Chapter 2

Materialized View Selection in Data Warehouses: Approaches, Issues and Challenges

2.1 Introduction

Research on the problem of selecting views to materialize in data warehouses started in the early nineties when several heuristic greedy algorithms were proposed [4, 6, 21, 22, 24]. With the increasing growth in dimensionality of data warehouses, the solution space also grows exponentially [3, 4, 21, 22, 33]. To address this issue, various stochastic, evolutionary, data mining and clustering based optimizing approaches have been proposed with different data structures and representations of the problem.

Several greedy approaches have been proposed by defining different cost and benefit parameters to deal with the view selection for materializing problem [4, 6, 21, 22, 24]. Most of these approaches use multidimensional lattice structures to select views for materialization, based on the original greedy algorithm proposed by Harinarayan et al. in [4], and popularly referred to as the HRU-Greedy algorithm. In [22] and [33], 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. Selection of views for materializing closely resembles with 1/0 Knapsack problem but with the difference that selection of a view depends on what are the other views that have been selected so far. With the exponential growth of solution space with increase in number of dimensions of data warehouses and candidate views for materialization the view selection problem becomes NP-hard [3, 4, 21, 22, 33]. Therefore, most recent approaches use stochastic or randomized algorithms like Simulated Annealing (SA), Genetic Algorithms (GA), Particle Swarm Optimization (PSO) etc.. Most of these approaches use graphical representations of query workloads. Wagner et al. designed an evolutionary algorithm for view selection problem by considering the amount and importance of

data retrieved by data warehouse queries [34]. Data mining techniques also have been used effectively on workloads (sets of queries) representative of data warehouse usages to deduce quasi-optimal configurations of materialized views and/or indexes [10–13].

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. In this chapter, a detailed review of literature surveyed on techniques proposed for selecting views to materialize in data warehouses are presented by introducing respective data representations with discussion on various research challenges and associated issues. Different representations used for handling this problem with their associated issues are presented in Section 2.2. In Section 2.3 existing techniques for selecting views to materialize in data warehouses are discussed. Section 2.4 presents a discussion on performances of solution models suggested so far with related issues and challenges. In Section 2.5, concluding discussion about contribution from this survey and limitations of the study are presented.

2.2 Representations of views in Data Warehouses

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 started in 1996 by representing views and data warehouses for applying greedy algorithmic approaches and heuristic approaches in materialized view selection problem. Later from late nineties query processing graph based models have been used for representing this problem. From 2005 onwards multiple query processing plan based model has become popular for incorporating SQL sub-expression results as views by considering indexes and keys of relational model to deal with general SQL queries that include select, project, join and aggregation operations. Parallel to these models few query-attribute-view matrix based models also have been proposed for applying data mining and clustering techniques for handling this problem. The surveyed literature on different representations and approaches are reviewed in following sub-sections.

2.2.1 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 [4]. 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. in [4] 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 materi-

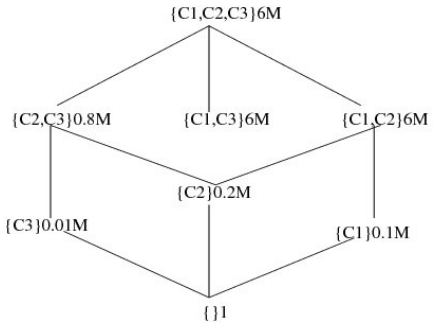

Figure 2-1: A lattice structure for 3 attributes

alizing 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 [4, 6, 35]. Each node of the lattice structure represents a view labeled 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 2-1, 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 [4] 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 $T = m \times A + C$, where T is the running time of the query on a view of size A. 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.

