Multi-Objective Optimization in Selection of Views to Materialize in Data Warehouses

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

by

Rajib Goswami

Enrollment No. CSP-10-001

Registration No. 003 of 2015

Department of Computer Science and Engineering

School of Engineering, Tezpur University

Tezpur - 784028, Assam, India

November, 2015

Dedicated to

my parents

Abstract

Materialized views in Data Warehouse is a promising solution to speed up the analytical processing of huge volume of historical data for running decision support applications to detect business trends by mining the data. By materializing or storing information organized as a set of views within a data warehouse from different production databases avoid the accessing of the original data sources and thereby it increases the efficiency of the warehousing queries. The large number of computation on huge volume of data, and space required for materialization of the views make it impractical to materialize all possible views. Therefore, there is a need for selecting an appropriate set of views to materialize. This selection of views is commonly referred to as the view selection for materialization problem. Any change in the source data is to be reflected in the materialized views and hence the materialized views are to be maintained by synchronizing with the source data. Hence the trade-offs between query efficiency, materialized view maintenance cost, number of materialized views and/or their sizes make the view selection problem as one of the most challenging problems in data warehousing. The classic approaches presented in literature for handling this problem was deterministic algorithms with heuristic greedy approaches. With increase in dimensions of the data warehouses, the solution space increases exponentially and deterministic algorithm becomes un-scalable. The problem is NP-hard. Therefore, stochastic and evolutionary algorithms are found to be most suitable for selecting a set of views for materializing with optimum associated costs and benefits. In most of the recent works on this problem Genetic Algorithm (GA) and other randomized algorithms like Simulated Annealing (SA) have been applied for optimizing the costs of materialized views. Unfortunately, these approaches could not handle the issues and challenges related to this problem beyond a certain limit. The majority of the approaches consider the problem as mere single objective optimization problem to optimize summed up cost of all associated costs of materialized views in relational model based data warehouses ignoring trade-offs between different costs. Also they do not consider using an adaptive system for selecting views for materializing in case of data warehousing with very large semi-structured databases which support collective data

types like in case of Big data framework based data warehousing and data processing paradigm using Distributed File System at cluster of low cost commodity hardware.

We begin by defining the materialized view selection problem as a multi-objective optimization problem for optimizing query processing cost and materialized view maintenance cost with constraint on availability of space for materializing in conventional RDBMS based data warehousing. It addresses the issue of considering trade-offs between inter related costs that are to be optimized. An already established input method using multiple view processing plan based representation of the problem has been used for implementation. Though the classic Multi-Objective Genetic Algorithm has been already applied in this problem successfully, it has been observed that Differential Evolution (DE) algorithm outperforms Genetic Algorithms on many numerical single objective optimization problem. Therefore a version of the multi-objective Differential Evolution algorithm has been designed for adapting with binary encoded solution population for implementing in view selection problem. It also addresses the problem of loss of significant solutions while filtering out representative solutions out of large number of non dominating solutions of the multi-objective optimization problem.

With the advent of Big data and MapReduce programming paradigm, next we investigate on view selection problem for materializing in Big data framework. The problem has been defined as a multi-objective optimization problem for minimizing (1) the total query processing MapReduce cost, (2) materialized view maintenance MapReduce cost and (3) the number of views selected for materializing with constraint on minimum size of materialized views. The *Forma analysis* based multi-objective DE for binary encoded data, that has been already used in materialized view selection for conventional data warehousing, termed as MODE-BE, is modified and applied in designing a view selection and recommendation system for materializing in Distributed File System data warehouse framework by promoting diversity of solutions in solution vector space. The popular elitist multi-objective GA termed as NSGA-II is also applied on this problem to analyze performances between NSGA-II based systems and MODE-BE based recommendation system in view selection for materializing in Big data management framework.

Finally, For comparative analysis of performances of Multi-Objective Evolutionary Algorithms (MOEAs) with Simulated Annealing (SA) based techniques in materialized view selection, the basic Archived Multi-objective Simulated Annealing (AMOSA) algorithm is customized for applying in materialized view selection in MapReduce based distributed file system framework. So far available data warehousing technologies in Big data framework do not support materialized view. It is expected that this dissertation will be useful for future research in designing

and developing analytical processing applications for Big data warehouses.

Keywords: Data warehouse, View materialization, Materialized view selection, Query processing cost, Materialized view maintenance, Multi-objective optimization, Pareto front, Genetic Algorithm, Differential Evolution algorithm, Big data, MapReduce, Multi-objective Simulated Annealing (MOSA), Archived Multi-objective Simulated Annealing (AMOSA).

Declaration

I certify that

- The work contained in the dissertation is original and has been done by myself under the general supervision of my supervisor.
- The work has not been submitted to any other institute for any degree or diploma.
- I have followed the guidelines provided by Tezpur University in writing the thesis.
- I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the university.
- Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the dissertation and giving their details in the references.

Rajib Goswami

Department of Computer Science & Engineering
Tezpur University
Napaam, Tezpur- 784028, Assam, India.

Prof. Malay Ananda Dutta
Professor

Phone: 03712-275352
Fax: 03712-267005
E-Mail : malay@tezu.ernet.in

Certificate

This is to certify that the thesis entitled **"Multi-Objective Optimization in Selection of Views to Materialize in Data Warehouses"** submitted to Tezpur University in the Department of Computer Science and Engineering under the School of Engineering in partial fulfillment of the award of the degree of Doctor of Philosophy in Computer Science and Engineering is a record of research work carried out by **Rajib Goswami** under my supervision and guidance.

All helps received by him from various sources have been duly acknowledged. No part of this thesis has been submitted elsewhere for award of any other degree.

Signature of Supervisor
(Malay Ananda Dutta)
Professor
Department of Computer Science and Engineering
Tezpur University
Assam, India-784028

Certificate

This is to certify that the thesis entitled **"Multi-Objective Optimization in Selection of Views to Materialize in Data Warehouses"** submitted by **Rajib Goswami** to Tezpur University in the Department of Computer Science and Engineering under the School of Engineering in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy in Computer Science and Engineering has been examined by us on ..... and found to be satisfactory.

The Committee recommends for award of the degree of Doctor of Philosophy.

Signature of Principal Supervisor

Signature of External Examiner

Acknowledgment

It's great pleasure for me to express my gratitude to all of them who have provided me guidance and support in bringing out successful completion of my doctoral program at Tezpur University. It would not have been possible for me to achieve success with my effort alone. There were a lot of people who extended their supportive hands towards me in the way of making my work a success. I would like to thank everyone who supported and assisted me while carrying out this work at Tezpur University.

First of all, I would like to thank to my supervisor Prof. Malay Ananda Dutta for his constant support, trust, valuable feedback, encouragement, innumerable advice and overall guidance. He gave freedom to pursue my ideas and work at my own pace, and was always available to discuss various problems on the way. I enjoyed spending these years with him both at work and otherwise. I would also like to extend my sincere thanks to Prof. D.K Bhattacharyya for his encouragement, feedback and valuable advice which have provided a good basis for completion of my research work.

I would like to acknowledge my sincere thanks and gratitude to Dr. Bhogeswar Bora and all the doctoral committee members of my research for their valuable comments and suggestions. I would also like to place on record the help and support received from the members of the faculty, Department of Computer Science and Engineering time to time with a special mention of a few - Prof. Nityananda Sarma, Prof. Shyamanta M. Hazarika, Dr. Sarat Saharia, Prof. Smriti Kr. Sinha, Prof. Utpal Sharma, and Prof. Dilip Kr. Saikia. I extend my sincere thanks to other members of the department specially Sanjib Kr. Deka, S. I. Singh, Dr. Debasish Das, Debojit Boro, Sanghamitra Nath, Dr. Arindam Karmakar, Dr. P Dutta, Dr. Bhabesh Nath and the non-teaching staff of the Department, for their generous help in various ways towards the completion of the work. I wish to place on record my sincere thanks to the members of my thesis review committee and the anonymous reviewers for their precious comments and feedback.

I am highly grateful to Dr. Biren Das, Registrar, Tezpur University, for his help, support and encouragement while carrying out this work.

A very special mention for my parents and my uncle who have been constantly encouraging and supporting me in every walk of my life. I will ever remain grateful to my sister, Rip, Riyam and in-laws for their unconditional love and

support.

I am deeply indebted to my wife- *Krishna* and my son- *Dapon*, whose unending help, inspiration, understanding, and love made me pursue my dreams.

Finally, I would like to thank all those who have directly or indirectly helped me in different capacities to complete my research work.

Rajib Goswami

Contents

List of Figures

List of Tables

List of Algorithms

Glossary of Terms

ACA	Ant Colony Algorithm
AMOSA	Archived Multi-objective Simulated Annealing
AMOSA-MVS	Archived Multi-objective Simulated Annealing for Materialized View Selection
AST	Abstract syntax tree
CPU	Central Processing Unit
CSA	Clonal Selection Algorithm
DAG	Directed Acyclic Graph
DE	Differential Evolution
DEMO^{NS-II}	Differential Evolution based variants of NSGA-II
DFS	Distributed File System
DIMMQ	Discardable In-Memory Materialized Query
DTLZ	K. Deb, L. Thiele, M. Laumanns and E. Zitzler defined test problems for multi-objective optimization techniques.
EA	Evolutionary Algorithm
GA	Genetic Algorithm
GDE3	Generalized Differential Evolution - 3
HDFS	Hadoop Distributed File System
HDP	Hortonworks Data Platform
HRU	Venky Harinarayan, Anand Rajaraman and Jeffrey D. Ullman, Stanford University
MA	Memetic Algorithm
MB	Mega Byte
MODE-BE	Multi-Objective Differential Evolution for Binary Encoded data
MOEA	Multi-Objective Evolutionary Algorithm
MOGA	Multi-Objective Genetic Algorithm
MOO	Multi-Objective Optimization
MOSA	Multi-objective Simulated Annealing
MVPP	Multiple View Processing Plan
NPGA	Niched Pareto Genetic Algorithm
NP-hard	Non-deterministic polynomial-time hard
NSGA-II	Non-dominated Sorting Genetic Algorithm-II
OLAP	On-Line Analytical Processing
OPTICS	Ordering Points to Identify Clustering

	Structure
PB	Petabyte
PGA	Polynomial Greedy Algorithm
PSA	Parallel Simulated Annealing
PSO	Particle Swarm Optimization
RDBMS	Relational Database Management System
RDD	Resilient Distributed Data-set (Spark's)
SA	Simulated Annealing
SMC	Simple Matching Coefficient
SQL	Structured Query Language
SSD	Solid-state drives
TB	Terabyte
TPC	Transaction processing Performance Council
TPC-H	Transaction processing Performance Council (Benchmark version H)
ZDT	Zitzler, Deb and Thiele's test problem for evaluating Evolutionary Multi-objective optimization techniques

Symbols and Notations

$\mathbb{B}$	Set of all truth values
$\mathbb{E}(S)$	The set of all equivalence relations over a given set S
Δdom	Amount of domination between solutions
$\Delta dom_{average}$	Average domination between solutions
Δdom_{min}	Minimum amount of domination between solutions
δE	Energy difference between solutions generated in Simulated Annealing algorithm
$\equiv$	Equivalent
γ	Convergence metric of multi-objective optimization technique
Γ	A real valued constant
μMax	Mean of maximum distances between each solution vectors w.r.t all other solution vectors obtained
$\not\prec$	Does not dominate
$\prec$	Dominates
σMax	Standard deviation of maximum distances between each solution vectors w.r.t all other solution vectors obtained
$\sim$	A relation
$\triangleq$	Defined to be equal to
Ξ	The set of *formae* or equivalence classes
T	System temperature in Simulated Annealing process
A	Availability of memory space for materializing views
$A_{V'}$	Space required for materializing the set of sub-queries V'
A_M	Space required for materializing the set of views M
$B(v, S)$	Total benefit of materializing view v if S is the set of view selected at an iteration of HRU-greedy algorithm
$C(v)$	Cost of processing view v
$C_a^q(v)$	Cost of accessing query q when view or vertex v of an MVPP graph is materialized
C_M^Q	The total cost of responding queries Q when the set of views M is materialized
$C_m^r(v)$	Cost of maintaining a view or vertex v due to change in the base relation r
$C_{total}(v)$	Total processing cost of view v in an MVPP graph.
$C_{V'}^Q$	The total MapReduce cost of processing set of

	queries Q when a set of sub-queries or sub-expressions as views V' is materialized
CR	Real valued cross-over probability constant in the range[0,1]
d	Number of dimensions of a data cube
D_Ψ	The set of different parameters between equivalence relations using basis Ψ .
$DE/x/y/z$	Version of Differential Evolution with x vector to be mutated, y specifying whether *random* or *best* and z specifies cross-over scheme
F	Real valued amplification vector in the range [0,1] in DE
$\tilde{\mathscr{F}}$	The current estimate of Pareto front
$\mathscr{F}$	Pareto front
f_q	Frequency of a query q
f_u	Updating frequency of base relations of an MVPP graph
f_v	Frequency of queries on view v of an AND-OR view graph
g	Generation number in an evolutionary process
g_{max}	Maximum number of generations in an evolutionary process
g_v	Updating frequency of a view v of an AND-OR view graph
L_i	Number of levels of hierarchies associated with dimension i of data warehouse
$\mathbf{M}$	Objectives of a multi-objective optimization problem
M_{de}	Formae basis based DE mutation operator template
$\mathbb{P}$	Probability (function value)
$\mathscr{P}$	Set of Pareto optimal solutions
$Q(v, M)$	Cost of answering sub-query v of an AND-OR view graph when set of views M is materialized
$Q_G(M)$	Total query processing cost of MVPP graph G, if the set of views M is materialized
$R(v)$	Resultant relation corresponding to vertex v in an MVPP graph
R_v	Cost incurred in reading the materialized view v of an AND-OR view graph
$randI()$	Random integer index generator
$randR()$	Evaluation of a uniform random number generator in the range [0,1]
$\mathbf{S}$	Set of vectors
T_q	Query processing tree for query q
$\tau(G, M)$	Processing cost of AND-OR view graph G when the set of views M is materialized
$U(M)$	The total updating cost of set of materialized views M
$U_G(M)$	Updating cost of MVPP graph G when set of views M is materialized
$UC(v, M)$	Maintenance cost of a sub-query or view v when the set of views M is materialized

Chapter 1

Introduction

We are drowning in an ocean of data. Every minute Email users send over 200 million messages, Twitter users tweet nearly 300,000 times, YouTube users upload 72 hours of new video content, Instagram users post nearly 220,000 new photos and Amazon generates over $80,000 in online sales[1]. Facebook stores more than 300 petabyte of data in their Hive technology based data warehouse with an incoming rate of around 600 terabyte of data every minute which is increasing three times every year[2]. During the month of August 2015, there were 1.49 billion daily users of Facebook[3].

Historical data are manipulated for deriving data for analysis by applying analytical operations such as ratios, totals, trends, allocations across dimensions and across hierarchical levels to plan and forecast for better productivity and earnings by defining gaps and deciding on new strategies. These analytical processing do not require any transaction processing. Therefore these historical data are kept separately in *data warehouses*. Inmon in [1] defines data warehouse as Definition 1 below.

Definition 1. *A **data warehouse** is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process.*

The results of some of the very expensive analytical operations on data warehouse may be saved for using them by other similar subsequent analytical processing on these huge volume of data. Therefore, data warehouse views are used to pre-compute and store aggregated data of a central theme such as the sum of sales, grouped by different classifications of the theme. For example, total sales from January to June in a year, total quarterly sales in a particular region of a country in a specific year, total sales by a dealer etc.. The views are often referred to as summaries in data warehouses, because they are used to store sum-

[1]Guneltus, S. in: The data explosion in 2014 minute by minute infographic, 2014. URL http://aci.info/

[2]As posted in "https://code.facebook.com" in October 2015

[3]The US edition of The Times of India, September 13, 2015.

marized data. They are also used to pre-compute database joins with or without aggregations.

Definition 2. *In database theory, a **view** is the result set of a stored query on the data, which the database users can query just as they would in a persistent database collection object.*

Materialized or stored views as defined in Definition 3 below are primarily used to eliminate the overhead associated with expensive joins and aggregations for a number of selected queries when they are executed again and again.

Definition 3. *In computing, a **materialized view** is a database object that contains the results of a query or a portion of a query.*

Materialized views are also can be used to replicate data at distributed sites and to synchronize updates at distributed sites with conflict resolution methods for distributed computing. Again materialized views may be used as replicas to provide local access to data that otherwise would have to be accessed from remote sites. In mobile computing, materialized views can be used to download a subset of data from central servers to mobile clients with periodic updates between the mobile clients and the central servers.

Large amounts of storage space are required for materializing the views. Materializing view also incurs maintenance costs for periodic refreshing and updates. Therefore it is not feasible to save all views of a data warehouse. And hence, an optimum set of views are selected for materializing.

In this thesis an analysis of applying few randomized search methods, customized for selecting views for materializing, is presented.

This chapter focuses on introducing the use of materialized views in data warehouse and the materialized view selection problem as multi-objective optimization problem. The objectives of this research work, contributions and organization of this thesis are also presented in this chapter.

