optimum solution is selected out of all the local optimum solutions in different iterations.

**Validation:** In Song & Gao (2010) it is shown that, using ACA it is easier to find an optimum set of views for materializing in data warehouse, compared to GA. With 32 candidate views, 10 numbers of ants, the convergence trend of query cost of ACA is found to be better than GA based view selection technique with respect to number of iterations. Under different space limitations (of ACA), the total query cost of materialized views by both GA and ACA are found to be almost same (Song & Gao 2010).

**Limitations:** The solutions by ACA approach for view selection problem used in Maniezzo et al. (2001), Gu et al. (2007) and Song & Gao (2010), largely depend on the parameters such as the defined pheromone (constant) in each path at the beginning, defined value of relative and expected effects of paths, and the constant number of ant tracks defined.

### 3.3 Data Mining based approaches

Data mining techniques have also been used to handle the view selection for materialization problem (Aouiche et al. 2006, Aouiche & Darmont 2009, Das & Bhattacharyya 2005, Kumar et al. 2012). In Das & Bhattacharyya (2005), a density-based view materialization algorithm is discussed using data cube lattice structure, view size, access frequency of the views, and support (frequency). In Aouiche et al. (2006), Aouiche & Darmont (2009) and Kumar et al. (2012), clustering techniques are used to cluster similar queries by analysing the query workload of the warehouse. For each cluster of queries, the candidate set of queries for materialization is decided. Then by a merging process on different query clusters, a configuration of candidate views is built. From the candidate views the final view configuration is created with a greedy algorithm.

#### 3.3.1 Clustering based materialized view selection model

Aouiche et al. (2006) present a clustering approach based materialized view selection technique. Later in 2009, this technique was extended for selecting relevant configuration of indexes and views for materializing (Aouiche & Darmont 2009). Workloads in data warehouse are sets of generalized projection-selection-join queries. In this technique, from the workload, the attributes that are present in 'where' and 'group by' clauses of each query are extracted along with aggregation operators and join conditions of different joins and tables. These attributes are termed *representative attribute*. Each query is represented as a row of 1s and 0s in a two dimensional matrix such that each cell is set to 1 if that representative attribute is present in the query and else 0. Thus, we get a two dimensional matrix where queries are rows and attributes are columns. The matrix is called *representative attribute matrix* of the workload queries. The associations between the join attributes and queries
are kept in another associated matrix. Using the representative attribute matrix of workload queries, the queries are clustered into a number of clusters of similar queries. Simple Hamming distance based similarity and dissimilarity functions are used for constructing the clusters. For each cluster of queries, a set of most shared views is selected and a merging process is used to merge some of these views to generate a new configuration for a candidate set of views for materializing. In the view merging process, views are selected for merging to one view when the accessing cost and space cost of the new (merged) view is less than the costs if they are not merged. This merging process reduces the number of views in each set of candidate configuration of views and indexes for materializing. A greedy algorithm evaluates the benefits of materializing the candidate sets of views by computing the access cost and storage cost and select the optimum set of views for materializing.

Validation: Clustering and merging of views to generate new sets of candidate views for materializing reduces the number of views that are to be supplied to the greedy algorithm for selecting the optimum set of views and thereby it reduces complexity. In Aouiche & Darmont (2009), presented by experimenting with an ad-hoc bench mark data warehouse that, the selection of views by clustering based model significantly improve query execution time considering availability of storage space for materializing views. Though it is obvious that increased number of materialized views by not considering storage space limitation means lesser query processing time, the study shows that the average gain in performance is 68.9% when 35.4% of available storage space is used. The gain in performance is 94.9% when 100% of available storage space is used.

Limitations: Simple Hamming distance based similarity and dissimilarity measures, as used in Aouiche et al. (2006), may lead to generation of less diverged candidate solutions. One big issue in clustering based optimization techniques is that the solution quality depends on the size or quality of clusters, and which depend on clustering parameters and the clustering algorithm used.

3.3.2 Association Rule Mining and Clustering in materialized view selection

Das et al., in 2005, present a density-based clustering for view materialization that uses association rule mining for selecting views for materializing in average runtime complexity $O(n\log n)$ (Das & Bhattacharyya 2005). The algorithm uses data cube lattice, view size, access frequency of the views and support (frequency) of the views in selecting the views to be materialized. Clusters of views are formed in this algorithm by computing a benefit function on candidate views of a specified workload assuming that the views are organized in the form of a lattice. For each cluster of views, the core subset of frequent views is selected by association rule mining for materialization.

Kumar et al. (2012) propose another approach that attempts to identify frequent information that is accessed by past queries on a data warehouse, using clustering and association rule mining techniques. In this technique authors attempt to form clusters of subject areas of past queries using a density based clustering algorithm known as OPTICS (Ordering Points to Identify Clustering Structure) (Ankerst et al. 1999). Overlapping of database relations among queries are used in evaluating similarity or dissimilarity while constructing clusters. A frequent set of views for each cluster of subject areas is then determined by using association rule mining. The identified frequent sets of views against different subject areas are considered for materializing to serve future queries on respective subject areas.