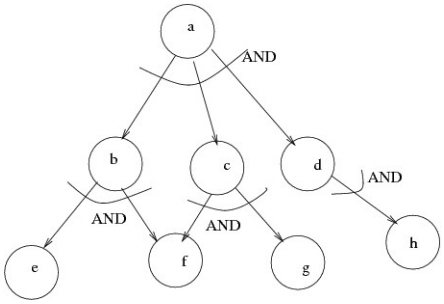

Figure 2-2: An Expression AND DAG

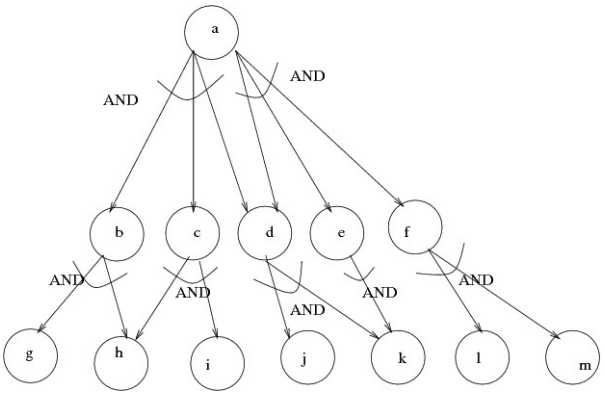

Figure 2-3: An Expression AND-OR DAG

2.2.2 AND-OR view graph representation of queries and views

In [22] and [33], 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 graph*s (DAGs). In this representation, an *OR-View graph* is defined to express that any view can be computed from any of its related views. An *AND-view graph* is used to express that a query, sub-expression of query or a view is uniquely evaluated by some other views. These DAGs are defined as *expression DAG*s of queries and views. Using these notions a directed acyclic graph termed as AND-OR view graph is defined. The AND DAGs and AND-OR DAGs are used to represent sub-expressions of queries. Therefore these are termed as *expression DAG*s. This model of representation is defined by three basic definitions as stated below.

Definition 5. *An **expression 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. All of the views* v_1, v_2, . . . , v_k *are required to compute the cost of u when a node or view u has outgoing edges to nodes* v_1, v_2, . . . , v_k. *This dependence is indicated by drawing a semicircle called an AND arc to indicate the dependency of evaluating u through the edges* (u, v_1), (u, v_2),..., (u, v_k) *[36].*

Such an AND arc has an operator and a cost associated with it, which is the cost incurred during the computation of u from $v_1, v_2, ...v_k$.

But AND DAGs do not depict alternative ways of evaluating a view. Therefore, the *expression AND-OR DAG* is defined which may have more than one AND arc at each node. An expression AND DAG and an expression AND-OR DAG is illustrated in Figure 2-2 and 2-3 respectively. The expression AND-OR DAG is defined below.

Definition 6. *An **expression AND-OR DAG** for a view or a query v is a DAG with v as a source and the base relations as sinks such that each non-sink node is associated with one or more AND arcs. More than one AND arc at a node depicts multiple ways of computing that node [36].*

Using Definitions 5 and 6 an *AND-OR view graph* is defined as Definition 7 for defining the materialized view selection problem.

Definition 7. *A DAG G, with base relations as the sink is called an AND-OR view graph for a set of views and query responses $v_1, v_2, \cdots v_k$, if for each v_i, there is a sub-graph G_i in G that is an expression AND-OR DAG for v_i. Each node v in an AND-OR view graph has the following parameters associated with it: space A_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) [36].*

The view selection for materialization problem using AND-OR View graph is stated as - given an AND-OR view graph G and a quantity A (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 A under a maintenance-cost constraint [22, 36].

In [36], 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 [37–40] and [41] 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 analyzing 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 thus, sharing of materialized views by multiple queries is ignored.

2.2.3 Optimal multiple query execution plan based graphical representation

Another approach used in view selection for materializing in data warehouses uses a DAG representing all frequently asked queries or a specific number of queries by a query processing strategy of warehouse views [7]. 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 2-4.

- Query 1:

```
SQL> select employee.name from employee,
dept where dept.location='Tezpur' and
employee.deptid=dept.deptid;
```

- Query 2:

```
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
dept.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>4000;
```