1.1 Data Warehousing and Materialized Views

Data warehouse consolidates data in multidimensional space by integrating data from multiple heterogeneous data sources. Data warehouses are used for interactive analysis of multidimensional data of different levels of granularity for making strategic decisions. They are maintained separately from operational database as they are used only for analysis for using as support system for management's decision making process and do not require any transaction processing. The basic operations of data analysis on data warehouse are analytical processing by On Line Analytical Processing (OLAP) system. Thus Data warehouses are historical operational data for analytical processing and data mining [2–4]. In case of operational database, basic operation is transactional processing. Therefore data in

operational database may change for every transactional operation performed on them. While making strategic decision by analytical processing, the database is considered to be unchanged during the considered period. Therefore, from multiple transactional databases data are uploaded periodically to a data warehouse for analytical processing later on. For analytical processing, aggregated or consolidated measures of some selected entities of the transactional information system over a large historical period are useful which are not very important in case of transactional database. Therefore the data warehouses are maintained separately from transactional database to maintain the performance of both the systems. Data warehouses are designed considering a central theme of measure on different related entities. The entities that are measured and used by OLAP functions for analyzing the central theme are termed as dimensions of the data warehouse. The number of enquiries may be the central theme of a data warehouse for Example 1 below.

Example 1. *The Indian Railways run around 7000 passenger trains daily. There are 18 different types of trains or services for passengers with 10 different classes of services in them connecting 7112 stations managed by 17 different administrative zones. Indian Railway carries around 13 million passengers every day. Most of the passengers make enquiry regarding availability of different services before booking a service*[4].

In Example 1, the enquiries are made on real time transactional database to get the latest updated information. On the other hand for strategic planning, the users needs may be analyzed by studying different enquiries triggered to the system without any transactional updating. All the queries made over a time period to the system may be recorded and loaded to a separate data warehouse. For strategic decisions, different OLAP operations are to be performed on this data warehouse of enquiries made regarding different services or entities at different times of the organization. Hence, aggregated values on number of queries on different dimensions or entities with different granularity may be recorded in the data warehouse for quicker response of OLAP queries.

1.1.1 Multidimensional data model

Data warehouse and OLAP functions are conceptualized based on a multidimensional data model. This multidimensional model is termed as *data cube*. The data cube concept visualizes data modeled in multiple *dimensions*. These dimensions are the entities, of which measures are to be recorded for analysis [2–4]. For our Example 1, we may design a data warehouse of central theme *enquiry* to keep number of queries made with respect to the dimensions zones, types of trains and time. In Relational Database Management System (RDBMS) based data warehouse, for each dimension, a table may be associated. These tables are called dimension tables. Numerical measures on different dimensions on the main subject of interest or central theme are called the *facts* of the data warehouse and

[4]http://www.indianrailways.gov.in/

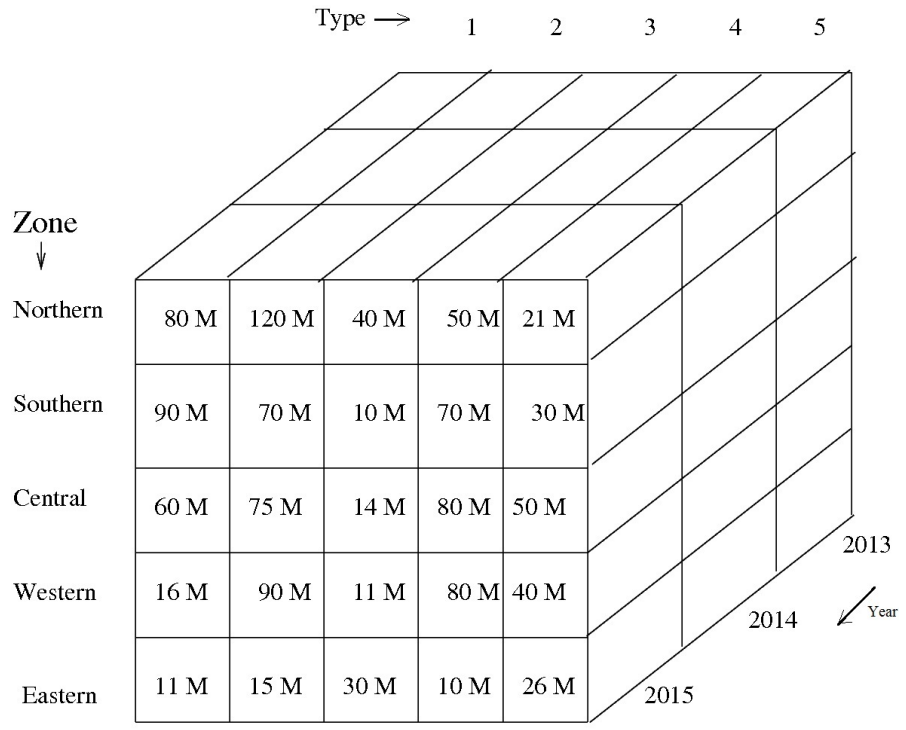

Figure 1-1: Data cube representing a data warehouse.

are represented by a central *fact table*. In case of Example 1, number of queries made with respect to different zones, stations, routes, types of services and time of inquiries are the facts that are to be recorded in a fact table. The data cube representing the data warehouse in Example 1 may be represented as Figure 1-1. A cube in geometry is a 3-D geometric structure, however in case of data warehousing, the data cube is of n-dimensional for n number of dimensions on which facts or measures are recorded. As depicted in Figure 1-1, *cuboid*s for each of the subsets of given dimensions may be generated. Different degrees of summarized measures or data are stored in these cuboids. Thus, *lattice* of cuboids providing data at a different level of aggregation or "group by" is formed. These lattices of cuboids are termed as data cubes. Figure 1-2 represents a lattice of cuboids as a data cube for dimensions type, zone and year. The cuboid keeping lowest level of summarized value is called the base cuboid and cuboid containing highest level of summarisation is called the apex cuboid. The value of a measure in a data cube is a numerical function evaluated for a given point by aggregating the data corresponding to the specific dimension and value pairs. Some example of aggregating functions are sum(), min(), max() and count(). Again a dimension may have several levels of concept hierarchies, e.g. the time dimension based on the attributes day, week, month, quarter and year on which aggregated values are to be kept (see Figure 1-3). For implementing data cube concept of data warehousing in entity-relationship data model, three types of modeling paradigm called star-schema, snow-flake schema and fast constellation schema are used [2,4]. These schema basically contain a central fact table containing the primary-keys of other dimension tables as foreign key and related measures or values. The dimension tables contain a primary key field and the hierarchical data. For ex-

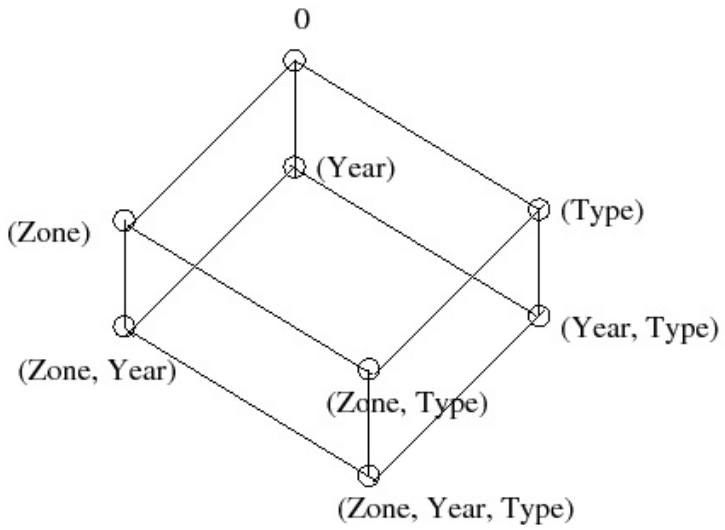

Figure 1-2: A lattice of cuboids

ample the fields in a fact table may be time_key, service_type_key, location_key and number_of_inquiries. Similarly dimension table time may contain fields like time_key, time_of_day, day, day_of_the_week, month, quarter, year. Mainly four types of OLAP operations are performed on multidimensional data model based data warehouses. The OLAP operations are termed as *Roll-up, Drill-down, Slice and dice* and *pivot (rotate)* [2]. The roll-up is performed for aggregating values reducing one or more dimensions from the data cube like representing number of queries about a service in a period of time by summing up values of all different zones where the zone dimension is removed. The Drill-down is the reverse operation of roll-up. That is, by drill-down, data in a new dimension are added by evaluating and exploding the aggregated values in the new dimension. The slice operation is selection of one dimensional data of a cube and the dice operation is for generating a sub-cube by performing a selection on two or more dimensions. The pivot is a data visualization operation by rotating the data axes to present an alternative view of the data in the data cube.

1.1.2 Aggregations of data as data warehouse views

The main design consideration of data cube based multidimensional data warehouse is efficient computation of aggregations of data across different dimensions. In SQL implementation of multidimensional data warehouse, the aggregations across dimensions are performed by group-by clause of SQL. Thus a cuboid is represented by an SQL group-by clause. Thus a set of SQL group-by forms a lattice of cuboids or data cube [2–4]. An example SQL implementation of data cube is presented below.

Example 2. *SQL implementation of a data cube*

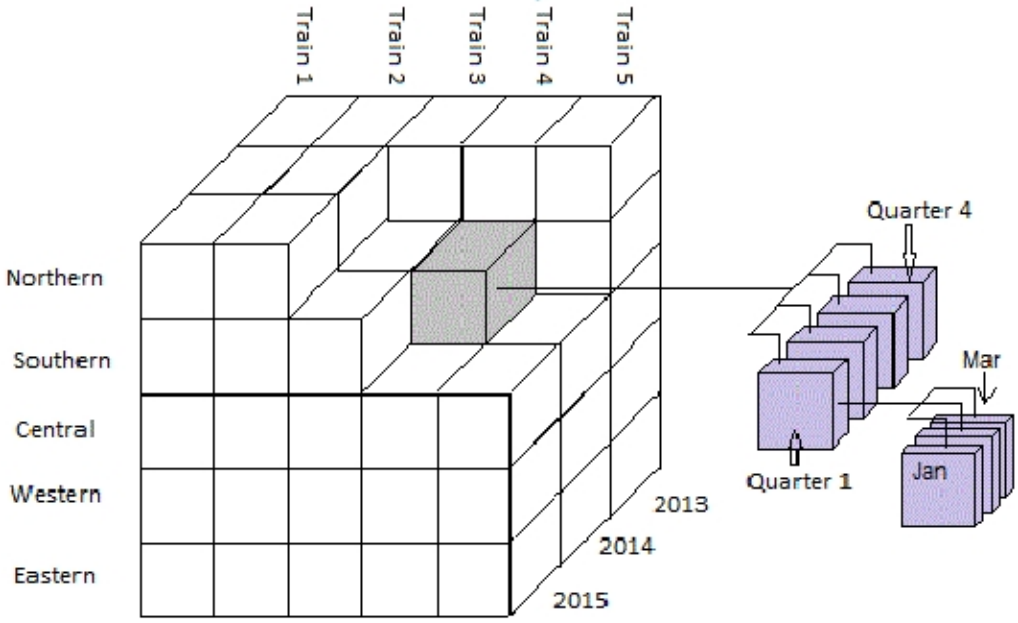

Figure 1-3: Representation of hierarchies in Data cube

```
SELECT s.time_key, s.service_type_key, s.location_in_zone_key,
SUM(s.numb_of_enquiry)
FROM enqiry s, time m, service_type t, zone z,
WHERE s.time_key=m.time_key
AND s.service_type_key=t.service_type_key
AND s.location_in_zone_key=z.location_in_zone_key
GROUP BY s.time_key, s.service_type_key, s.location_in_zone_key;
```

For three dimensions there may 2^3=8 number of possible "group-by"s like (zone, type, year), (zone, type), (zone, year), (type, year), (zone), (type), (year), (). Thus for n-dimensional data cube, there will be total 2^n number of cuboids. Again many of the dimensions have hierarchies. So, for n-dimensional data cube, $\Pi_{i=1}^{n}(L_i+1)$ number of cuboids can be generated, where L_i represents the number of levels or hierarchies associated with a dimension i. (L_i+1) is used to include the top level i.e group-by all [2]. Analytical processing on data warehouse accesses different cuboids for responding decision support queries. Therefore, all or at least some of the cuboids in a data cube may be computed in advance during an analytical processing session to make query response faster. These pre-computed aggregations are termed as data warehouse **views**.

1.1.3 Materialized views for efficient computation of data cubes

Some data warehouse views may be generated frequently in different analytical processing sessions by some frequent queries in a period of time in the past on the data warehouse. Therefore these views may be saved or stored in the data warehouse as *materialized views* to avoid redundant computations while responding OLAP queries. Thus materialized views make query response significantly faster. For large number of dimensions, conceptual hierarchies in dimensions, and their cardinality, the storage space requirement for materializing these views may exceed the size of the data warehouse as many of the group-by's size may exceed the size of the input relation or base table [2,4]. Again, the data warehouses are to be periodically updated with corresponding operational database. Whenever there is a change in the data warehouse, the materialized views are also to be updated [3]. Therefore it is not realistic to materialize all of the views [3,4]. The computation cost of aggregations while responding OLAP queries may be substantially reduced by accessing materialized views. But substantial overhead for materializing these views do not encourage materialized views. In [4], a greedy algorithm referred as *HRU Greedy algorithm* has been proposed to determine which cuboids should be precomputed for materializing.

1.1.4 Materializing views in RDBMS technology

Most of the popular RDBMS technologies have incorporated the materialized view feature as a component in their data warehousing platforms, where SQL query can be used to materialize view for improving query response time. In RDBMS based data warehousing environment, the schema could be a star-schema, snow-flake schema or fast constellation schema without any restriction on which schema is to be used. Before creating a materialized view, the first step is to review the schema and identify the dimensions. A dimension here is an object which defines a hierarchical relationships between columns, where all the columns do not have to come from the same table [5]. Example 3 below defines a time dimension, which contains a hierarchy defining the relationship between a day, month, quarter and year in Oracle Database 10g Release 2 for a Snowflake schema of a data warehouse.

Example 3. *An SQL statement in Oracle Database 10g Release 2 to create Time dimension*

```
CREATE DIMENSION times_dim
LEVEL day IS TIMES.TIME_ID
LEVEL month IS TIMES.CALENDAR_MONTH_DESC
LEVEL quarter IS TIMES.CALENDAR_QUARTER_DESC
LEVEL year IS TIMES.CALENDAR_YEAR
HIERARCHY cal_rollup
(day CHILD OF month CHILD OF quarter CHILD OF year)
ATTRIBUTE day DETERMINES
```

```
(day_number_in_week, day_name, day_number_in_month,
calendar_week_number)
ATTRIBUTE month DETERMINES
(calendar_month_desc, calendar_month_number, calendar_month_name,
days_in_cal_month, end_of_cal_month)
ATTRIBUTE quarter DETERMINES
(calendar_quarter_desc, calendar_quarter_number, days_in_cal_quarter,
end_of_cal_quarter)
ATTRIBUTE year DETERMINES
(calendar_year, days_in_cal_year, end_of_cal_year);
```

Once the dimensions have been defined, the materialized views can be created using SQL statements. A materialized view definition in Oracle Database 10g Release 2 can include aggregation, such as SUM, MIN, MAX, AVG, COUNT(*), COUNT(x), COUNT(DISTINCT), VARIANCE, STDDEV, and a GROUP BY clause of SQL with one or more tables joined together.These views may be indexed and partitioned where basic DDL operations like CREATE, ALTER, and DROP may also be applied [5]. A materialized view is created using the CREATE MATERIALIZED VIEW statement of SQL. Example 4 below illustrates the creation of a materialized view called number_of_enquiries_mv that computes the sum of enquiries by time and service_ type_name. The SELECT list in CREATE MATERIALIZED VIEW statement must contain the entire GROUP BY columns.

Example 4. *SQL Statement to create Materialized View*

```
CREATE MATERIALIZED VIEW  number_of_enquiries_mv
PCTFREE 0 STORAGE (initial 8k next 8k pctincrease 0)
BUILD IMMEDIATE
REFRESH  FAST ON DEMAND ENABLE QUERY REWRITE
AS
SELECT time_id, service_type_name, SUM(no_of_enquiries)
AS sum_of_enquiries, COUNT(no_of_enquiries) AS
record_count_of_enquiries,
COUNT(*)  AS cnt
FROM enquiries c, types_of_services p
WHERE c.service_id = p.service_id
GROUP BY time_id, service_type_name;
```

1.1.5 Common SQL sub-expressions of SQL queries as materialized views

It has been observed that, in one hand, no materialization of pre-computed cuboids leads to expensive computing of multidimensional aggregations, and on the other

hand, pre-computing all the cuboids and materializing requires huge memory space and to incur maintenance costs. Therefore, optimum selective materialization is suggested. But clear guidelines for determining which views are to be materialized is not defined. In basic multidimensional data model of data warehousing, computing an *iceberg cube* was suggested, which is a data cube storing only those cube cells whose aggregate value is above some minimum support threshold [2]. In [4,6] authors present greedy algorithms referred as *HRU Greedy algorithm* and *Polynomial Greedy Algorithm* (PGA) to determine which cuboids should be pre-computed for materializing.

The algorithms presented in [4, 6] are to select views to materialize considering only queries with aggregate functions involved for OLAP applications but not to deal with general SQL queries that include select, project, join and aggregation operations. Most of the recent works on selecting views for materializing consider common sub-expressions among multiple queries executed frequently in analytical processing sessions or periods on data warehouse for efficient multiple query execution [7–14]. This approach of considering shared common sub-expressions of queries on data warehouse as views for materializing is thus basically outcome of the problem of selecting multiple query optimization plan [7]. Multiple-Query optimization means finding an optimal execution plan for multiple queries executed by sharing some temporary results from some common sub expressions so that the total query processing cost of all the queries are minimum. But multiple-query execution cost by sharing results of intermediate sub-expressions as views depend on how the base relations are defined and how the intermediate select, join and project operations are defined. In case of very less number of base tables an optimum query execution plan by sharing multiple number of sub-expressions or intermediate relations as views may be easier but in case of larger number of considered queries on large number of base table or relations it becomes a complex problem of deciding how the intermediate results are to be shared for minimizing the total cost of execution of the considered queries. In [7], the authors design an algorithm for generating multiple view processing plan (MVPP) for optimizing total query processing cost of multiple queries by connecting or sharing pre-identified common sub-expressions of the queries as views. By SQL queries as presented in Example 5 and 6, how sharing of result of intermediate sub-expressions of queries as materialized views can reduce total query execution cost can be shown. These two SQL queries are designed for version H benchmark database of Transaction Processing Performance Council [15].