Validation: Association rule mining based view selection techniques are used in identifying frequent database relations or views that may be materialized for quick response
to future queries in respective subject areas. The study by Aouiche et al. (2006) shows that just for 0.05% storage space occupation by selected views can obtain 22.95% of the query results without further processing. Thus, even for small storage space availability for materializing views, this strategy helps building views for materializing that cover large number of queries.

**Limitations:** Some infrequent relations or views may also have importance in some query processing scenarios. These relations may not be considered in association rule mining based strategy for view selection. Dynamic clustering is yet to be implemented in this problem. Another limitation of this strategy is that the solution quality by association rule mining largely depends on the support and confidence thresholds used.

## 4 Solutions and Discussion

Based on our study and analysis, we observe that deterministic and heuristic algorithms for the view selection problem are often not truly scalable i.e., these methods are effective only with a small number of views. Since it is an NP-hard problem, several randomized and Evolutionary Algorithms (EA) have been introduced. However, they have limitations as well.

Genetic Algorithmic approaches are able to perform better in multi-directional search over a set of candidate views in the search space. Information exchange occurs among candidate solutions to lead the search to regions of search space where good candidates survive while bad candidates die. Thus, GA approaches that operate in a multi-dimensional fashion can provide effective search performance and find a solution near a global optimum in the view selection problem. However, the SA approach generates solutions with (view maintenance and query processing) costs up to 70% less than the GA and other heuristic approaches in this problem (Derakhshan et al. 2006, 2008). Another major limitation of the evolutionary approach is that it hard to acquire good initial solutions, and therefore in the view selection problem, GA-based approaches converge slowly. It has been observed that, SA (Derakhshan et al. 2006) out performs Heuristic algorithmic approach (Yang et al. 1997) and EA (with heuristic view processing plan selection) approach (Zhang et al. 2001), in case of query processing plan based view selection models as presented in Figure 4. Particle Swarm Optimization (PSO) and memetic algorithm (MA) approaches may achieve better performance than GAs in the view selection for materializing in data warehouse (Sun & Wang 2009, Qingzhou et al. 2009) as presented by a graph in Figure 5.

In data mining approaches, the basic assumption is that the queries of the same cluster can be answered competently by the same set of materialized views. Therefore, all queries are not necessarily analysed for generating candidate views. This reduces the number of candidate views. By changing the clustering parameters, the number of clusters can be controlled. Clustering is performed using some kind of similarity thresholds among queries. Thus the cluster quality depends on parameters. Hence the candidate view itself are quasi-optimal and due to this the final selection of views may not be the most optimum. However, unlike the other methods, the data mining approaches generate a representative attribute matrix of workload queries, which is simple for building and browsing.
Table 2  Different algorithms used in view selection techniques and issues involved.

<table>
<thead>
<tr>
<th>Types of algorithms for view selection</th>
<th>Major issues involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic Algorithms</td>
<td>HRU-Greedy Algorithm, PGA</td>
</tr>
<tr>
<td></td>
<td>Optimal query plan based algorithm</td>
</tr>
<tr>
<td>Randomized algorithms</td>
<td>SA and Parallel Simulated Annealing (PSA)</td>
</tr>
<tr>
<td></td>
<td>GA</td>
</tr>
<tr>
<td></td>
<td>PSO and ACA</td>
</tr>
<tr>
<td></td>
<td>MA</td>
</tr>
<tr>
<td>Data Mining with Clustering algorithm and Association Rule mining</td>
<td>Merging of different sets of candidate views for different clusters of queries depends on merging parameters. Clustering depends on parameters used and thereby solution depends on these parameters. Solution quality depends on thresholds on support and confidence measures used by Association rule mining.</td>
</tr>
</tbody>
</table>
Figure 4: Relative costs of Heuristic, Evolutionary and Simulated Annealing algorithm in view selection using query processing plan graph representation.

Figure 5: Comparison of GA, PSO and MA based view materialization models with respect to total query processing costs vs. space used by materialized views.
5 Conclusion

5.1 Contribution

In this survey, we have analysed and discussed various techniques used in view selection for materialization in data warehousing. By analysing the problem representations, data structures, algorithms and parameter selections in different models proposed so far, we have identified and reported the associated issues and challenges in addressing this NP-hard problem. It is expected that by addressing these issues and challenges, the complexity of the view selection problem can be reduced and scalability is achieved.