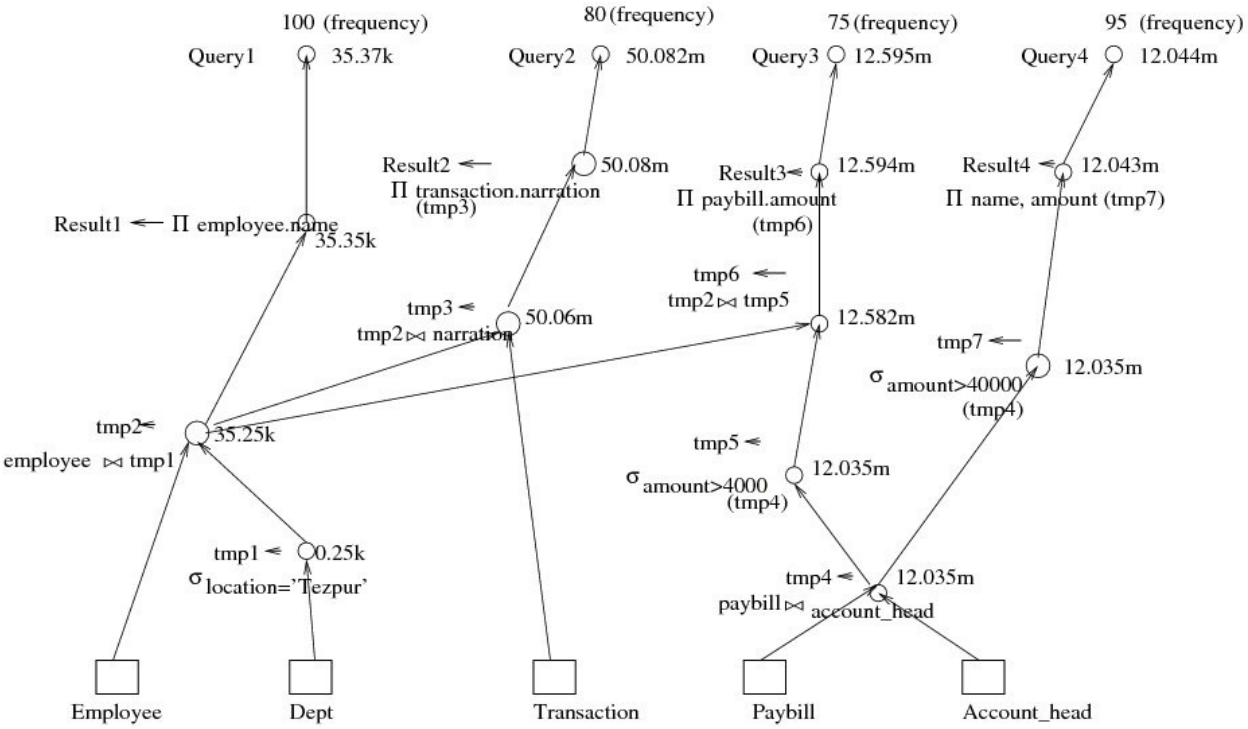

Figure 2-4: An MVPP graph

- 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 2-4. 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 2-4, the 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 $\forall v \in V$, $R(v)$ is the result relation generated by corresponding vertex v, then to determine a set of vertices in V, such that if $\forall v \in V$, $R(v)$ is materialized, the cost of query processing and view maintenance is minimal.

Yang et al. [7] designed a heuristic algorithm to select views for materializing by using MVPP DAG. Derakhshan et al. [8,9] applied Simulated Annealing algorithm using this representation in view selection for materializing problem. In [14], 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.

2.2.4 Query - Attribute matrix representation

This approach is based on detection of common sub-expressions within workload queries and finding the underlying views [10–13, 42]. A workload is syntactically analyzed to enumerate relevant candidate views. The warehouse's transaction logs are first analyzed over a certain time period and the most appropriate workload is considered for anticipating future workload of the system by the warehouse administrator. In [10,11,13], all the queries and the attributes in them are identified and then by analyzing 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 [10,11]. 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.

2.2.5 Associated issues in different representations

The pioneering view selection for materialization algorithms, the HRU-Greedy algorithm and the Polynomial Greedy Algorithm (PGA) [4, 6] 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 mostly used in stochastic 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* [43] represent multidimensional data cubes by decomposing the cubes into an indexed hierarchy of wavelet

Table 2.1: Representations used in view selection algorithms and associated issues

Notions	*View selection algorithms used*	*Associated issues*
Lattices	HRU-Greedy, PGA	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.
AND-OR graphs	Heuristic, GA, MA, PSO	Plan for multiple query processing is not considered and therefore sharing of materialized views by multiple queries are ignored.
MVPP graphs	Heuristic algorithm, Simulated Annealing(SA), Parallel Simulated Annealing(PSA)	Cost of building view graphs when the query processing plan changes and input workload is large.
Wavelet -Dwarf structure	Heuristic-greedy algorithm by physical re-designing of Data warehouse	To change physical design of data warehouse.
Query vs. attribute binary matrix	Clustering	Requirement of scanning through numerous subqueries and intermediary results.