Example 5. *A query on TPC-H benchmark database*

```
SELECT s_acctbal, s_name, n_name, p_partkey,
p_mfgr, s_address, s_phone, s_comment
FROM  part, supplier, partsupp, nation, region
WHERE p_partkey=ps_partkey AND s_suppkey=ps_suppkey
AND p_size=:1 AND p_type like '%:2'
AND s_nationkey=n_nationkey
```

```
AND n_regionkey=r_regionkey AND r_name='MIDDLE EAST'
AND  ps_supplycost=(
SELECT MIN(ps_supplycost) FROM partsupp, supplier, nation,
region WHERE p_partkey=ps_partkey AND s_suppkey=ps_suppkey AND
s_nationkey=n_nationkey AND n_regionkey=r_regionkey AND
r_name='MIDDLE EAST')
ORDER BY s_acctbal, n_name, s_name, p_partkey;
```

Example 6. *A query on TPC-H benchmark database*

```
SELECT s_acctbal, s_name, n_name, p_partkey,
p_mfgr, s_address, s_phone, s_comment
FROM  part, supplier, partsupp, nation
WHERE  p_partkey=ps_partkey AND s_suppkey=ps_suppkey
AND p_size=:1 AND  p_type like '%:2'
AND s_nationkey=n_nationkey
AND n_name=':1'
ORDER BY  s_acctbal, s_name, p_partkey;
```

Though both the queries in Example 5 and 6 use different join conditions, both of them share some common sub expressions like accessing part table of 26260520 number of rows for selecting parts of "p_size=':1'" and "p_type like '%:2'" returning 26260 number of rows. If in a considered period, the frequency of execution of query in Example 5 is 20 and that of Example 6 is 10 then total $20 \times 26260520 + 10 \times 26260520$ number of rows to be accessed. By materializing or saving the intermediate result of the SELECT operation for "p_size=':1'" and "p_type like '%:2'" from the part table, these queries can be designed to access only $20 \times 26260 + 10 \times 26260$ number of rows. If number of rows to be processed or accessed is considered as query processing cost, then it is observed that there is a large amount of savings in terms of query processing cost by materializing the result of this SELECT operation as a database table called view. But during this period, the saved results or materialized views may have to be refreshed or updated. Generally the updating frequencies are much less than that of accessing the materialized views for processing the queries. During the considered period of these queries, let the materialized views are to be updated two times. Then 2×26260520 rows to be processed for maintaining the materialized view. Again for saving or materializing the intermediate results, additional space for storing 26260 numbers of rows will be required. In case of very huge data warehouse like in Example 1, there may be a large number of OLAP operations that are to be performed. In these cases, out of the large number of complex queries, some queries may be triggered very frequently. These frequent complex queries again may share a large number of sub-expressions. Therefore results of a set of sub-expressions may be materialized for minimizing the total query processing cost with minimized materialized view maintenance cost and space cost.

But, the derived relations and base relations for select, project and join operations in relational model based query processing are to be designed considering the indices and keys used in the relations. Therefore grossly selecting shared

temporary results of sub-expressions may prove to be a bad decision when indices on base relations are defined [7].

In [7], a directed acyclic graph (DAG) of multiple view processing plan (MVPP) by connecting the base relations and the selections, joins and projections considering the respective keys and indices are proposed for saving results of the derived relations as materialized views.

1.1.6 Materializing results of common sub-expressions in Big data framework

For managing very huge volume of incongruent data (in terabytes , zetta (10^{21}) bytes etc.) at right speed for analyzing in right time frame, a system model called *Big data* has been evolved. In Big data a computing paradigm called MapReduce is introduced [16]. In MapReduce model, computation tasks on Big data are to be broken up into units that can be distributed around a cluster of commodity hardware and server class hardware for providing cost-effective processing with scalability (see Figure 1-4). This system was designed for handling very big and semi structured data in a single table. A *Distributed File System* (DFS) computing framework has been designed based on the MapReduce model [17]. This DFS is designed such that each split of the data for MapReduce computation is from a single big table or file instead of large number of small files because in this DFS, the block size is at least 64MB and smaller file size of less than the block size imposes unnecessary overhead.

Hive is a framework for data warehousing on top of Hadoop DFS (HDFS) [17] of Apache with a query language called *HiveQL* similar to SQL, which was initially designed for OLAP like operations on huge volume of data that Facebook stored in HDFS [17]. Hive runs on workstations and convert HiveQL query into a series of MapReduce jobs. Hive organizes data into tables as a mean for providing structure to data stored in HDFS. But Hive is integrated with another platform called *HBase* which supports row-level update like delete, update etc. on big column oriented table of key-value type storage. Therefore, Hive supports collection data type columns STRUCT, MAP and ARRAY. Using collection data type may break normal form. In Big data systems a benefit of sacrificing normal form is higher processing throughput [18]. Again Hive has very limited indexing capabilities. Therefore, in Big data management, for analytical processing queries, the use of indices and keys in the relations are very limited. And hence, common or shared sub-expressions' results of frequent queries may be saved as materialized views. Materialized view is not currently supported by Hive. Research work is going on for supporting materialized views in HDFS [19, 20].

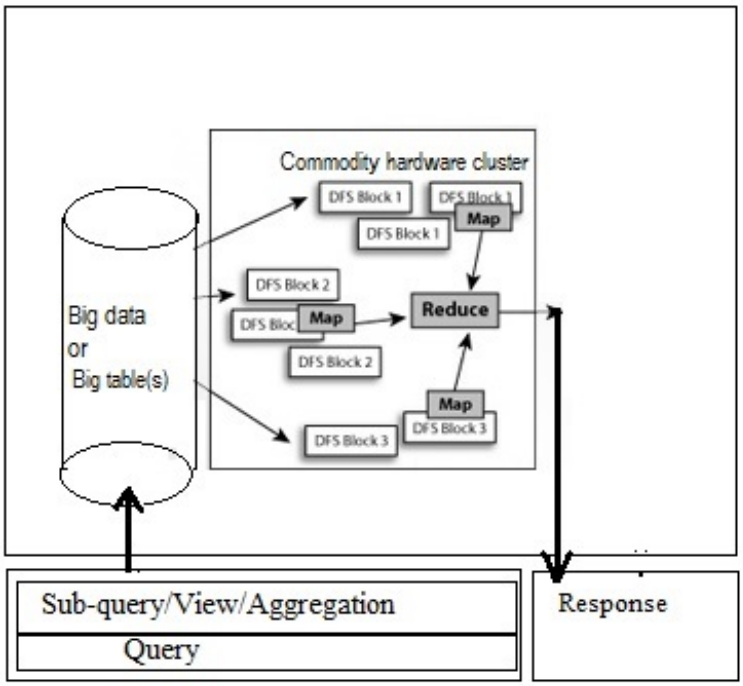

Figure 1-4: Analytical processing in MapReduce framework.

1.2 Multi-Objective Optimization to Select Views for Materializing

View materialization for reducing query processing costs in data warehouse applications requires a large amount of space. Again, materialized views are to be updated or maintained in response to changes in the base data periodically. On the other hand, not materializing any view requires lots of redundant on-the-fly computations. Therefore, it is necessary to select an optimum set of views to materialize to increase analytical query processing performance with optimized view maintenance cost and space for materializing the views.

1.2.1 The view selection problem for materializing

The materialized view selection problem is stated as- given a set of data warehouse queries, select a set of views to materialize so that the total query processing cost, materialized view maintenance cost and storage space for materializing the views are minimized [4, 21, 22]. Formally the problem may be defined as Definition 4.

Definition 4. *Given a set of n frequent queries* $Q = \{q_1, q_2, \cdots, q_n\}$ *on a data warehouse, and the set of m views* $V = \{v_1, v_2, \cdots, v_m\}$ *generated while responding the queries* Q*, a set of views* $M \subseteq V$ *are to be selected for materializing, such*

that, if A_M is the space requirement for materializing M, C_M^Q is the total cost of responding queries Q when M is materialized and $U(M)$ is the maintenance cost of materialized views M, then the selection M minimizes C_M^Q, $U(M)$ and A_M.

1.2.2 The costs to be minimized and trade-offs

The costs stated in Section 1.2.1 that are to be minimized and their dependencies are as follows:

- The maintenance or updating cost of each of the views of the set selected for materializing depends on underlying relations or base relations or tables from which that particular view is derived and the updating frequency of corresponding base relations.
- The total analytical query processing cost on the warehouse that can be minimized by materializing a set of views depends on aggregation functions, database functions like select, project, (expensive) join and "group by" on corresponding underlying base relations that are used to construct each of the selected views and frequencies of the analytical processing data warehouse queries that access these views.
- Thirdly, the total space requirement for materializing depends on individual size of the views that are selected but not on cardinality of the set of views materialized.

Thus the costs that are to be minimized depend on some underlying or internal factors but independent of each other. But there are **trade-offs** to be made because materializing more number of views may reduce the total query processing cost but there may be increase in maintenance cost and space cost of materializing the views. Also reducing number of materialized views may decrease materialized view maintenance cost and materializing space cost but it may increase the total analytical query processing cost of the data warehouse. Thus when one minimization objective improves others may degrade. Therefore, simultaneous minimization of all the costs defined in Definition 4 is not possible and trade-offs between them are to be decided.

1.2.3 Multi-Objective Optimization (MOO)

Multi-Objective optimization (MOO) is simultaneous optimization of more than one objective functions where optimal decisions are to be made considering the presence of trade-offs between objective functions, i.e, when increase in one objective function value decreases the objective function values of at least one of the other objective functions of the problem. Here trade-off means the balancing factors between the objective function values that can not be simultaneously minimized or maximized.

Formally in mathematical terms multi-objective optimization for minimization problem can be formulated as below [23].

$$minimize(f_1(x), f_2(x), \cdots, f_{\mathbf{M}}(x)),\ such\ that,\ x \in \mathbf{S} \tag{1.1}$$

where $\mathbf{M} \geq 2$ is the number of objectives and $\mathbf{S}$ is the set of all feasible solution vectors.

Multi-objective optimization is defined for maximization or minimization of objective function values where there does not exist typically a single solution that maximizes or minimizes all the objective functions simultaneously. Therefore, the multi-objective optimization technique finds the solutions of the problem that can't be farther improved for any of the objectives without farther degrading at least one of the other objective function values. In the terminology of MOO, these solutions are called *non-dominated solutions*. But by simply generating all the non-dominated solutions without indicating their distribution in objective function space it may not be possible to find the most applicable solutions or solution vectors by decision makers. Therefore, the non-dominated solutions are presented or expressed as curve (in case of two objectives) or surface (in case of more than two objectives) in objective function space known as **Pareto front** or **Pareto optimal front** such that decision maker can easily decide on trade-offs to be considered in objective function space.

1.2.4 Selecting views by multi-objective optimization technique

The view selection problem for materializing in data warehouses may be solved by converting it into a single objective optimization problem of minimizing the summed up cost function values of all the objectives functions C_M^Q, $U(M)$ and A_M for a set of materialized views M as defined in Definition 4 with some associated constraints on the costs. It returns many solution sets of views corresponding to the minimum value of summed-up cost function values. For selecting solution set of views instead of defining too many constraints on each and every objective function by the decision makers before hand, trade-offs may be decided between C_M^Q, $U(M)$ and A_M. As there are trade-offs to be considered and simultaneous minimization of all the costs defined in Definition 4 is not possible, instead of converting the problem to a single objective optimization problem by summing-up all the associated costs, the problem can be defined as a multi-objective optimization problem for solving by multi-objective optimization techniques.

Research on this problem started in the early nineties when several heuristic greedy algorithms were proposed [4, 6, 21, 22, 24]. As the problem is found to be NP-hard [4, 7, 9, 21, 22, 25], various stochastic or evolutionary algorithms and clustering based approaches have been proposed.

1.3 Motivation of this Research

In Section 1.2, it is shown that the view selection for materialization in data warehouses is basically an optimization problem with multiple objectives. But so far, in most of the works it is observed that, the problem is addressed by converting it into a single objective optimization problem. When an optimization problem with multiple non-dominating objectives is converted into single objective, it ignores that different solutions may offer trade-offs between the objectives. Though recently few attempts have been made to handle the problem by using multiple objective optimization techniques [26], yet there is scope of using other multiple objective optimization techniques in this area.

In last two decades a number of multi-objective evolutionary algorithms (MOEAs) have been developed. In [27], it is shown that Differential Evolution (DE) algorithm [28] can achieve better results than Genetic Algorithms (GAs) on numerical multi-objective optimization problems. Again non-dominated sorting genetic algorithm-II (NSGA-II) [29] was found to be able to maintain a better spread of solutions and converge better compared to other elitist MOEAs. The DE is a powerful stochastic optimization algorithm for real parameter optimization [28]. A support system may be designed for selecting views for materializing by applying evolutionary algorithms like DE, NSGA-II [29] or Multi-objective Simulated Annealing (MOSA) [30].

For managing large amount of analytic workloads on Big data systems, research works have been started for materializing frequent queries in clusters of disks, solid-state drives (SSD) and other memories [19, 20]. To cope up with the changing paradigm of very large data processing, the materialized view selection problem have to be designed in the context of distributed computations in Big data framework and to evaluate the performance of different techniques for selecting frequent sub-queries of frequent queries as views to materialize for minimizing the costs of analytical query processing.

This research is largely motivated by the observation that for strategic decision making by efficient analytical processing of historical data in data warehouses, some most relevant intermediate views with minimum associated costs are to be selected for materializing in data warehouses by a suitable optimization technique.

1.4 Research Objectives

The broad objectives of this research work are to (i) design a system for recommending a set of views or frequent sub-queries for storing as materialized views in data warehouse, considering the associated costs of maintaining the materialized views and space, and savings in total query processing costs, (ii) to find a suitable optimization algorithm for selecting the views by studying the existing works on this problem and finding the associated issues, (iii) to design input and

input methodology for the recommendation system and finally (iv) to develop a comprehensive materialized view selection mechanism compliant to the changing data warehousing and query processing paradigm.

The basic objectives of the research work are therefore may be listed as follows :

- To study different view materialization and materialized view selection techniques used in conventional data warehousing
  - To identify the associated issues and challenges involved in the problem.
  - To identify possible pitfalls and advantages in different existing representations and formulation of the problem and solutions.
  - To find the advantages and limitations of different techniques that have been so far incorporated in materialized view selection.
  - To identify the implications to theory and practices.
- Defining the view selection for materializing problem
  - By formulating and modeling a suitable representation to address the issues of existing approaches.
  - To improve the quality of solutions and performances by incorporating new or modified notions or representation of the problem and technique adaptable to the new definition.
  - To design a test bed for evaluating the performances of different view selection techniques.
- Designing a view selection system compliant to evolving Big data framework
  - By defining the problem of selection of sub-queries and aggregations for materializing in data warehouse.
  - To select a suitable optimization technique for customizing it for the problem.
  - To implement the optimization technique customized for view selection.
  - To perform experimentation for evaluating the proposed system in an accepted framework.
- Evaluating different state of the art optimization techniques for implementing in recommendation system for selecting views
  - By implementing modern and established stochastic and evolutionary optimization algorithms.
  - By measuring convergence to the optimum solutions by different techniques.
  - By designing test data for experimenting with suggested Big data paradigm and applicable algorithms.
  - By designing a prototype framework of recommendation system to provided optimum set of views for materializing.

1.5 Contributions

In this research, the problem representations, data structures and algorithms in different models proposed so far in view selection for materialization in data warehousing have been surveyed and the associated issues and challenges in addressing this NP-hard problem have been identified and reported. The view selection problem for materializing in data warehouse is defined as a multi-objective optimization problem using a cost model representing query processing plan of a set of frequent queries represented by Directed Acyclic Graph (DAG) for conventional RDBMS based data warehousing. The view selection to materialize also has been defined for Big data based Distributed File System (DFS) framework. The Multi-objective Differential Evolution (DE) algorithm has been customized for binary encoded solution representation using *Formae analysis* [31] for view selection in case of conventional RDBMS based data warehousing as well as Big data/DFS based framework. The Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [32] has also been applied using the same test-bed and test-data for comparing performances between the algorithms in materialized view selection. Finally a version of Archived Multi-Objective Simulated Annealing (AMOSA) [30] has been implemented on this problem and its performance is analyzed with other MOEAs based on *purity* and *convergence* towards Pareto front by solutions obtained and the *minimal spacing* between the solutions.

In this dissertation, the works done for achieving the above mentioned goals and contributions are discussed. The organization of the dissertation is presented in the next Section 1.6.

1.6 Organization of the Thesis

The rest of the dissertation is organized as follows.

- **Chapter 2**: *Approaches and Issues in Materialized View Selection of Data Warehouses* - In this chapter a comprehensive survey of approaches introduced to address the materialized view selection problem have been discussed. The key issues and research challenges in handling the problem are identified here with discussion on implications to related theory and practices. A comparative analysis of different approaches also have been presented based on theoretical analysis and empirical results presented in related literature.
- **Chapter 3**: *Multi-Objective Differential Evolution Algorithm for Selecting Views to Materialize* - In this chapter the view selection process for materializing in data warehouse is defined as a multi-objective optimization problem. This chapter presents an implementation of Multi-objective Differential Evolution (MODE) algorithm for binary encoded data to select views for materializing.