5.2 Limitations of the study

For critical analysis of different techniques in any area, researchers and practitioners need a common protocol for performing experiments using standard datasets and standard benchmarking. Although it is a difficult task to introduce one common framework or a single generalized software environment for comparison of all techniques, it will be very beneficial to move toward the use of a common dataset and benchmarking for evaluation. For extensive analysis of different approaches, it is expected that Transaction Processing Council (TPC) will come-up with voluminous benchmark dataset (Hsu et al. 2001), with a standard framework for experimentally evaluating these techniques for view selection for materialization problem.

5.3 Implications to theory and practices

We have identified the following implications to theory and practices with different approaches in handling the view selection for the materialization problem.

- **Scalability:** Deterministic search for solution using heuristics in the view selection problem decreases the solution space. But when the size of the data warehouse is very large, scalability is a big issue due to exponential complexity. Though some heuristic algorithms have been designed with reduced time complexity, they are yet to be tested on very large databases and a large number of complex queries. Evolutionary approaches like GA, determine a solution to be the fittest depending on predefined numbers of generations and iterations. Defining a scalable generation number, iterations per generation and penalty functions are the main problems with EA. EA and other randomized algorithms in the view selection problem use AND-OR view graph of queries as input. Application development for analysis of a large number of complex queries for AND-OR view graph generation is yet to be done. Soft-computing approaches in the view selection problem use clustering and associative rule mining on a query-view matrix. The quality of the quasi-optimum solutions discovered by these techniques depends on the quality of clusters and/or cluster sizes and thereby they depend on pre-defined clustering parameters. Measures needed in association rule mining like support and confidence, largely depend on the size of the database or the matrix used.

- **Data structure:** Heuristic view materialization techniques use the lattice representation of views. This makes it a non-polynomial problem. Methods suggested to convert the conventional heuristic view selection techniques to polynomial
complexity need a lot of pre-computation (Nadeua & Teorey 2002). Randomized algorithm based techniques in the view selection problem use query processing plan graphs or AND-OR view graphs. Though most studies on the applicability of randomized algorithms talk about the superior performance of the algorithms in handling the problem, detailed analysis on the data structure is lacking. The query-view matrix representation as used in clustering and associative rule mining techniques is only specific to clustering algorithms and parameters used.

• **Cost model:** The HRU-greedy algorithm and the polynomial greedy algorithm for the view selection problem compute benefit of materializing a set of views by computing the total query processing cost and the cost savings by the selected views heuristically. The query processing cost is the number of rows that are to be accessed by aggregating functions used in the lattice representation of a data warehouse. Query frequencies and materialized view maintenance costs are not considered. Some other heuristic as well as randomized algorithmic approaches consider query frequency and view updating frequency as shown in Equation 1 or 2 in their cost model for computing benefits of candidate solutions. In multi-objective optimization based solution models, where query processing costs and materialized view maintenance costs are the objectives for optimization, extending the degree of diversity among selected solution population from a large number of solutions generated in intermediate iterations are related issues. The data mining based model aims to minimize the execution cost of a set of workload queries under storage space constraint. The quality of solutions of these models largely dependent on the support threshold used. Estimation of appropriate support threshold and fulfilling the completeness criteria are additional research issues in minimizing the query execution costs by data-mining based approaches.

• **Parameter selection:** Solution quality for randomized algorithms, including evolutionary algorithms, largely depends on the number of iterations or the number of generations specified. In simulated annealing approaches, the solution quality depends on parameters such as the initial temperature, the final temperature and the rate of temperature decrement. To use data mining in the view selection for materializing problem, algorithms are to be designed in such a way that they perform consistently with varied clustering parameters and associative rule mining measures like support and confidence levels. When using multi-objective optimization techniques in the view selection problem, selecting filtering parameters for increasing the degree of diversity among a large number of pareto-optimum solutions is an open issue.

5.4 **Further research directions**

From our humble survey on approaches in view selection for materializing in data warehouse for efficient query response, we found that a technique applicable for large high dimensional realistic data warehouses, independent of its schema, as well as applicable for big-data framework (Philip Chen & Zhang 2014) with reasonable run time and space complexity is needed to be designed. A cost effective method to input queries from large query workload based data warehouse and a generalized data structure for storing them, are also to be developed. Designing a flawless test-bed with unprejudiced (benchmark) databases to evaluate different approaches is yet to be taken up for handling this NP-hard problem. Finally, the future focus should be on developing an analytical model for big and complex view processing environment which can handle all the issues and challenges discussed here.
Approaches and Issues in View Selection for Materializing in Data Warehouse

Acknowledgement

The authors would like to thank the anonymous reviewers for their detailed and constructive feedback along with the IJBIS editors, who greatly helped in improving this paper.

References


Gupta, H. (1999), Selection and maintenance of views in a data warehouse, Ph.d. thesis.


Gupta, H. & Mumick, S. (2005), ‘Selection of views to materialize in a data warehouse’, IEEE Transactions on Knowledge and Data Engineering 17(1), 24–43.


Approaches and Issues in View Selection for Materializing in Data Warehouse