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, [44] 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, analyze queries to find subexpressions or intermediate views within frequent queries 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 [11]. The various data structures and concepts used in different view selection algorithms and associated issues are presented in Table 2.1.

2.3 Existing View Selection Techniques

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

2.3.1 Greedy algorithmic approaches

Most of the heuristic approaches in materialized view selection are descendants of the view selection algorithm for materializing in data warehousing called the HRU-Greedy algorithm [4]. 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. Nadeua et al. [6] 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, but the scalability problem remains. In [3, 22, 33, 36] a greedy algorithm framework for the view selection problem using the AND-OR view graph is used. Yang et al. in [7] present a heuristic algorithm which can provide a feasible solution based on individual optimal query plans. In [45], 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.

2.3.1.1 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 [4]. It uses the hypercube lattice notion to represent the various views or GROUP-BY statements in queries as discussed in Section 2.2. 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 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.

Let us consider Figure 2-1. If $\{C2, C3\}$ is selected, the total benefit will

Algorithm 1: The HRU Greedy algorithm

Require: k number of candidate views $v_1, v_2, \cdots, v_k$ and the top-view
Ensure: The selected set of views S for materializing
1: $S \leftarrow$ {top-view}
2: **for** $i = 1$ to k **do**
3: Select a view v_i which is not in S, such that $B(v_i, S)$ is maximized
4: $S \leftarrow S \cup \{v_i\}$
5: **end for**
6: **Return** The set of views S

be $(6-0.8)M \times 4$ or $20.8M$, 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 $(0.8-0.01)M \times 2$, i.e., $0.79M \times 2$ 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.

Advantage: The HRU-Greedy algorithm needs to know the size of each of the views beforehand. And by knowing this, it computes the benefit of each and every combination of views if materialized. Therefore the most beneficial views can be determined for materializing. Hence this algorithm can yield the most optimum solution of the problem.

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

2.3.1.2 The Polynomial Greedy Algorithm (PGA)

In PGA model [6], each iteration of the HRU-Greedy algorithm is divided into a nomination phase and a selection phase to tame the exponential growth 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 benefiting node is selected for materializing and put into the list of selected views. 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.

Advantage: 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 reduce the complexity of the HRU-Greedy algorithm, the HRU algorithm is better than the PGA algorithm in terms of the quality of solutions [6]. The limitation of exponential growth of nodes for lattice representation of candidate views still remains.

2.3.1.3 AND-OR View Graph based greedy algorithm

Gupta et al. in [22] and [36] 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 expression AND-OR DAGs. The nodes in the final AND-OR view graph represent candidate views 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 A. 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 $q_1, q_2, \cdots, q_k$ and a quantity A, the view selection problem is to select a set of views or nodes $M = \{v_1, v_2, \cdots, v_m\}$, that minimizes $\tau(G, M)$ in Equation 2.1, where under the constraint $\sum_{v \in M} A_v \leq A$, A_v is the space occupied by the view v, f_q is query frequency and g_v is update frequency of

view v.

$$\tau(G, M) = \sum_{i=1}^{k} f_{q_i}.Q(q_i, M) + \sum_{i=1}^{m} g_{v_i}.UC(v_i, M) \tag{2.1}$$

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, ζ, E), where Q is the set of queries to be supported and ζ is a subset of all views V. An edge (q, σ) is in the set of edges E iff the query q can be answered using the views in the set σ and the cost associated with the edge is the cost of answering q using σ.

In AND-OR Greedy algorithm at every stage a connected sub-graph H of F_ζ 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 A.

Advantages: In [36], 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 [36]) is extended to the general AND-OR view graphs.

Limitations: The evaluation of the algorithm in terms of the quality of solutions is not provided. 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.