- **Chapter 4**: *Materialized View Selection by Evolutionary Algorithm for Big Data Query Processing* - The view selection to materialize for speeding up query processing in Big data framework is defined as a multi-objective optimization problem in this Chapter. Implementation of a modified version of Multi-objective Differential Evolution (MODE) algorithm with binary encoded data and implementation of Non-dominated Sorting Genetic Algorithm -II have been discussed in this chapter with their performance analysis in handling this problem.

- **Chapter 5**: *Multi-objective Simulated Annealing Algorithm in Big data View Selection for Materializing* - This chapter presents how multi-objective Simulated Annealing algorithm based techniques may be applied in selecting sub-query results or views in MapReduce based query processing framework. A comparative performance analysis of this technique with respect to common EA based techniques in view selection problem in this paradigm has also been presented in this chapter.

- **Chapter 6**: *Conclusion and Future Direction* - This Chapter concludes the dissertation by summarizing the overall contribution and identifies some future directions of research in this area.

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-

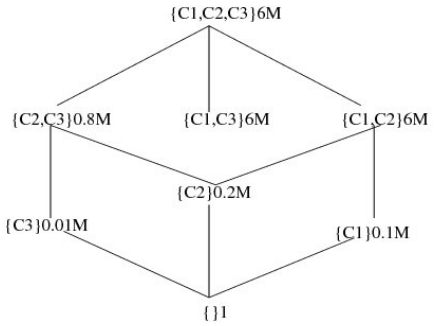

Figure 2-1: A lattice structure for 3 attributes

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.