2.3.1.4 Optimal query execution plan and heuristic algorithm

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 [7]. 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 2.2. 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 $v \in M$, $C_a^q(v)$ and $C_m^r(v)$ 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.

$$C_{total}(v) = \sum_{q \in R} f_q.C_a^q(v) + \sum_{r \in L} f_u.C_m^r(v) \tag{2.2}$$

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 2.3. 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 2.4.

$$w(v) = \sum_{q \in O_v} f_q.C_a^q(v) - \sum_{r \in I_v} f_u(r).C_m^r(v) \tag{2.3}$$

$$C_s = \sum_{q \in O_v} \{f_q.(C_a^q(v) - \sum_{u \in s_v \cap M} C_a^q(u))\} - \sum_{r \in I_v} \{f_u(r).C_m^r(v)\} \tag{2.4}$$

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.

Advantages: 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 solution using 0-1 integer programming. Yang et al. in [7] 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 [9].

Algorithm 2: View selection using optimal query plan

Require: An MVPP graph
Ensure: A 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 sub-tree 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** set of views M

2.3.2 Stochastic algorithmic approaches

Stochastic 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 modeling errors. As the problem at hand is NP-hard, several heuristic and stochastic optimization methods have been proposed [8, 9, 37, 39–41, 46, 47].

2.3.2.1 Simulated Annealing (SA) algorithm based approach

Derakhshan et al. in [8, 9] 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 labeled 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 labeled 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.

Algorithm 3: Simulated annealing for selection of views to materialize

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

Define: Initial temperature T, terminating temperature T', space constraint C, maximum number of iteration I_{max}
Initialize a candidate solution string S such that it satisfies space constraint C
repeat
  for $I = 1$ to I_{max} **do**
    $S' \leftarrow perturb(S)$
    $E = cost(S)$
    $E' = cost(S')$
    if $(E' < E)$ or $(random() < e^{(E-E')/T})$ **then**
      if S' satisfies the constraint C **then**
        $S \leftarrow S'$
      end if
    end if
  end for
  $T = decrement(T)$
until $T > T'$
Return S

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. in [9] 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 [14] 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 [30].

Yuhang et al. [48] 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 premature 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 SA metropolis criterion in the variation process, the algorithm decides whether to accept the new antibody (candidate solution) for subsequent steps of SA or not. This process is repeated till it reaches the minimum temperature specified.In [48] it has been claimed that this hybrid algorithm has more chance of escaping from local optimum and reaching the global optimum compared to Genetic Algorithm (GA) and CSA.

Advantages: Experimental results as reported by [8] show that the cost of selected views is considerably better than ones obtained by the previously reported heuristics. By using SA, the cost of a selected set of materialized views comes down by up to 70% [9] than the cost obtained by genetic and heuristic algorithms. Also, in [9] 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 [48], 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.

2.3.2.2 Genetic Algorithm (GA) based approach

As the views selection for materializing in data warehouse is an NP-hard problem, Evolutionary Algorithms (EAs) such as GA is likely to provide efficient solutions [37]. To obtain better solutions from a large number of views taking into account view maintenance and query processing costs, GAs have been used [37–39]. 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 [39]. That is, the sets of candidate configurations (views and indexes) are referred to as genomes of the candidate population. The 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 to consider the other constraints of the problem and objective. 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 2.1. 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 [39]. In [34] an 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.

Advantages: 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 [39]. In [39], it has been presented that their solution, in speeding up materialized view selection, is better than the existing solutions in terms of expected run-time behavior 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 [39], 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.

2.3.2.3 Memetic Algorithmic (MA) model

The memetic algorithm (MA), first proposed by Moscato in [49], 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 [40], 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 2.1. 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. **Advantage:** With the model suggested by [40], 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 most of the heuristic algorithms and GA in all cases regardless of storage space [40].

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 [40]. However, the other drawbacks of GA remain in MA based approach.

2.3.2.4 Particle Swarm Optimization (PSO) in selecting views

The PSO technique has also been used in the materialized view selection problem [41]. Sun and Wang in [41] 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 view graph as in Equation 2.1. Like GA and MA, in PSO as presented in [41], 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 2.1. 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. A solution configuration or a solution set of materialized views x_i is changed to a new solution as expressed by Equation 2.5 using a velocity value $v_i(t+1)$. The velocity of each particle is modified according to the Equation 2.5, where, t is the iteration number, p_i is ith particle's personal best position, p_{gb} is global best fit position, $x_i(t)$ is the position of ith particle at iteration t, i_{max} is the maximum number of iterations and weight, $w_i = w_{max} - ((w_{max} - w_{min})/i_{max}) \times i$. c_1 and c_2 are two constants preferably equal to 2 and r_1 and r_2 are random variables in the range [0,1].

$$v_i(t+1) = w_i v_i(t) + c_1 r_1 (p_i - x_i(t)) + c_2 r_2 (p_{gb} - x_i(t)) \tag{2.5}$$

The position of each particle is modified according to the Equation 2.6.

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2.6}$$

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.

Advantage: 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 and Wang in [41] 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 [41] 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 [50].

2.3.2.5 Ant Colony Algorithm (ACA) for optimizing view selection

In [51–53], 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.

Advantages: In [53], 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 almost same [53].

Limitations: The solutions by ACA approach for view selection problem used in [51, 52] and [53] 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.

2.3.3 Data mining based approaches

Data mining techniques have also been used to handle the view selection for materialization problem [10–13]. In [12], a density-based view materialization algorithm is discussed using data cube lattice structure, view size, access frequency of the views, and support (frequency). In [10,11] and [13], clustering techniques are used to cluster similar queries by analyzing 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.

2.3.3.1 Clustering for materialized view selection

Aouiche et al. [10] 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 [11]. 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.

Advantages: 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 [11], 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 [10], 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.

2.3.3.2 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 $O(nlogn)$ [12]. 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. in [13] 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) [54]. 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.

Advantage: 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 in [10] 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.

Table 2.2: Stochastic algorithm based materialized view selection techniques and associated issues.

Techniques used	*Years*	*Experimental framework and data set used*	*Associated issues*
Genetic Algorithm (GA) [39]	2001	Randomly generated data and synthesized queries	No guaranteed convergence to global minimum and no proof of validity.
Simulated Annealing (SA) [8]	2006	TPC-D [55] benchmark data warehouse	Convergence is slower than other versions of SA in materialized view selection.
Multi-objective GA [26]	2006	World hydro logical data and four synthetic data sets	No validity proof on solutions obtained.
Parallel Simulated Annealing(PSA) [9]	2008	Data warehouse generated from real life production database, TPC-D	Dependency on initial temperature and iterations. It is designed for view selection problem as single objective optimization problem.
Particle Swarm Optimization (PSO) [41]	2009	TPC-D benchmark data warehouse	Premature convergence and getting trapped in local optima [50].

Table 2.2: Stochastic algorithm based materialized view selection techniques and associated issues.

Techniques used	*Years*	*Experimental framework and data set used*	*Associated issues*
Memetic Algorithm (MA) [40]	2009	TPC-D benchmark data warehouse	More number of functional evaluations and no proof of convergence.
Ant colony algorithm (ACA) [53]	2010	Randomly generated data warehouse	Largely depend on parameters.
Clonal selection based SA [48]	2010	TPC-D benchmark data warehouse	Number of functional evaluations and parameter selection.
Multi-objective SA [14]	2012	TPC-H benchmark data warehouse [15]	Filtration of significant solutions based on diversity, elitism etc. are not mentioned.

2.4 A Brief Discussion on Existing Approaches

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 EAs have been introduced. However, they have limitations as well.

GA-based 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

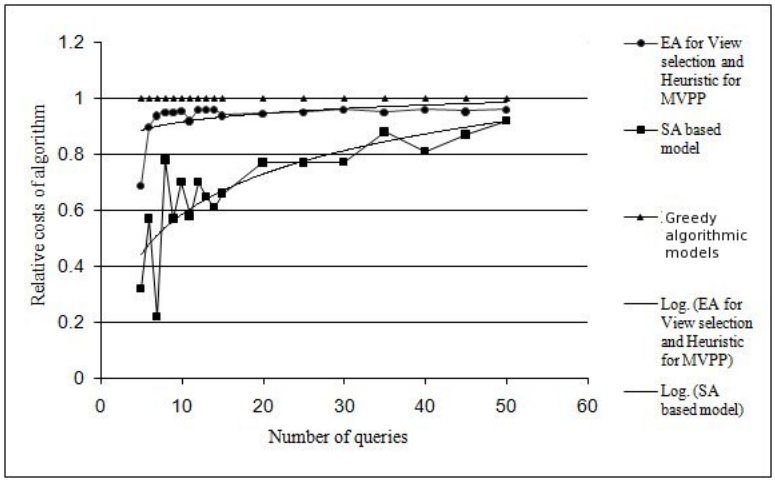

Figure 2-5: Relative costs of Heuristic, Evolutionary and Simulated Annealing algorithm in view selection using query processing plan graph representation.