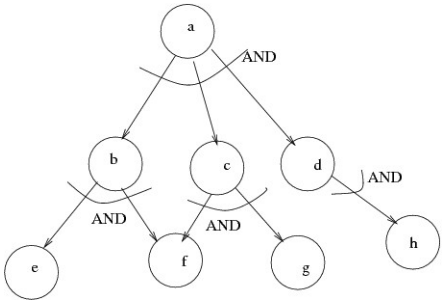

Figure 2-2: An Expression AND DAG

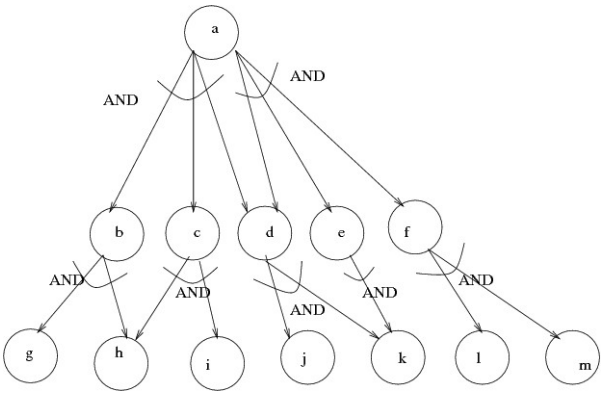

Figure 2-3: An Expression AND-OR DAG

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;
```

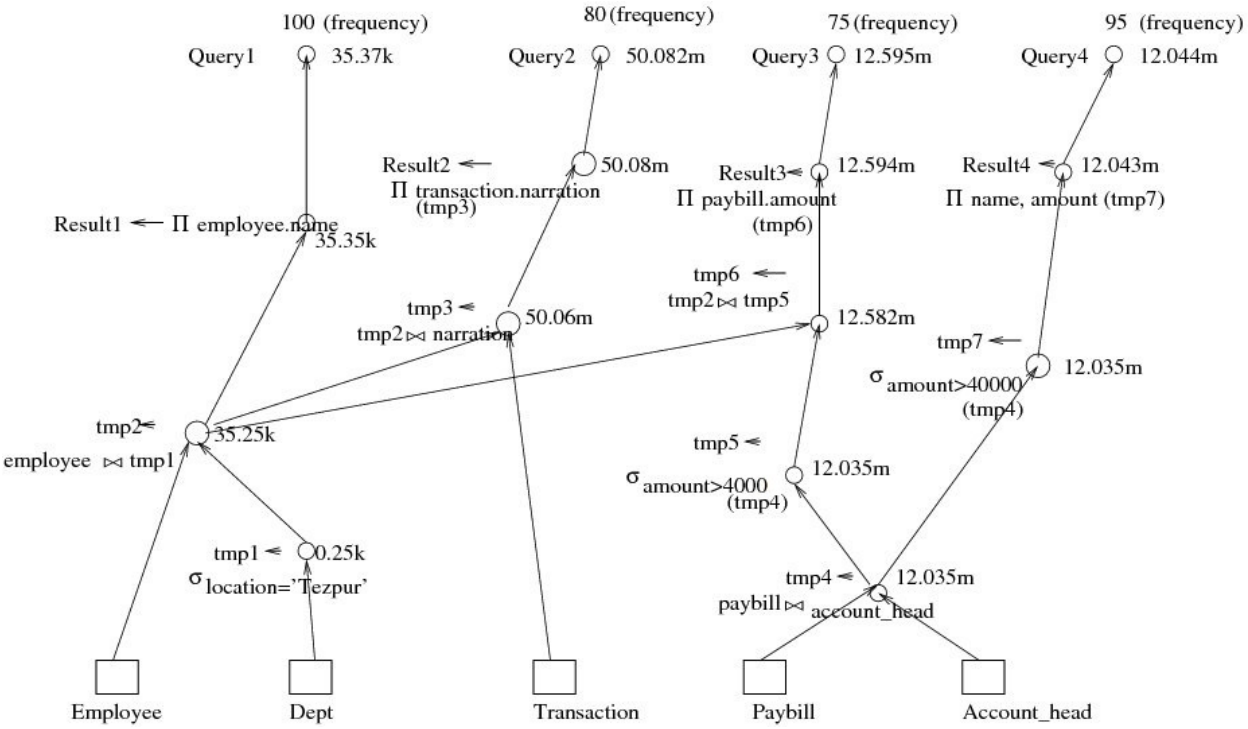

Figure 2-4: An MVPP graph

- 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

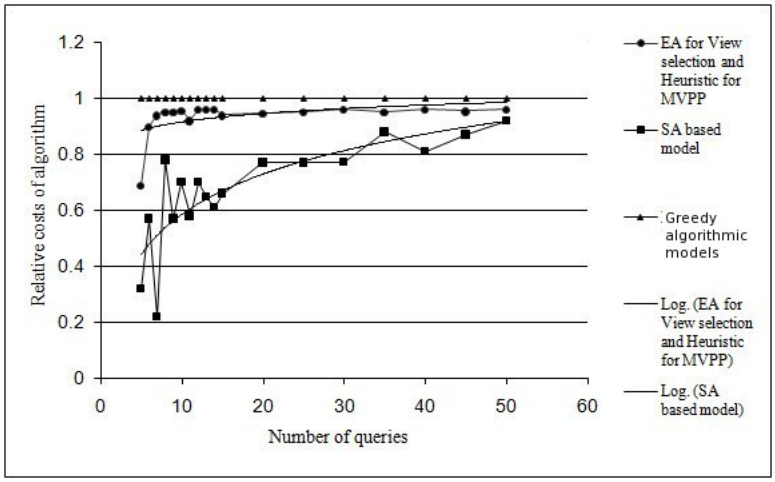

Figure 2-5: Relative costs of Heuristic, Evolutionary and Simulated Annealing algorithm in view selection using query processing plan graph representation.

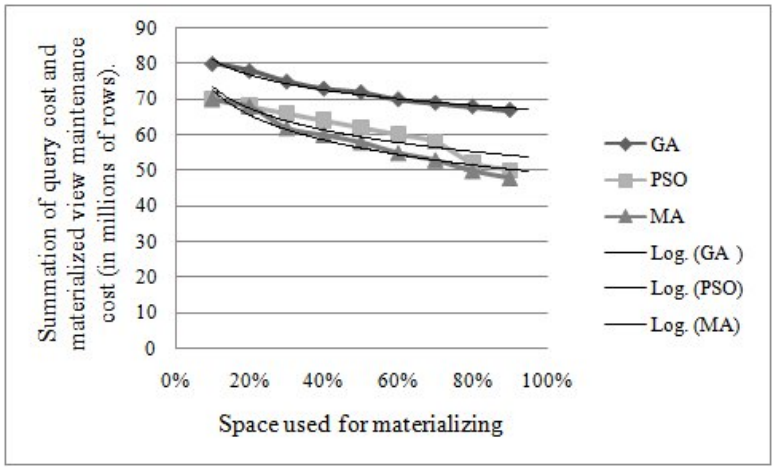

Figure 2-6: Comparison of GA, PSO and MA based view materialization models with respect to total query processing costs vs. space used by materialized views.

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.

Chapter 3

Multi-Objective Differential Evolution Algorithm for Selecting Views to Materialize

3.1 Introduction

The problem of data warehouse view selection for materialization has been established to be NP-hard [3,4,21,22,33]. Therefore, various stochastic or evolutionary algorithms and data mining based approaches have been proposed with different data structures and representations or notions. Most of these efforts treat this problem as single objective optimization problem. While the basic Multi-Objective Evolutionary Algorithm (MOEA) known as Multi-Objective Genetic Algorithm (MOGA) has been successfully used in view selection problem for conventional data warehousing [26] for its ability to find multiple Pareto-optimal solutions in one single run, extensive work by applying other modern effective and robust multi-objective optimization techniques are yet to be explored well.

3.1.1 Motivation

In [38], it has been explained that this problem is best suited for applying randomized algorithms. While applying basic *Multi-Objective Evolutionary Algorithms* (MOEAs) like the *Multi-Objective Genetic Algorithm* (MOGA) and *Niched Pareto Genetic Algorithm* (NPGA) in view selection problem, experiments on both real and synthetic data sets with varying distributions show that these non-elitist MOEAs are very competitive against the leading greedy algorithms [26]. Although basic MOEAs are able to find multiple Pareto-optimal solutions in one single run [32], elitism by preserving diversity in intermediate generations is still not considered.

The Differential Evolution (DE) algorithm introduced by Storn and

Table 3.1: Performances of multi-objective DE and NSGA-II with respect to DTLZ test problems.

Test Problems	Algorithms	Performances compared to NSGA-II	
		2 objectives	3 objectives
DTLZ1	DEMO^{NS-II}	no significant difference	**better**
	GDE3	**better**	**better**
DTLZ2	DEMO^{NS-II}	no significant difference	no significant difference
	GDE3	not evaluated	not evaluated
DTLZ3	DEMO^{NS-II}	**better**	**better**
	GDE3	not evaluated	not evaluated
DTLZ4	DEMO^{NS-II}	**better**	no significant difference
	GDE3	**better**	**better**
DTLZ5	DEMO^{NS-II}	no significant difference	no significant difference
	GDE3	not evaluated	not evaluated
DTLZ6	DEMO^{NS-II}	**better**	**better**
	GDE3	not evaluated	not evaluated
DTLZ7	DEMO^{NS-II}	no significant difference	no significant difference
	GDE3	not evaluated	not evaluated

Price [28] outperforms GAs on many numerical single objective optimization problem [27]. The original DE designed for single objective optimization has been recently developed for multi-objective optimization in different approaches that use non-dominated sorting of Pareto-ranks and crowding distance for elitism [27, 56–59]. While evaluating with the set of 9 test problems termed as DTLZ proposed by Deb et al. in [60] for testing and comparing performances between different MOEAs, Kukkonen et al. in [59] found that the generalized DE algorithm for multi-objective optimization (GDE3) with certain parameters on bi-objective and tri-objective test problems, DTLZ1 and DTLZ4 performs better than the NSGA-II proposed by Deb et al. in [32]. Another multi-objective DE named DEMO, that uses NSGA-II like elitist diversity preservation in solution, shows comparable performances with respect to NSGA-II in case of DTLZ1 to DTLZ7, when tested for 2,3 and 4 objectives [27]. From the performance evaluation results of two popular versions of DE presented by [27] and [59], for 2 and 3 objectives, compared to elitist GA (NSGA-II) for DTLZ1 to DTLZ7 is summarized in Table 3.1.

3.1.2 Contribution

In this chapter I present my attempt to find sets of views, from views generated while processing a set of complex frequent queries, so that if the select set of views are materialized, then the total query processing cost and the total materialized view maintenance cost is optimum in limited availability of space for materializing. For optimizing total query processing cost and maintenance cost of a select set of complex queries in a specific period, I use availability of space for materializing,

query frequencies during the period, and data warehouse updating frequencies in the period as parameters. The problem is defined as a multi-objective optimization problem and an attempt has been made to solve the problem using Multi-objective Differential Evolution algorithm. An initial version of this work can be found in [25]. The DE algorithm is a powerful stochastic real-parameter optimizer for non-linear and non-differentiable continuous space function [28]. Gong et al. in [31] present the use of *forma analysis* to exploit usage of DE for discrete optimization problem. Here, I attempt to use formae analysis as presented in [31] to implement multi-objective DE in selecting views for materializing in data warehouse using MVPP graph framework [7].

In Section 3.2 the view selection problem for materializing in data warehouse is defined as a multi-objective optimization problem using a cost model representing query processing plan of a set of frequent queries represented by Directed Acyclic Graph (DAG). Section 3.3 describes the design of a Multi-objective DE for binary encoded solution representation using *algebra of GA* and formae analysis for materialized view selection. In Section 3.4 the experimentation and evaluation are presented with a discussion and analysis of obtained results. Finally concluding remarks and ensuing works are presented in Section 3.5.

3.2 The View Selection for Materializing as a Multi-objective Optimization Problem

In data warehouses, a view consists of the result of an aggregation function on some other views or base tables of the warehouse produced while generating responses to queries. Thus, views are dependent on the contents of other views and base tables. To make query response faster, optimization is critical in selecting some or all of these views for materializing in the data warehouse. The goal is to find a set of views for materializing to obtain an optimum query response cost in terms of response time or the total number of rows to be processed, an optimum maintenance cost or updating cost of the materialized views, with optimum space requirement for materializing the views. Generally in real life data warehousing, it is feasible to reserve a specific amount of memory for materializing the selected views. Therefore, the view selection for materializing problem is to select some or all of the views that are generated frequently while processing queries on a data warehouse, such that, the space requirement to materialize the selected views is less than or equal to the space reserved for that, and if the selected views are materialized, the total query processing cost and materialized view maintenance cost becomes minimum.

While processing a set of n frequent queries $Q = \{q_1, q_2, \cdots, q_n\}$ on a data warehouse, if it generates m intermediate views $V = \{v_1, v_2, \cdots, v_m\}$, the materialized view selection problem is to select an optimum set of views $M \subseteq V$ for materializing within a given available space A (for materializing), such that, if the set M is materialized and A_M is the space requirement for materializing the set of views M , it minimizes C_M^Q, the total cost of answering query Q when M is

materialized, and $U(M)$, the maintenance cost of M due to updating of the base tables used by queries Q, with the constraint $A_M \leq A$.

3.2.1 DAG representation of multiple query processing plan

The view selection for materializing in data warehouse may be represented by a Directed Acyclic Graph (DAG) that considers a set of frequent (OLAP) queries on a data warehouse on a specific period and assuming that the overall query processing efficiency will be maintained if the intermediate views generated, in the middle of processing these queries, are considered for materializing. The Query Processing Plan DAG framework to represent the view selection problem was originally defined by Yang et al. in [7] as Multiple View Processing Plan (MVPP) framework. The DAG in this framework represents a query processing strategy on a data warehouse. The leaf nodes of the DAG correspond to the base relations and the root nodes represent the responses of queries.

Definition 8. *Query processing tree for a query q is a DAG, $T_q = (V, A)$, where,*

- *V is the set of vertices representing intermediate select, join and project sub-expressions of the query q,*
- *A is the set of arcs $\{a_1, a_2, \cdots, a_n\}$, such that each arc $a_i \in A$, either*
  - *connects a vertex $u_i \in V$ to $v_i \in V$, directing u_i to v_i, if v_i returns a number of rows of a database relation by processing rows of database relation at u_i,*
  - *or connects a leaf node or base relation $b_j \in B_q$ to $v_i \in V$, directing b_j to v_i, where B_q are the base tables used by q such that the processing at v_i needs the data in b_j,*
  - *or it is connecting a vertex $v_i \in V$ to the root node representing the final response of the query q [7].*

Definition 9. *An MVPP Directed Acyclic Graph (DAG) G is a graph generated by merging query processing trees $T_{q_i}, i = 1, \cdots, n$, for a set of queries $Q = \{q_1, q_2, \cdots, q_n\}$ on a data warehouse considered, whenever sub-graphs of the query processing trees are shared [7].*

To generate the MVPP DAG framework, all the considered frequent queries are analyzed and all the independent selection, projection, join and base relations are identified to represent as vertices of the DAG as defined in Definition 8 and 9. The nodes of the DAG are then labeled in a specific order. By considering the frequencies of the queries and updating frequencies of the base tables, sizes of the resultant sub-expressions represented as vertices of the graph are evaluated. For every query, there may be several plans of execution. Therefore every query may have an execution plan which is the optimum in terms of

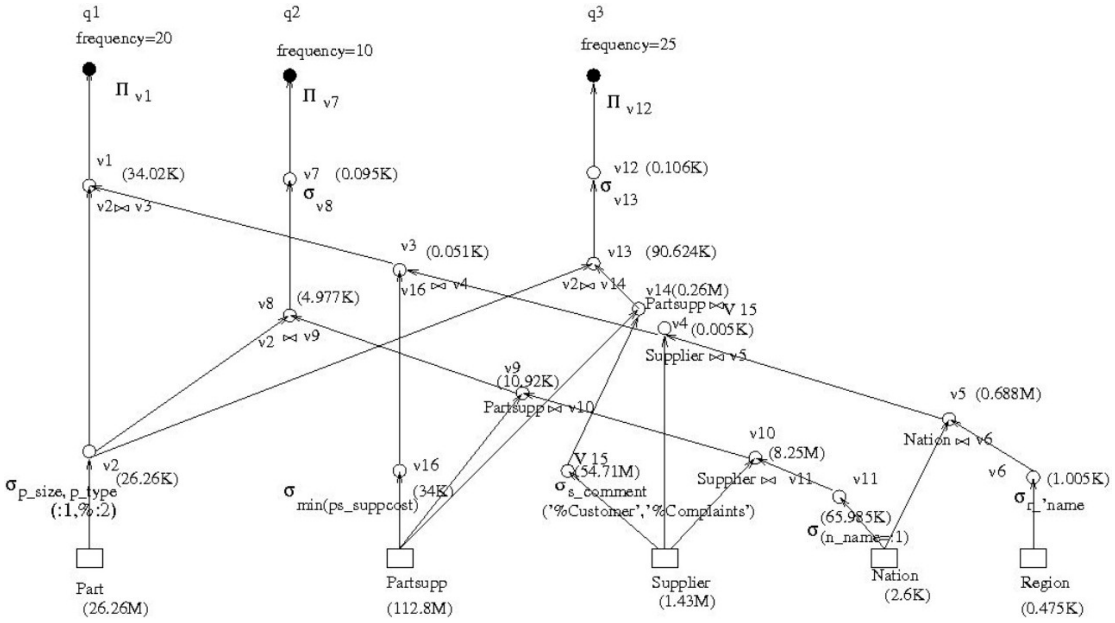

Figure 3-1: An example MVPP graph using TPC-H benchmark data warehouse

rows to be accessed. But simple merging of these optimal query execution plans or optimal query processing trees may not lead to an optimum common multiple multiple execution plan as there may be several *select, project, join* and *aggregation* that are shared among the queries in different ways. For designing an MVPP DAG for using as input to materialized view selection algorithm, Yang et al. in [7] suggest that, first optimal query processing plan for each of the considered queries are to be designed individually by pulling up all select, project and aggregation and pushing down the *join* because join operations are the most expensive operations in query processing. Then the individual query trees are merged into a single query execution plan termed as MVPP DAG by pushing down and pulling up the select, project, aggregation and join operations such that the shared join operations are merged as early as possible.

An MVPP DAG may be constructed as depicted in Figure 3-1. The view selection problem using MVPP DAG framework is to select a set of nodes to materialize with space constraint for materializing the views from an MVPP DAG, excluding the leaf and root nodes of the DAG, to minimize the total query processing cost and materialized view updating cost.

3.2.2 The cost model

The query processing cost for a query is the total size of data that are to be accessed to generate the result or response. A query may have several sub-expressions as select, join and project relations. Each of these sub-expressions return some amount of data by reading other sub-expressions or base relations. Thus query processing cost of a query is the total amount of data that are to be accessed from each of the sub-expressions and base tables. When a particular node or view of an MVPP graph is materialized, the queries that access this view do not have

to access the other nodes used to generate this particular materialized node or view. For example, in an MVPP DAG as in Figure 3-1 where M stands for millions of rows and K stands for thousand rows, let v_2 is the result of a select operation by accessing 26.26 millions of rows from Part table returning 26.26 thousand rows of data. If v_2 is materialized, then for processing q_1, q_2 and q_3, the 26.26K rows of v_2 are to be accessed 55 (20+10+25) times instead of accessing the Part table of data size 26.26M for 55 times. But during the period in which query frequencies 20, 10 and 15 are considered for the selected queries, if the base tables are updated 2 times, then v_2 is to be reconstructed by accessing the Part table of 26.26M rows 2 times. Thus the total benefit of materializing v_2 is $55 \times (26.26M + 26.26K) - (55 \times 26.26K + 2 \times 26.26M)$.

Thus, for each query in the MVPP DAG considered, first the total amount of data that are to be read are computed by considering the materialized views. The result is then multiplied by the frequency of the corresponding query to get the query processing cost for the query of the MVPP DAG. The summation of query processing cost of all the queries of the MVPP DAG by considering a set of materialized views is the query processing cost of the MVPP DAG.

The query processing cost $Q_G(M)$ for a set of materialized views M, of an MVPP DAG G is expressed as Equation 3.1 by Yang et al. in [7], where, V is the set of vertices of G and $M \subseteq V$, R is the set of root nodes of G, f_q is the access frequency of query $q \in R$, and $C_a^q(v)$ is the query processing cost of query q by accessing vertices $v \in V$ when M is materialized.

$$Q_G(M) = \sum_{q \in R} f_q(C_a^q(v)) \tag{3.1}$$

The maintenance cost of materialized views is the cost of re-constructing the materialized views when the base tables are updated. Suppose a view v is generated by reading views v_1 and v_2 and to generate views v_1 and v_2, base tables b_1 and b_2 are to be accessed. If the view v is materialized, then whenever b_1 and b_2 are updated, the materialized view v is to be reconstructed by accessing the records of v_1, v_2, b_1 and b_2. This accessing cost is termed as view maintenance cost. During a specific period, if the base tables are updated n number of times, then the materialized view maintenance cost is multiplied by n to compute the total maintenance cost of the materialized view v during that period. When a set of views $M = \{v_1, v_2, \cdots, v_n\}$, $M \subseteq V$, of an MVPP DAG G is materialized where V is the set of vertices of G, then the summation of all the maintenance cost for materialized views $v_1, v_2, \cdots, v_n$ is the maintenance cost of the MVPP DAG G.

The materialized view maintenance cost of an MVPP DAG G is expressed as Equation 3.2 below by Yang et al. in [7].

$$U_G(M) = \sum_{m \in M} f_m(C_u^m(r)) \tag{3.2}$$

where, V is the set of vertices of G representing views, $M \subseteq V$ is the set of views materialized, f_m is the maintenance frequency of materialized view $m \in M$,

$C_u^m(r)$ is the cost of updating materialized view $m \in M$ by accessing the vertices r, $r \subset V$.

Example 7. *Considering the MVPP DAG in Figure 3-1 as G, Let* $M_1 = \{v_2\}$ *and* $M_2 = \{v_{10}, v_{14}\}$ *are two solutions of the materialized view selection problem. Then for* M_1 *the total query processing cost of MVPP DAG G is*
$Q_G(M_1) = 20 \times (0.034 + 0.02626 + 0.000051 + 0.000005 + 112.8 + 0.688 + 1.43 + 0.001 + 0.00265 + 0.000095 + 0.000475) + 10 \times (0.004977 + 0.0109 + 0.02626 + 8.255 + 112.8 + 0.0659 + 1.43 + 0.0001 + 0.00265) + 25 \times (0.0906 + 0.2603 + 0.02626 + 54.71 + 112.8 + 0.034 + 1.43)$
$= 7658.5M$,
the maintenance cost $U_G(M_1) = 2 \times 26.26M = 52.52\ M$,
the space requirement for materializing M_1, $A_{M_1} = 0.02626M$
Similarly,
$Q_G(M_2) = 20 \times (0.034 + 0.02626 + 26.26 + 0.000051 + 0.000005 + 0.688 + 1.43 + 0.001 + 0.00265 + 0.000095 + 0.000475) + 10 \times (0.004977 + 0.0109 + 0.02626 + 26.26 + 8.255) + 25 \times (0.090624 + 0.260352)$
$= 923.17M$
$U_G(M_2) = 2 \times (112.8 + 0.065985 + 1.43 + 0.000106 + 0.00265) + 2 \times (26.26 + 0.02626 + 54.717 + 112.8 + 0.034 + 1.43) = 619.2M$
$A_{M_2} = (8.255 + 0.260352) = 8.515M$.

3.2.3 Multi-objective optimization

Multi-objective optimization is simultaneous optimization (i.e, maximization or minimization) of a set of objective functions on some parameter vectors. In single objective optimization there is only one global optimum value of the objective function and therefore there may be just one optimum solution. But in case of multi-objective optimization there is a set of solutions that are equally acceptable because when one solution yields the best value for one objective function, it may not produce the best result for the other objective functions of the problem.

For two solutions of a multi-objective optimization problem, when a particular solution yields objective function values that are not worse compared to the objective function values produced by the other solution but better for at least one objective function of the problem, then it is said that the first solution *dominates* the second solution. In case of multi-objective optimization, the problem is to find the solutions that are not dominated by any other solutions for the set of objective functions of the problem. The non dominated set of solutions of the entire solution search space is called the global *Pareto optimal* solutions of the multi-objective optimization problem. In case of Pareto optimal solutions, when one objective function value increases other objective function values decrease. Therefore, as simultaneous optimization of all objective functions is usually not possible, the graphical representation of the Pareto optimal solutions or points in objective function space is concave (for minimization) or convex (for maximization). The curve (for bi-objective optimization) or surface (for three or more objectives) describing the tradeoffs in objective function value space by the Pareto

optimal solutions are called the *Pareto front.*

Formally the multi-objective optimization problem may be defined by following definitions.

Definition 10. *For a set of vectors* $\mathbf{S}$ *and* $\mathbf{M}$ *number of real valued objective functions* f_i, $i = 1, 2, \cdots, \mathbf{M}$, $f_i \colon \mathbf{S} \to R$; *multi-objective optimization is to find the parameter vectors* $\overline{x^*} \in \mathbf{S}$ *of* D *dimensions* $\{x_1^*, x_2^*, \cdots, x_D^*\}$ *for* f_i, *such that* $f_i(\overline{x^*})$ *is simultaneously maximized or minimized for* $i = 1, 2, \cdots, \mathbf{M}$.

Definition 11. *For* $\mathbf{M}$ *objective real valued minimization problem* $f_i \colon \mathbf{S} \to R, i = 1, 2, \cdots \mathbf{M}$ *a solution* $\overline{u} \in \mathbf{S}$ *is said to dominate* $\overline{v} \in \mathbf{S}$, *if* $\forall i \in 1, 2, \cdots, \mathbf{M}$, $f_i(\overline{u}) \leq f_i(\overline{v})$ *and* $\exists i \in 1, 2, \cdots, \mathbf{M}$, *such that* $f_i(\overline{u}) < f_i(\overline{v})$. *This domination relation* $\overline{u}$ *dominates* $\overline{v}$ *is denoted by* $\overline{u} \prec \overline{v}$.

Definition 12. *For objective functions* $f_i, i = 1, 2, \cdots, \mathbf{M}$ *of a multi-objective optimization problem defined by the Definition 10, solutions such as* $\overline{x^*} \in \mathbf{S}$ *are called Pareto optimal solutions if* $\nexists \overline{x} \in \mathbf{S}$ *such that* $\overline{x} \prec \overline{x^*}$. *The set* $\mathscr{P} = \{\overline{x^*} \in \mathbf{S} \colon \overline{x^*}$ *is Pareto optimal*$\}$ *is called the Pareto optimal set of the multi-objective optimization problem.*

Definition 13. *For objective functions* f_i, $i = 1, 2, \cdots, \mathbf{M}$, $f_i \colon \mathbf{S} \to R$ *of a multi-objective optimization problem where* f_i *are components of a vector function* F, *the curve or surface produced by graphical representation of* $F(\mathscr{P}) \colon = \{F(\overline{x^*}) \colon \overline{x^*} \in \mathscr{P}\}$ *is called the Pareto front of the multi-objective optimization problem.*

For two solution vectors $\overline{u}$ and $\overline{v}$ for a set of objective functions of an optimization problem, if $\overline{u}$ does not dominate $\overline{v}$ and $\overline{v}$ also does not dominate $\overline{u}$, expressed as $\overline{u} \not\prec \overline{v}$ and $\overline{v} \not\prec \overline{u}$, then $\overline{u}$ and $\overline{v}$ are called non-dominated solutions. Multi-Objective Evolutionary Algorithms (MOEAs) and other multi-objective stochastic algorithms like multi-Objective Simulated Annealing algorithms can yield set of mutually non-dominated solutions which are only an approximation or estimation of the true Pareto front of $\mathscr{P}$. The *estimated Pareto front* is popularly denoted by $\mathscr{F}$ [61]. These multi-Objective optimization techniques are designed to find an estimated Pareto front $\mathscr{F}$ asymptotic to the true Pareto front for $\mathscr{P}$.

3.2.4 The view selection problem as multi-objective optimization problem representation

Using equation 3.1 and 3.2, if V is the set of vertices of MVPP DAG G, and M is the set of views that are materialized, i.e. $M \subseteq V$, then under the constraint $\sum_{v \in M} A_v \leq A$, where A_v denotes the space required for materializing the view v and A is the total space available for materializing the views; the view selection problem is to find M to:

$$Minimize, \mathbf{Y} = \mathbf{F}(M) \equiv (Q_G(M), U_G(M)) \tag{3.3}$$

If S_0 and S_1 are two solutions of the Equation 3.3 under the constraint $\sum_{v \in M} A_v \leq A$ then S_0 dominates S_1, expressed as $S_0 \prec S_1$, if and only if both the following logical conditions 3.4, 3.5 are satisfied.

$$(Q_G(S_0) \leq Q_G(S_1)) \; and \; (U_G(S_0) \leq U_G(S_1)) \tag{3.4}$$

$$(Q_G(S_0) < Q_G(S_1)) \; or \; (U_G(S_0) < U_G(S_1)) \tag{3.5}$$

If $S_0 \not\prec S_1$ and $S_1 \not\prec S_0$, where $\not\prec$ denotes does not dominate, then S_0 and S_1 are said to be non-dominating solutions. The view selection for materializing in data warehouse is the problem of finding out a set of non-dominating solutions, which is an approximation to the *true Pareto front* of the problem defined by Equation 3.3.

In Example 7, for input MVPP DAG G in Figure 3-1, both the solutions $M_1 = \{v_2\}$ and $M_2 = \{v_{10}, v_{14}\}$ are two non-dominating solutions for the objective functions 3.1, 3.2 of the minimization problem 3.3, considering that the available storage space for materializing is more than the space required for saving 8.515 millions of rows.

3.2.5 Solution representation

Using the notations used in the algebra of Genetic Algorithms [62], for $\mathbb{B} \triangleq \{ 0, 1 \}$ being the set of all truth values and $\mathbb{B}^m$ denoting the set of binary strings of length m, a solution S of the view selection problem may be represented such that $S \in \mathbb{B}^m$. This representation of solutions are found to be suitable for meta heuristic non deterministic multi-objective optimization like multi-objective Evolutionary Algorithms and Simulated Annealing algorithms [9,62–64]. To represent solutions of MVPP DAG framework as suggested by Yang et al. in [7], all views represented as vertices in an MVPP DAG are labeled and indexed. The solutions are represented as a string of 1s and 0s such that if a particular view is selected for materialization, then the corresponding bit in the solution string is represented as 1 and else 0. For m number of views of an MVPP DAG considered for selection, the views are indexed from 0 to $m - 1$. The solution strings of length m are represented such that if ith view v_i is selected for materialization then the ith bit of the solution string is set to 1 and else it is set as 0.

3.3 Multi-objective Differential Evolution Algorithm for Selecting Views to Materialize in Data Warehouse

3.3.1 The Differential Evolution (DE) algorithm

The DE algorithm is a stochastic parallel direct search method for global optimization problems over continuous space using NP D-dimensional vectors $x_{i,g}$, $i = 1, 2, \cdots, NP$, representing NP as the population size for generation g of an evolutionary system [28].

DE generates a new solution vector by adding the weighted difference between two population vector to another population vector of the NP population which is called *mutation*. The resultant vector is called *mutant vector*. Then some parameters of the mutant vector are exchanged with parameters of another predetermined vector of the population called *target vector* to yield a new vector called *trial vector*. This exchange of parameters is called *cross over*. If the fitness or cost function values generated by the trial vector is more preferable (or better) than the target vector, the trial vector replaces the target vector for next generation and this replacement of vector in the solution population is called *selection*. In each generation, each of the population vector is to be treated as the target vector once for selection operation.

But there are different variants of DE. The notation used to specify different variants of DE is *DE/x/y/z*. In this notation x specifies that the vector to be mutated is whether *random* or the *best* so far in the population. The y specifies the number of difference vectors used. The z specifies the cross over scheme.

In original DE, it is specified that the size of the population, NP, is unchanged during the optimization. But it has been observed that in case of applying DE for multi-objective optimization, the population size may grow in each generation and is to be controlled by maintaining the original basic characteristics of the population.

For mutation, in one variant of DE, known as *DE/rand/1/bin*, new population vectors are generated by finding the weighted difference between two random population and then by adding it to a third random population vectors of the NP population. In *DE/rand/1/bin*, "rand" indicates that the donor vector selected to compute the mutation values is chosen at random. "1" is the number of pairs of solutions chosen to compute the mutation differential and "bin" indicates that binomial recombination is used. The other mostly used similar variants of DE are *DE/rand/1/exp*, *DE/best/1/bin* and *DE/best/1/exp*. There are few other versions of DE like *DE/current-to-rand/1*, *DE/current-to-best/1* and *DE/current-to-rand/1/bin* which use arithmetic and arithmetic-discrete recombination [65].

In *DE/rand/1/bin* version of DE, the mutant vector for next generation

$g+1$ for each target vector $x_{i,g}$, $i = 1, 2, \cdots, NP$, is generated as Equation 3.6.

$$v_{i,g+1} = x_{r_1,g} + F.(x_{r_2,g} - x_{r_3,g}) \tag{3.6}$$

where $r_1, r_2, r_3 \in \{1, 2, \cdots, NP\}$, $r_1 \neq r_2 \neq r_3$, F is a real constant factor $\in [0, 1]$ and $F > 0$. Here F is used to scale the influence of the randomly selected population vectors $x_{r_2,g}, x_{r_3,g}$ while calculating the mutation value.

The mutant vector's parameters are then mixed with the target vector to yield a vector called trial vector. The crossover is introduced here to increase the diversity of the perturbed vectors. The trial vector is formed by crossover as expressed by Equation 3.7 and 3.8.

$$u_{i,g+1} = (u_{1,i,g+1}, u_{2,i,g+1}, \cdots, u_{D,i,g+1}) \tag{3.7}$$

$$u_{j,i,g+1} = \begin{cases} v_{j,i,g+1}, & \text{if } (randR(j) \leq CR) \text{ or } j = randI(i) \\ x_{j,i,g}, & \text{if } (randR(j) > CR) \text{ or } j \neq randI(i) \\ j = 1, 2, \cdots, D \end{cases} \tag{3.8}$$

In Equation 3.8 $randR(j)$ is the jth evaluation $\in [0, 1]$ of a uniform random number generator. The $CR \in [0, 1]$ is a real constant termed as *crossover constant*. The CR controls the influence of the parent in their next generation of offspring. If a higher value of CR is taken there will be less influence of the parent in the offspring vectors. The $randI(i)$ is a random index $\in 1, 2, \cdots, D$.

In selection operation of DE, if the trial vector $u_{i,g+1}$ yields a lesser cost function value than $x_{i,g}$, then $x_{i,g+1}$ is set to $u_{i,g+1}$ and otherwise $x_{i,g+1}$ is set as $x_{i,g}$. The process is continued till a pre-defined maximum value of g, say, g_{max} is reached. The best solution from the population in the final generation is selected as optimum solution.

3.3.2 Differential Evolution algorithm adapted with binary encoded data

The DE introduced by Storn and Price in [28] was originally designed for global optimization problem over continuous spaces using solution population of real vectors. Gong et al. in [31] used forma analysis [66,67] to derive discrete DE operators for discrete optimization problem. For a population vector $\Psi = \{\psi_1, \psi_2, \cdots, \psi_D\}$ of D dimensions, each decision variable ψ_i may be considered as a single dimension which may have either 0 or 1 as it's value. Thus Ψ may be represented as a string of bits, where each bit represents a particular dimension to represent a solution population vector as binary encoded data. To compute mutant vector in DE, difference between two random vectors of a population, say, $x_{r_2,g}$ and $x_{r_3,g}$ is to be amplified by a real amplification factor F and it is to be added to another

random vector $x_{r_1,g}$ of the population. In binary encoded representation, $x_{r_2,g}$ and $x_{r_3,g}$ are two vectors having binary parameters as dimensions. But as F is a real number, the resultant vector $F.(x_{r_2,g} - x_{r_3,g})$ of Equation 3.6 will be converted into a real vector and the mutant becomes incompatible with the binary string representation of the problem. Gong et al. in [31] used forma analysis [62, 68] to generate mutant vector of DE as a binary string.

Radcliffe et al. in [62, 66–68] defined forma analysis based on Algebra of GA to capture and express domain specific knowledge for performance of evolutionary algorithms. For enumerative solution search by GAs, the concept of *equivalence relations* over the solution search space S is introduced in forma analysis. A *relation*, $\sim$ in algebra of GA is a property between every pair of members or solutions of S which is either *true* or *false*. A relation, $\sim$ is called an equivalence relation when the relation is reflexive, symmetric and transitive. The equivalence relations of search space S partitions S into disjoint classes termed as *equivalence classes*. For example, in a human population, the eye colour may be an equivalence relation and the sets of people in the population with the various eye colours like blue eyed people, brown eyed people, black-eyed people etc. are different equivalence classes. Radcliffe in [66] used the term *forma* with it's plural *formae* to refer to an equivalence class.

Definition 14. ***Equivalence relation****: For* $\mathbb{B} \triangleq \{ 0, 1 \}$ *being the set of all truth values and* $\mathbb{B}^n$ *denoting the set of binary strings of length* n*, equivalence relation* ψ *is a function over the* S

$$\psi : S \times S \longrightarrow \mathbb{B} \tag{3.9}$$

iff $\forall x \in S : \psi(x, x) = 1$, $\forall x, y \in S : \psi(x, y) = 1 \Longrightarrow \psi(y, x) = 1$, *and* $\forall x, y, z \in S : \psi(x, y) = \psi(y, z) = 1 \Longrightarrow \psi(x, z) = 1$.

$\mathbb{E}(S)$ is used to denote the set of all equivalence relations over a given set S. For a given equivalence relation $\psi \in \mathbb{E}(S)$, Ξ_ψ is used to denote the set of formae or equivalence classes induced by ψ. Thus for a set of equivalence relations $\Psi \subset \mathbb{E}(S)$ where $\Psi = \{ \psi_1, \psi_2, \cdots, \psi_{|\Psi|} \}$, Ξ_Ψ is the vector of formae for Ψ.

Definition 15. ***Intersection of equivalence relations****: For* $\mathbb{B} \triangleq \{ 0, 1 \}$ *being the set of all truth values, and equivalence relations* $\psi, \phi \in \mathbb{E}(S)$*, the intersection* $\psi \cap \phi : S \times S \longrightarrow \mathbb{B}$ *is defined by*

$$(\psi \cap \phi)(x, y) \triangleq \psi(x, y) \wedge \phi(x, y) \tag{3.10}$$

where $\wedge$ *denotes logical "and" (and* $\triangleq$ *denotes defined to be equal to).*

Thus, two solutions are equivalent under the intersection of a pair of equivalence relations if they are equivalent under each of the pair.

Definition 16. ***Span****: For a set of equivalence relations* $E \subset \mathbb{E}(S)$*, the span of* E *is the set of all equivalence relations that can be constructed by intersection of any subset of* E.

Definition 17. ***Independent set of equivalence relations****: A set of equivalence relations $E \subset \mathbb{E}(S)$ is said to be independent if there is no member in E that can be constructed by intersection of other members of E.*

Definition 18. ***Basis****: If the set of equivalence relations E is a subset of another set of equivalence relationsΨ, i.e $E \subset \Psi$ and $\Psi \subset \mathbb{E}(S)$, then E is said to constitute a basis for Ψ, if and only if E spans Ψ and E is independent.*

These concepts of forma analysis can be used to derive operators to manipulate a given set of equivalence relations. An operator's behavior may be specified in terms of a basis. Thus by combining the basis with domain independent operators a given set of equivalence relations may be explicitly manipulated.

Tuson in [69] designed an operator template for changing k parameters of a forma with basis Ψ as Equation 3.11 below, where D_Ψ denotes the set of different parameters between equivalence relations using basis Ψ .

$$O_k(x, k, \Psi) = \{y \in S || D_\Psi(x, y)| = k\} \tag{3.11}$$

For representing the binary-string solutions of D dimensions, so that they can be manipulated by forma analysis, Gong et al. in [31] defined the basis as:

$$\psi_i(X, Y) = \begin{cases} 1, & \text{if } x_i = y_i; \\ 0, & \text{otherwise} \end{cases} \tag{3.12}$$

Gong et al. [31] defined formae basis [67] based *DE mutation operator template* M_{de} as Equation 3.13 below.

$$M_{de}(x_1, x_2, x_3, F, \Psi_i) = \{m \in S | D_{\Psi_i}(x_1, m) = k \wedge k = F \times D_{\Psi_i}(x_2, x_3)\} \tag{3.13}$$

where x_1 represents the base vector selected from the population, x_2 and x_3 are the vectors of the population to produce the difference, m represents the mutant vector and Ψ_i represents the basis constructed for the i-th dimension.

Now let for the basis given by Equation 3.12 for binary string representation $\Psi = \{\psi_1, \psi_2, \cdots, \psi_D\}$, where each bit is considered as decision variable of single dimension, the distance between two solutions are computed by the binary distance between the bits i.e either 1 or 0. If $x_{r_2,g}$ and $x_{r_3,g}$ are considered as two strings of bits of length D, each jth dimension difference between $x_{r_2,g}$ and $x_{r_3,g}$, $D_{\Psi_j}(x_{r_2,g}, x_{r_3,g})$ can be represented using the formae basis for Ψ_j defined in Equation 3.12 as Equation 3.14 below.

$$D_{\Psi_j}(\mathbf{x}, \mathbf{y}) = \begin{cases} 0, & \text{if } x_j = y_j \\ 1, & \text{otherwise} \end{cases} \tag{3.14}$$

This makes the jth bit of the vector difference between $(x_{r_2,g}, x_{r_3,g})$, represented as $D_{\Psi_j}(x_{r_2,g}, x_{r_3,g})$, to either 1 or 0. But, in DE, F is a real number in the range $[0, 1]$. Hence, $F.D_{\Psi_j}(x_{r_2,g}, x_{r_3,g})$ will be real value F or 0. Therefore, to

interpret the scaled difference $F.D_{\Psi_j}(x_{r_2,g}, x_{r_3,g})$ of jth dimension rounded to 1 or 0, for applying mutation operator template defined in Equation 3.13, following Equation 3.15 may be used to randomly decide whether it is to be rounded to 1 or 0.

$$F.D_{\Psi_j}(x_{r_2,g}, x_{r_3,g}) = \begin{cases} 1, & \text{if random}[0,1] < F \wedge (D_{\Psi_j}(x_{r_2,g}, x_{r_3,g}) = 1) \\ 0, & \text{otherwise} \end{cases} \tag{3.15}$$

The mutant vector $v_{i,g+1}$, using expressions 3.14 and 3.15 is thereby generated as:

$$v_{j,i,g+1} = D_{\Psi_j}(x_{r_2,g}, F.D_{\Psi_j}(x_{r_2,g}, x_{r_3,g})) \tag{3.16}$$

This type of mutation operation is termed as *restricted-change mutation* [31].

The DE cross-over operation, as expressed by Equation 3.8, manipulates population vector in a discrete form. Therefore, the DE crossover operator is directly usable for discrete or binary encoded data.

3.3.3 Multi-objective DE

Simultaneous optimization of several objectives is not usually possible to attain. Therefore, in multi-objective optimization, a set of globally non-dominating solutions are to be determined. To solve multi-objective optimization problem by using DE algorithm, initially Pareto-dominance based approach was used where a trial solution vector is constructed for each member $x_i, i = 1, 2, \cdots, NP$, of the NP candidate solution population by mutation and crossover operation on three other randomly selected different solution vectors x_{r_1}, x_{r_2} and x_{r_3} of the solution population. The trial vector is selected to replace x_i for next generation population if the trial solution vector dominates the parent. Otherwise the trial vector is discarded [70]. In recent popular multi-objective DE techniques [27,58,59,71], the trial vector replaces the parent when it dominates the parent, and the trial vector is discarded in case the parent dominates the trial solution vector. But when the parent does not dominate the candidate and the candidate also does not dominate the parent, the candidate solution or the trial vector is added to the solution population without considering whether any other solution other than the parent in the solution population dominates the candidate or the candidate dominates any other solution of the population. This is done to increase diversity in the solution population of intermediate generations and thereby to avoid getting trapped in local optimum. To control the increasing population size in intermediate generations different techniques have been used as discussed in the Section 3.3.4. The DE runs for a specified number of generations and at the end of the evolutionary process, the non-dominated solutions are filtered out.

Algorithm 4: Multi-objective DE using Binary Encoded Data for selecting views to materialize in data warehouse

Require: NP, g_{max}, F, CR, D, MVPP DAG G, constraints
Ensure: A set of non-dominated solutions

Generate NP random vectors $x_1, x_2, \cdots, x_{NP}$ of dimension D that satisfy the specified constraints
$N \leftarrow NP$
$g \leftarrow 1$
repeat
  for $i = 1$ to N **do**
    select x_i and $x_{r_1}, x_{r_2}, x_{r_3}$, such that $x_i \neq x_{r_1} \neq x_{r_2} \neq x_{r_3}$
    for $j = 1$ to D **do**
      $v_{j,i} \leftarrow D_{\Psi_j}(x_{r_1}, round(F.D_{\Psi_j}(x_{r_2}, x_{r_3})))$
where $round(F.D_{\Psi_j}(x_{r_2,g}, x_{r_3,g})) =$

$$\begin{cases} 1, & \text{if random}[0,1] < F \wedge (D_{\Psi_j}(x_{r_2,g}, x_{r_3,g}) = 1) \\ 0, & \text{otherwise} \end{cases}$$

$$D_{\Psi_j}(x_{r_2}, x_{r_3}) = \begin{cases} 0, & \text{if } x_{j,r_2} = x_{j,r_3} \\ 1, & \text{otherwise} \end{cases}$$

      $j \leftarrow j + 1$
    end for
    $rand_i = random[1, D]$
    for $j = 1$ to D **do**

$$u_{j,i} \leftarrow \begin{cases} v_{j,i}, & \text{if}(random[0,1] \leq CR) \text{ or } j = rand_i \\ x_{j,i}, & \text{otherwise} \end{cases}$$