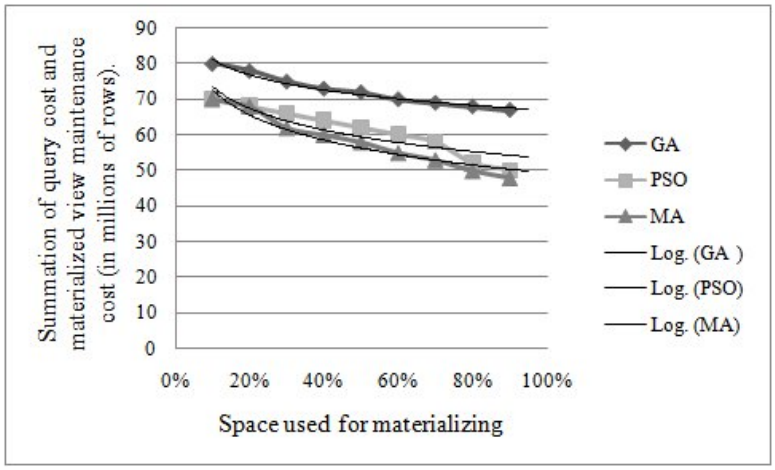

Figure 2-6: Comparison of GA, PSO and MA based view materialization models with respect to total query processing costs vs. space used by materialized views.

up to 70% less than the GA and other heuristic approaches in this problem [8, 9]. Another major limitation of the evolutionary approach is that it is hard to acquire good initial solutions, and therefore in the view selection problem, GA-based approaches converge slowly. It has been observed from results presented in surveyed literature that, SA [8] out performs Heuristic algorithmic approach [7] and EA (with heuristic view processing plan selection) approach [38] in case of query processing plan based view selection models as presented in Figure 2-5.

PSO and MA-based approaches may achieve better performance than GAs in the view selection for materializing in data warehouse [40, 41] as presented by a graph in Figure 2-6.

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 analyzed 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 views themselves 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.

2.4.1 Issues and challenges

The following issues and research challenges with implications to implementations in case of different approaches in handling the view selection problem for materializing in data warehouses have been identified.

i. **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 heuristic 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.

ii. **Data structure:** Heuristic view materialization techniques use the lattice representation of views. This makes it a non-polynomial complexity problem. Methods suggested to convert the conventional heuristic view selection techniques to polynomial complexity need a lot of pre-computation [6]. Heuristic techniques in the view selection problem use query processing plan graphs or AND-OR view graphs. Though most studies on the applicability of heuristic 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.

iii. **Cost model:** The HRU-greedy algorithm and the PGA 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 and randomized algorithmic approaches consider query frequency and view updating frequency as shown in Equation 2.1 or 2.2 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.

iv. **Parameter selection:** Solution quality for heuristic algorithms, including EAs, largely depends on the number of iterations or the number of generations specified. In SA approaches, the solution quality depends on parameters such as the initial temperature, the final temperature and the rate of temperature decrements. 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.

2.5 Discussion

In this survey, we have analyzed various techniques used in view selection for materialization in data warehousing. By analyzing 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.

For critical analysis of different techniques in any area, researchers and practitioners need a common protocol for performing experiments using standard data sets 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 data-set and benchmarking for evaluation. For extensive analysis of different approaches, it is expected that Transaction processing Performance Council

(TPC) will come-up with voluminous benchmark data-set [55], with a standard framework for experimentally evaluating these techniques for view selection for materialization problem.

It has been observed that the view selection problem for materializing in data warehouses is so far mostly handled by converting it into a single objective optimization problem of minimizing the summed up cost function values of different associated costs. But there are trade-offs to be considered among the costs. To address this, in next chapter the view selection problem is defined as multi-objective optimization problem for minimizing total analytical query processing cost of data warehouse by selecting a set of views for materializing within limited available memory space with minimized maintenance cost of the materialized views.