      $j \leftarrow j + 1$
    end for
    if u_i satisfies the constraints **then**
      Evaluate query processing cost and materialized view maintenance cost of G for u_i, x_i
      if $u_i \prec x_i$ **then**
        $x_i \leftarrow u_i$
      else
        if $x_i \not\prec u_i$ **then**
          $NP \leftarrow NP + 1$
          Append u_i to the population
        end if
      end if
    end if
    $i \leftarrow i + 1$
  end for
  if $NP > N$ **then**
    Keep the elite N members from NP population in the list and discard the rest
    $NP \leftarrow N$
  end if
  $g \leftarrow g + 1$
until $g < g_{max}$
Return Non-dominated solutions from the final population list

3.3.4 Multi-objective DE with binary encoded data for view selection : MODE-BE

For using binary encoded data in multi-objective DE, while generating the mutant vector, the forma basis may be used as discussed in the Section 3.3.2. In solution representation of the problem of data warehouse view selection for materializing, each solution have been defined as a string of bits.To adapt the solution representation with multi-objective DE, each population vector of the evolutionary system is considered as the solution string of bits where each bit represents a decision variable of a population vector. Thus a set of NP initial solutions $x_1, x_2, \cdots, x_{NP}$ that satisfy the space constraint are generated for a given MVPP DAG G. Using this population of size NP, in each generation of an evolutionary process g, against each solution vector $x_i, i = 1, 2, \cdots, NP$, a mutant vector $v_{i,g+1}$ is to be generated. In solution representation presented in 3.2.5, each bit is in single dimension that may have value either 1 or 0 depending on whether a particular view is selected or not. Therefore, restricted-change mutation operation [31] may be applied for this solution representation. Thus the mutant vector $v_{i,g+1}$ is generated as expressed in Equations 3.15 and 3.16. The trial vector $u_{i,g+1}$ is formed by crossover as expressed by Equations 3.7 and 3.8.

To adapt the problem of view selection to materialize in data warehouse, the query processing cost and materialized view maintenance cost of the given MVPP DAG G, $Q_G(x_{i,g})$, $Q_G(u_{i,g+1})$ and $U_G(x_{i,g})$, $U_G(u_{i,g+1})$ are computed using Equations 3.1 and 3.2. If $u_{i,g+1} \prec x_{i,g}$ then $x_{i,g+1}$ is set as $u_{i,g+1}$, else if $x_{i,g} \prec u_{i,g+1}$ then $u_{i,g+1}$ is discarded. Otherwise, in case $u_{i,g+1} \not\prec x_{i,g}$ and $x_{i,g} \not\prec u_{i,g+1}$, $u_{i,g+1}$ is appended to the population for next generation $g+1$. Thus the population may go on increasing. To control the population growth in each generation of DE, when the population size touches a limit, NP elite solution population that maintains diversity in the solution population are filtered out as discussed in Section 3.3.4.1. This evolutionary process is continued till it reaches a maximum number of generation specified, say g_{max}. The dominated solutions in the final population are then deleted to return the non-dominated solutions of the problem. This version of multi-objective DE using binary encoded solution population is henceforth referred as MODE-BE.

3.3.4.1 Promoting elitism and diversity in solution population

To control the population size by keeping the diversity in solution population in intermediate generations of DE, different techniques are used. In [56], the fast non-dominated sorting and ranking selection scheme of NSGA-II [32] is incorporated. In Pareto based multi-objective DE suggested by Xue et al. in [57] Pareto based evaluation and selection is done using NSGA-II algorithm. In both the techniques, the individuals within each rank of Pareto-front that reside in the least crowded regions are given more priority for including into the population for further iterations. In NSGA-II, a *crowding distance* metric [32] in objective function space is calculated to determine the crowding density of the solutions of a partic-

ular *Pareto rank*. To allow individuals in lower rank to enter the next generation, giving importance in keeping diversity among ranks, Xue et al. [57] use another parameter to specify how close the solution is in its surrounding solutions' objective functions space. This parameter is used for fitness ranking among solutions in terms of objective functions to select the best population from both parents and offspring. There are few other techniques which also use crowding distance in objective functions space and Pareto ranking among solutions (as suggested by NSGA-II) to select solution population for subsequent generations [58, 59].

Algorithm 5: Selecting elite N solutions by NSGA-II based non-dominated sorting and SMC based diversity in solution space in Multi-objective DE using Binary Encoded Data

Require: NP, Solution population $x_1, x_2, \cdots x_{NP}$, N $(N < NP)$
Ensure: Elite N solution population
1: Do non-dominated sorting of the population $x_1, x_2, \cdots x_{NP}$ and assign each solution population vector with corresponding Pareto-front
2: For each $x_i, i = 1, 2, \cdots, NP$ compute maximum SMC distance to other solutions in the solution population Max_i and assign it to x_i
3: Sort the NP solution population in ascending order of their Pareto-front and descending order of Max_i
4: **Return** The top N solution population from the sorted list.

In the problem of selecting views to materialize in data warehouse, diversity among non-dominating solutions in terms of their constituent views are preferable as in that case the set of solutions may get representations from a larger number of views of the MVPP graph. Therefore, in our approach, diversity in solution space is promoted with necessary elitism. The solutions of intermediate generations are ranked according to their Pareto dominance levels as discussed in [32]. For each solution of the population, the maximum distance to other solution vectors in the population is measured by *Simple Matching Coefficient (SMC) distance* measure [72]. The SMC distance measure is used because the representation of a bit as 1 or 0 does not mean any preference over each other in the solution string representation. The solution population is then first sorted in ascending order of their Pareto fronts and then on descending order of their maximum distances to other solutions in the population. From the doubly sorted solution population, the top NP solutions are selected as next generation population. Thus the density in solution space is used instead of crowding distance in objective function value space to promote diversity considering that: if S is a vector and $f(S)$ is a scalar valued function on S, then for vectors S_i, S_j, S_k and S_l, where $S_i \neq S_j \neq S_k \neq S_l$, if $|f(S_i) - f(S_j)| > |f(S_k) - f(S_l)|$, then it does not imply that $||S_i| - |S_j|| > ||S_k| - |S_l||$. This is also evident in our experimental results as presented in Figure 3-5.

3.3.5 Complexity analysis

In each generation of DE a loop over NP is conducted that contains a loop over the dimensions D of the population vector. That is, the mutation and crossover operations are to be performed for each dimension for each population vector. Therefore, the number of operations in *DE/rand/1/bin* is proportional to the total number of basic loops executed till it reaches the termination criteria. As the algorithm stops after a specified number of generations, g_{max}, the run time complexity is $O(NP.D.g_{max})$ [73]. As the solution space of the problem increases exponentially with the dimensions of the problem, the space complexity usually increases with the size of the problem. However, the space complexity of DE is relatively low compared to some other competitive optimizer [73].

In multi-objective DE for M objectives, to control the population sizes in intermediate generations within N, first non-dominated sorting is done as suggested in NSGA-II. The overall complexity of this non-dominated sorting for M objectives is $O(MN^2)$ [32]. Secondly, for computing SMC based maximum neighborhood distances of each of the N solutions, complexity is $O(N^2)$. Then for two independent sorting of at most N solutions, complexity is $O(2NlogN)$. Thus these three processing are governed by non-dominated sorting complexity $O(MN^2)$. Therefore, the overall complexity may be expressed as $O(g_{max}(N.D + MN^2))$ i.e. $O(g_{max}(N(D + MN)))$. For a series of experiments, dimension D and objectives M (i.e $M = 2$ in this application with one constraint for space) are held constant. Thus the overall approximate run time complexity is $O(g_{max}.N^2)$.

3.3.6 Convergence

The DE algorithm is a non-deterministic global optimization algorithm. Therefore it has weaker convergence theory than deterministic global optimizers like Simplex algorithm for linear programming, Branch-and-Bound algorithm for mixed integer linear programs, polynomial integer programs etc.. Though in general cases how many iterations a deterministic global optimizer will take to converge to global optimality is not known, in certain cases a deterministic global optimizer converges to global optimum within a certain number of steps. But in case of non-deterministic global multi-objective optimizers the proofs of convergence are basically only to prove that a particular algorithm converges (asymptotically) to global optimum solutions with probability 1 [63, 64, 74]. The proof of convergence to Pareto optimality with probability one is only to indicate that - if the algorithm runs for a very long time (leading to infinity if the search space is infinite) it will eventually search the entire space and return the global optimum. Though this proof is not very useful, yet, in my literary search I found that it is the only theoretical proof available that may be applicable for multi-objective Differential Evolution algorithm using *elitism*. One such proof of convergence forwarded by Villalobos-Arias et al. in [63] is briefly discussed below proving that meta-heuristic multi-objective optimization algorithm converges when elitism is adopted.

If the function of a multi-objective optimization evolutionary algorithm is modeled as Markov chain $\{X_k : k \geq 0\}$ where each solution is a string of bits of length l and S is the set of all such possible population vectors, then let $i \in S$ be a state of the algorithm having a set of population such that i can be represented as $i = (i_1, i_2, \cdots, i_n)$ of n possible entries of solutions and transition probabilities from one state to another use uniform mutation rule. Now for the Markov chain, let the transition probability is given by $\mathbb{P}(X_{k+1} = j | X_k = i)$. The algorithm is said to be converging to the Pareto optimal set $\mathscr{P}$ with probability 1 if :

$$\mathbb{P}(\{X_k\} \subset \mathscr{P}) \to 1 \; as \; k \to \infty. \tag{3.17}$$

In case of elitist multi-objective evolutionary algorithms, after each iteration an elitism operation is applied to the population that accepts a new state if there is an element in the population that are improved as a better solution in the elite set. This state change of elite sets for elitism is represented as $\hat{i} = (i^e; i) = (i_1^e, i_2^e, \cdots, i_r^e; i_1, i_2, \cdots, i_n)$, where $i_1^e, i_2^e, \cdots, i_r^e$ are the elite members of that state where $r \leq n$ and the number of Pareto optimal solutions of the problem is more than or equal to r, i.e, $|\mathscr{P}| \geq r$. In such cases where elitism is used, in expression 3.17 X_k can be replaced by X_k^e and therefore if $X_k = i$, we may express $X_k^e = i^e$.

Villalobos-Arias et al. in [63] defined two states of the population in intermediate generations of these algorithms as *inessential state* denoted by I and *essential state* denoted by E. A state of resultant solution population of these algorithms i is called *inessential state* if there exists a state j such that if i leads to j, expressed as $i \to j$, but j does not lead to i, i.e $j \nrightarrow i$. Otherwise the state i is called an *essential state*. It is also defined that $S = E \cup I$. It has been proved that -

$$\mathbb{P}(X_k \in I) \to 0 \; as \; k \to \infty. \tag{3.18}$$

This also shows that the states of which elite set contains elements that are not Pareto optimal are *inessential state*.

The elitist multi-objective evolutionary algorithms are designed in such a way that if there is a state $(i^e; i)$ in which the elite set $i_{s1}^e, \cdots, i_{sk}^e$ contains elements that are not Pareto optimal then there is a state $(j^e; j)$ of solutions in a later generation such that all Pareto optimal points of the elite set i^e are in j^e and replaces the other non Pareto optimal points of i^e with corresponding $j_{s1}^e, \cdots, j_{sk}^e$ and thereby all elements of j^e becomes Pareto optimal. Villalobos-Arias et al. [63] also show that (i^e, i) can pass to the state (i^e, j) with probability greater than zero. Therefore, elitism operation can be applied to pass from (i^e, j) to (j^e, j), i.e, it implies that $\hat{i} \to \hat{j}$. But if all the elements in the elite set of a state are Pareto optimal, then any other state that contains non Pareto optimal population in its elite set is not accepted. Therefore, $\hat{j} \nrightarrow \hat{i}$ and hence $\hat{i}$ is an *inessential state*. Thus by expressions 3.17 and 3.18, $\mathbb{P}(\{X_k^e\} \subset \mathscr{P}) = \mathbb{P}(X_k \in E) = 1 - \mathbb{P}(X_k \in I) \to 1 - 0 = 1$ as $k \to \infty$. This proves that if elitism is used, multi-objective evolutionary algorithm converges.

3.3.6.1 Convergence of MODE-BE generated solutions in view selection

Deb et al. in [32] suggest a convergence metric denoted by γ for measuring the extent of convergence by an algorithm to a known set of Pareto optimal solutions. In real life situations non-deterministic multi-objective optimization techniques are applied on those problems where the Pareto optimal solutions are not known. Therefore, this metric, γ can not be used for any arbitrary problem. But the testing of a multi-objective optimization algorithm is generally conducted on problems having a known set of Pareto optimal solutions or with finite solution space. In convergence measure γ, a set of uniformly spaced solution from the known set of Pareto optimal solutions are considered and then the Euclidean distances from each solution yielded by the algorithm (to be tested) to all the selected solutions in the true Pareto front are measured. For each solution obtained by the algorithm, the minimum Euclidean distance to points on the true Pareto front are computed. The average of these minimum Euclidean distances is used as the convergence metric γ for the algorithm. The smaller the value of the metric γ means the better the convergence towards the true Pareto front. In case all obtained solutions lie exactly on the chosen solutions of the Pareto front, the value of γ becomes zero. Therefore, even when all solutions converge to the true Pareto front, the convergence metric may not have a zero value because all solutions may not lie exactly on the chosen points of the front.

Table 3.2: Convergence metric γ of solutions produced by NSGA-II and MODE-BE in different test problems.

Binary coded NSGA-II (γ)					MODE-BE (γ)
ZDT1	ZDT2	ZDT3	ZDT4	View selection	View selection
0.000894	0.000824	0.043411	3.227636	0.071569735	0.05316552

It has been observed that in case of meta-heuristic multi-objective optimization where transition probabilities from one state of solution population to another state use uniform mutation rule, the algorithm converges only if elitism is used. For experimenting with MODE-BE, the convergence metric γ for solutions generated by MODE-BE are computed with respect to the set of Pareto optimal solutions generated by NSGA-II [29] on our experimental data that have been presented in Section 3.4. The convergence metric γ found to be 0.05316552 when the minimum Euclidean distances in objective function space between each solution yielded by MODE-BE and already obtained Pareto optimal solutions by NSGA-II are normalized (see Table 3.5) for view selection problem for materializing in data warehouses. In our experimentation on view selection problem with Binary-coded NSGA-II, the convergence metric γ is found to be 0.071569735. The value of γ by Binary-coded NSGA-II for test problems ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 suggested by Zitzler et al. in [75] are found to be 0.000894, 0.000824, 0.043411, 3.227636 and 7.806798 respectively by Deb et al. [32]. With the convergence metric γ of 0.05316552 for selecting view for materializing by MODE-BE is therefore considered to be acceptable as asymptotic to the true Pareto front.

3.4 Experimentation and Observations

In this Section I report my experimentation and analysis of solutions by implementing MODE-BE for generating non-dominated solution sets of views for materializing in data warehouses.

3.4.1 The test-bed used

The effectiveness of multi-objective DE using binary encoded data in handling the problem of view selection for materializing in data warehouses, is studied using TPC-H [15] benchmark data warehouse. The TPC-H schema is built and loaded in Oracle10g RDBMS for experimentation. The TPC-H benchmark queries are generated using *qgen* utility of TPC-H framework and three queries are selected and reconstructed with minor changes for constructing a test MVPP DAG which go well with our application. The query access frequencies of the selected queries in a specific period on the data warehouse are assumed as 20, 10 and 30 for first, second and third query respectively. During the considered period, it is assumed that the base tables were updated two times. Number of rows in the considered base tables of TPC-H framework are presented in Table 3.3. The sizes of candidate views for selecting are computed using *EXPLAIN PLAN* utility of Oracle10g, and are presented in Table 3.4. In our experimentation, to make the cost computation process simpler, it has been assumed that the space requirement of each row of the candidate views are equal because the differences in space requirement for different rows of different relations are comparatively very small considering the space availability for materializing the views in a data warehouse.

3.4.2 Control parameters

The three main control parameters of DE algorithm are: the mutation scaling factor F, the crossover constant CR and the population size NP. Storn and Price in [28] mention that the value for NP could be chosen around $10 \times D$, where D is the dimension of the problem. In our experimentation it is observed that when NP is chosen to be in between 100 to 250 the results are found better for analysis. The results presented in this chapter are based on 242 random initial solution population that produces the best convergence values most of the times in a series of experimentation. In [28], Storn and Price suggest effective range of F between 0.4 and 1. The crossover control parameter CR controls how many parameters are expected to change in a solution vector. Low value of CR means a small number of parameters are to be changed in each generation, whereas high value of CR i.e. near 1 means the mutant vector inherits most of the dimensions [73]. In our experimentation, significant results are found when CR is chosen as 0.6 and F is chosen to be 0.5. The number of iterations g_{max} in DE is fixed depending upon the complexity of the objective functions. In an instance of experimentation, result of which is discussed in the next sub-section, the g_{max} value is set as 40.

Table 3.3: Base tables used in our experimental MVPP

Table	Size (in rows)
Parts	26260520
Partsupp	112800000
Supplier	1430000
Nation	2650
Region	475

Table 3.4: Views generated in our experimental MVPP

View	Size (in rows)
v_1	34020
v_2	26260
v_3	51
v_4	5
v_5	688000
v_6	1005
v_7	95
v_8	4977
v_9	10920
v_{10}	8254740
v_{11}	65985
v_{12}	106
v_{13}	90624
v_{14}	260352
v_{15}	54717984
v_{16}	34000

3.4.3 Observations

In a particular instance of experimentation using the MVPP DAG presented in Figure 3-1 using data presented in Table 3.3 and 3.4 to select a set of views, we put a space constraint for materializing as maximum 80000 rows only. Initially 242 non-duplicate solutions were generated which satisfy the space constraint. In Figure 3-2 the distribution of the initial random solution population generated for the instance of experimentation is presented in objective function space. The MODE-BE is applied on this initial set of solutions for 40 iterations, i.e. $g_{max} = 40$, with other parameters set as discussed in the Section 3.4.2. By applying maximum dissimilarity based measure in solution space for filtering significant non-dominated solutions produced by MODE-BE, it has been observed that the non-dominated solutions generated by MODE-BE are with less total cost function values than that of NSGA-II yielded solutions for the same set of data for 40 iterations. It has been observed in Figure 3-3 that the solutions yielded by both NSGA-II and multi-objective DE algorithms are distributed in similar curve in the objective function space. The convergence metric γ of the MODE-BE yielded solutions are presented in Table 3.5. The γ is found to be 0.05316552 when the minimum Euclidean distances in objective function space between each solution yielded by MODE-BE and already obtained Pareto optimal solutions by NSGA-II are used for view selection problem for materializing in data warehouses. The convergence metric γ in case of NSGA-II yielded solutions in this experimentation is found to be 0.071569735. Therefore, MODE-BE may be stated as better converging towards Pareto front than NSGA-II in materialized view selection.

In the considered instance of experimentation with multi-objective DE,

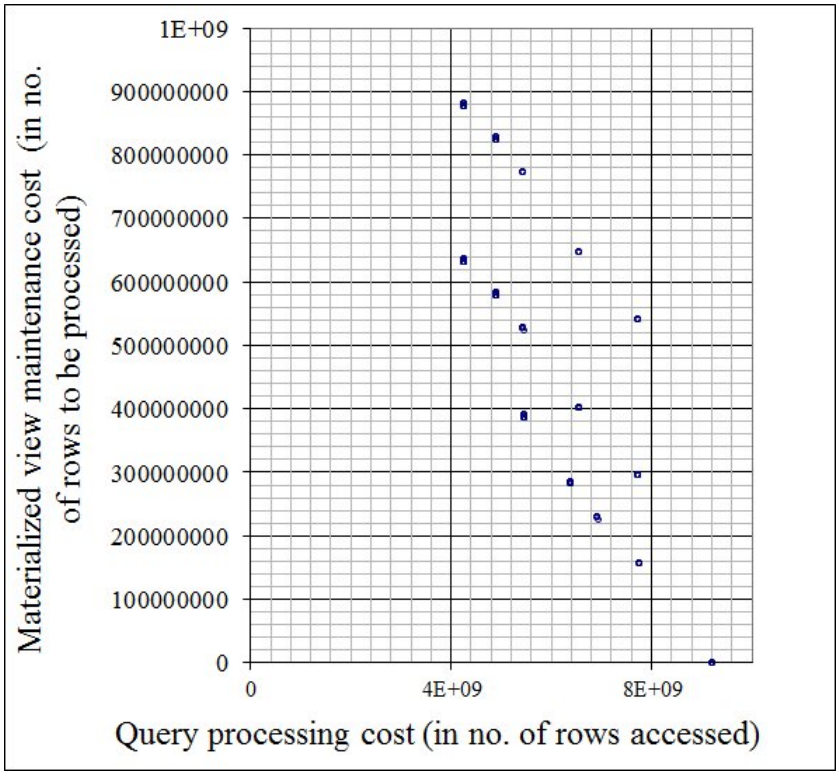

Figure 3-2: Randomly generated 242 solutions

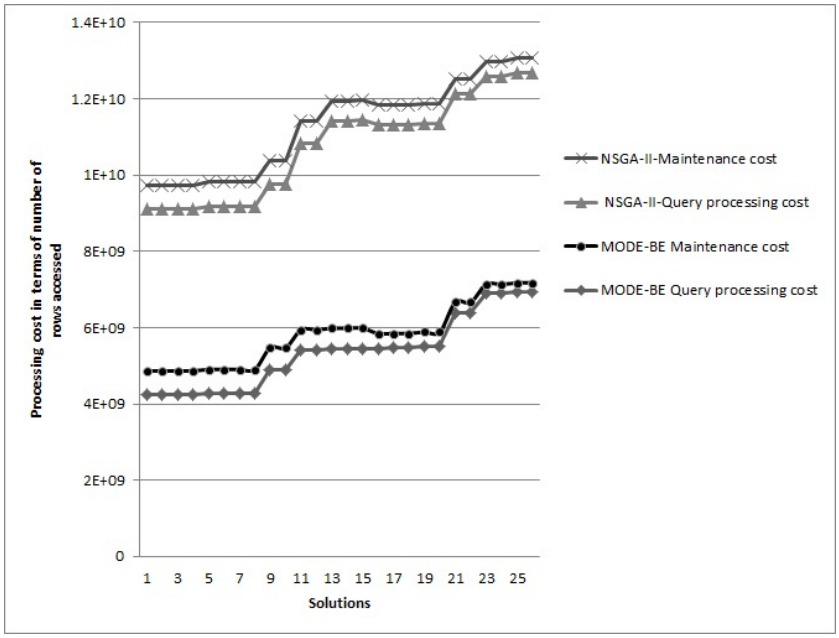

Figure 3-3: Distribution of costs by obtained non-dominated solutions after 40 iterations

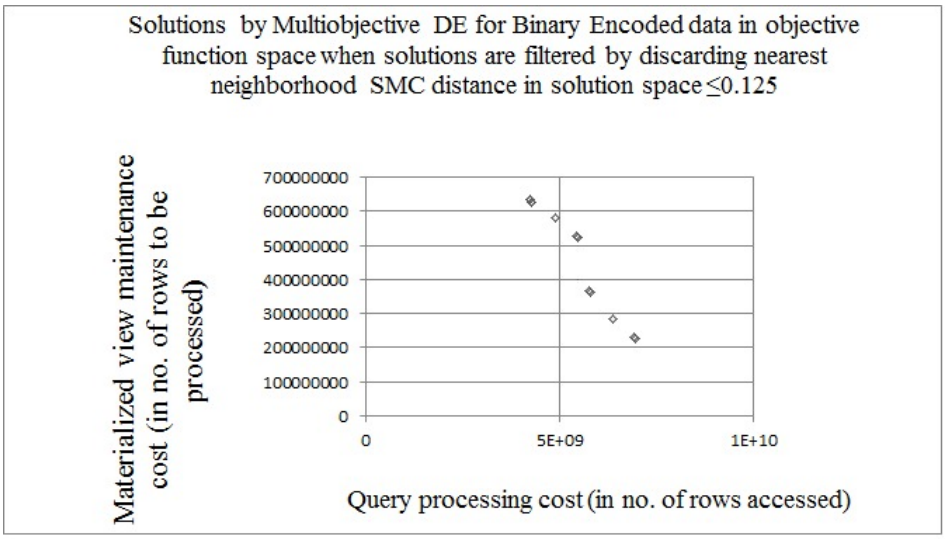

Figure 3-4: Distribution of significant representative solutions in objective space

the mean of maximum SMC measure based distances of each non-dominated solution with respect to other non-dominated solutions is found to be 0.46305. When the solutions having maximum distance to all other solutions in the population less than that of the mean of the maximum distances of all non-dominated solutions to other non-dominated solutions in the population are discarded, approximately half of the total non-dominated solutions were filtered out. Another observation is that there are many non-dominated solutions either in high crowding density region of objective function space or have the same objective function values but are distributed distantly in their vector space (Figure 3-5). For example, three non-dominated solutions generated in our experimentation $\{v_1, v_2, v_9, v_{12}\}$, $\{v_1, v_2, v_7, v_8\}$ and $\{v_1, v_2, v_4, v_7, v_9, v_{12}\}$ yield same objective function values, i.e., the overall query processing cost and materialized view maintenance cost of 4234819870 rows and 632619684 rows respectively within the specified space requirement constraint. In such kind of situations for maintaining diversity in objective function space instead of solution space, other criteria like minimum space requirement or maximum number of views in the solution set can be used for breaking ties. Thus if only crowding density in objective function space is used for filtering out representative solutions, some otherwise significant solutions in selecting views for materializing in data warehouse may be lost.

3.5 Discussion

An approach for selecting views for materializing in data warehouse using multi-objective differential evolution algorithm using binary encoded representation of solutions has been presented. The DE algorithm is basically used for real parameter optimization. Though Gong et al. in [31] present the use of forma analysis to exploit usage of DE for discrete optimization problem, it has not been used so far for multi-objective optimization problem. We have implemented forma analysis based multi-objective DE for selecting views to materialize in data warehouse for

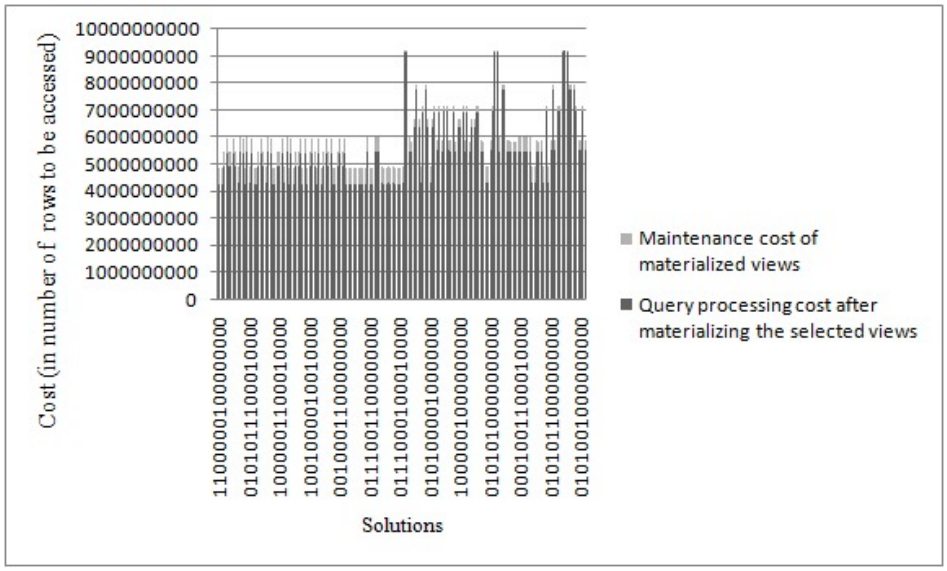

Figure 3-5: Objective function values of non-dominating solutions

efficient OLAP query processing. It has been observed that the solution quality of MODE-BE generated solutions in this problem are somewhat better than that of NSGA-II with respect to convergence property and total cost function values.

The basic assumption in our approach in selecting views to materialize in data warehouse is that the frequent OLAP queries and the intermediate views generated while processing the queries on a data warehouse in a specific period of time will resemble with future frequent OLAP query processing on the data warehouse. Generally in multi-objective optimization, decision maker selects one or more non-dominating solutions from a set of large number of non-dominating solutions by using suitable criteria for their application. In our experimentation it has been observed that there may be multiple number of non-dominating solutions very close in their objective function space but are different in their vector space (Figure 3-5). In such cases solutions containing larger number of views may be preferred as there is a chance that it may cater well the efficient processing of future queries. The use of space requirement in materialized views may be defined as another objective function for this problem as discussed in [26]. In our approach we used space requirement for materializing the selected views as a constraint but it is explained that it may be used as a selecting criterion while selecting solutions from multiple number of non-dominated solutions. We propose to use number of views in the solution sets and space requirement in materializing the selected views as objective functions in our ensuing work.

Though there are several approaches for handling the view selection problem as multi-objective optimization problem [14, 26], the problem is not yet handled by using multi-objective DE algorithm. The approach suggested by Michael Lawrence in [26] presented a multi-objective evolutionary algorithm for view selection problem where the basic Genetic algorithm was used extensively. In this work a comparative analysis of MODE-BE in materialized view selection with NSGA-II algorithm based approach is presented. The solution quality and performances with respect to other evolutionary algorithmic approaches like in [26] and

stochastic algorithmic approaches like simulated annealing algorithmic approach suggested in [14] are having prospective scope of study. The approach presented here is a scalable one in terms of number of queries and candidate views for materializing. But as analysis of solutions with very high dimensional vectors often becomes too complex, and at present the input multiple query processing plan used in this experimentation are generated off-line by a separate procedure only, therefore a simple MVPP DAG found to be sufficient for comparative analysis of view selection algorithm applied. Moreover it is expected that Transaction processing Performance council (TPC) [15] will come-up with voluminous benchmark test data set for this kind of experimentation on data warehouse for future research.

The problem and solution model reported in this chapter define views as some derived functions or relations on some normalized relations or tables. This model does not support databases with very little indexing capabilities as used in recently developed Big data [76] framework based data warehousing. In next chapter the view selection problem has been defined for Big data query processing framework and our attempt of handling this problem by evolutionary algorithms has been reported.

Table 3.5: MODE-BE generated solutions and Convergence metric

Solution	Query processing cost (rows to be accessed)	Materialized view maintenance cost (rows to be accessed)	Normalized minimum Eucliden distance to Pareto optimum	Convergence metric γ
0001000000000000	6947916700	225600000	0.002249476	0.05316552
0001001000010000	6947880700	225606250	0.002249476	
0011010000000000	6905472100	229845600	1.000044766	
0011000000010000	6905445600	229849760	0.006367362	
1001001000000000	6379735480	282418122	0.006367362	
1001000000010000	6379708980	282423422	0.006367362	
0101000000010000	5503561600	383168420	0.002249476	
0101011000000000	5503523700	383169560	0.006367362	
0111001000000000	5461143500	387407580	0.006367362	
0111011000010000	5461117000	387413070	4.47662E-05	
1101001000010000	5460590780	387465502	0.006367362	
0001000010010000	5459514090	523280522	0.002249476	
0001010010000000	5459449690	523286962	0.011161634	
0001010100010000	5459340490	523308802	0.012999601	
0011001010000000	5417069490	527524982	0.026238337	
0011001100000000	5416960290	527546822	0.026238337	
1001000010000000	4891332870	580098644	0.026238337	
1001001100010000	4891223670	580120484	0.026238337	
0101001010000000	4277781190	628323552	0.002249476	
0101011010010000	4277726290	628329042	0.011161634	
0101001100000000	4277671990	628345392	0.002249476	
0101010100000000	4277617090	628350882	0.012999601	
0111000010000000	4235346090	632567062	0.026238337	
0111001100010000	4235236890	632588902	0.026238337	
1101000010010000	4234819870	632619684	0.098522132	
1101000100010000	4234710670	632641524	0.026238337	

Chapter 4

Materialized View Selection by Evolutionary Algorithm for Big Data Query Processing

4.1 Introduction

Data warehouses traditionally support structured data because they are tied with operational or transactional systems for analytical processing by financial analysts and business lines for making decision on business strategies. However, with the advent of Big data, the conventional data warehouse concept is now changing because organizations are going for expanding and modifying the data warehouse for relevance in Big data environment of huge volume of semi structured or *key-value paired* [76] data in highly distributed computational framework.

A computing paradigm called MapReduce have been introduced in Big data [16]. In MapReduce model of Big data systems, computation tasks are broken up into units that can be distributed around a cluster of commodity hardware and server class hardware for providing cost-effective processing with scalability. The MapReduce system was designed and is best suited for handling big and semi structured data such that each split of the data for MapReduce computation is from a single big table or file instead of large number of small files. This is because in Distributed File System (DFS) based Big data processing such as *Hadoop Distributed File System* (HDFS), the block size is kept as at least 64MB (or multiple of that), as in this frame work of processing, a very large number of distributed commodity memory is available and smaller file size of less than the block size imposes unnecessary overhead. The technologies used in Big data warehousing organizes data into tables as a mean for providing structure to data stored in DFS and supporting row-level update like *delete, update* etc. on big column oriented table of key-value type storage. These technologies support *collection data type* columns STRUCT, MAP and ARRAY sacrificing *normal form* for higher processing throughput [18]. Again, as these technologies support collection data type,

therefore they are designed with very limited indexing capabilities. Hence, in Big data systems, for analytical processing queries, the use of indexes and keys in the relations are very limited. And therefore, common or shared sub-expression results of frequent queries may be utilized as materialized views without the need of designing a common optimized query execution plan based on indexes and relations for the considered set of queries.

4.1.1 Materialized views and materialized queries in Big data

Big data analysis by Distributed File System (DFS) is a cost-effective framework that binds very large data sets in a cluster of computers into a pool for distributed processing [77]. The imposed programming model termed as *MapReduce* breaks-up computation tasks into smaller jobs for distributing them around the data created by splitting large amount of data into a cluster of commodity computer hardware for distributed processing [17, 77]. The distributed file system (DFS) used in Big data by *Apache* [78] is termed as Hadoop Distributed File System (HDFS) [17, 79]. For Data Warehousing applications in HDFS, Hadoop [79] provides a technology called *Hive* and an SQL like language called *HiveQL* [80]. In Big data the total MapReduce cost against generating responses of a set of queries depends on MapReduce splits. MapReduce costs are thus involved in creation of temporary views while processing queries. To make query responses faster, if these temporary views are saved for future query processing, MapReduce cost is also to be incurred for updating the views. The problem of view selection for materializing is that - a set of views, generated while processing a set of queries are to be selected for materializing, so that if this set is materialized or saved, the total query MapReduce cost and MapReduce cost for maintaining the materialized views are minimum.

Julian Hyde in [19] proposes an extension to materialized views called *Discardable, In-Memory Materialized Query - DIMMQ for Hadoop*. DIMMQ proposes that the resultant data-set of some frequent queries reside in the memory of one or more nodes in the cluster. Discardable means that the system can remove the in-memory materialized queries when they are not used for a long time. Here it is proposed that some sub-queries may be saved in the memory of hardware cluster with mapping to their resultant data in disk. But even here the MapReduce overheads for job submission and job scheduling remains along with the maintenance cost for refreshing the mapping between in-memory queries and the disk data. Therefore whether it is materialized views or in-memory materialized queries, a sub-set from the candidate set of views or queries are to be selected for materializing such that all related costs are minimized.

4.1.2 View selection for materializing in Big data

In Big data framework, a query is executed by accessing data spread over a cluster of hardware storage or data-nodes as distributed file system (DFS) [79] by MapReduce jobs. Therefore the query processing cost is not just the cost of accessing rows of base tables stored in disk. The DFS overhead of distributing data into data-nodes, mapping and tracking of processing, and then reducing the results also are involved. As every complex query may have several sub-queries with multiple number of aggregation functions, therefore either these sub-queries may be materialized in memory of hardware cluster with mapping to their resultant data in disk or the intermediate result of sub-queries may be materialized in disk as materialized views. Thus by materializing these intermediate views, the MapReduce cost of repeated processing of these sub-queries can be avoided. But the DFS overhead cost for materializing these views and refreshing them periodically are still to be incurred. The materialization of temporary views also needs space in the hardware cluster. Therefore a system may be designed to recommend a set of intermediate views so that if they are materialized, the total query processing cost savings for the selected set of frequent queries is maximized with minimized materialized view refreshing cost and space cost. Therefore the materialized view selection problem is defined as an optimization problem.

If, the number of queries considered as frequent queries increases, then the number of candidate views or sub-queries for materializing also increases. Different query processing costs and other associated costs for different combination of views are independent of the total number of views selected. Thus the solution space increases exponentially with increased number of queries and underlying views considered.

4.1.3 Contribution

In this chapter, design of a system is proposed for finding solution set of views for materializing to optimize query processing cost, materialized view maintenance cost, and storage for materialized views from views generated while processing a set of queries on Big data warehousing. The problem is defined as a multi-objective optimization problem for finding non-dominated solution set of views using Multi-objective Differential Evolution algorithm and Non-dominated Sorting Genetic Algorithm-NSGA-II [32]. Here, *forma analysis* based multi-objective DE for binary encoded data, termed as MODE-BE, presented in Chapter 3 is modified and implemented for selecting views for materializing in Big data framework based data warehouse by promoting diversity in solution vector space. In NSGA-II the diversity of large number of solutions are promoted by computing crowding distances between solutions in objective function value space. The NSGA-II is also implemented on this problem to analyze performances between NSGA-II based systems and MODE-BE based recommendation systems for materialized view selection in Big data management.

The problem of selecting views for materializing in Big data warehousing is defined in Section 4.2. In Section 4.3, materialized view selection in Big data framework has been defined as a multi-objective optimization problem. An improved version of Multi-objective Differential Evolution algorithm with Binary Encoded solution representation for materialized view selection, MODE-BE, and implementation of Non-dominated sorting Genetic Algorithm, NSGA-II, in materialized view selection for Big Data warehousing are presented in Section 4.4. In Section 4.5, the experimentation process of implementation of algorithms and the test data used in the experimentation are presented along with an analysis on obtained results. Finally in Section 4.6 concluding discussion and perspectives are presented.

4.2 The Problem of Selecting Views for Materializing in Big data Framework

To make query response faster, a set of views are to be selected for materializing to minimize total query response cost of a set of frequent data warehouse queries with optimum maintenance cost or updating cost of the materialized views. Hive uses the advantage of Big data framework's scale out and robust capabilities for data storage and processing on large number of commodity hardware. HiveQL enables to do ad-hoc query processing, summarization and data analysis on massive data easily [80]. The DFS and MapReduce paradigm are used for working with massive data for storage and analysis at Internet scale which is otherwise unmanageable by conventional data processing with database management system. HiveQL [80] query processing on Hadoop version 1 often had to submit number of MapReduce jobs to complete a query processing. With Hadoop-2 and *Tez* platform the cost of job submission and scheduling is minimized by removing the restriction that the jobs are to be done only by Map and Reduce for all kind of processing [20]. But in general, for Big data, processing standard is by MapReduce [79]. Though Map tasks write intermediate output to the local disks, input to a single Reduce task is normally the output from all Map tasks. The Map outputs are transferred across the network to the node running Reduce tasks and the merged output is to be passed to the user-defined Reduce functions. Thus the intermediate MapReduce result sets are needed to be stored in DFS and thereby the MapReduce jobs in the system degrades the system performance. Also submitting jobs and scheduling them across the DFS adds extra costs [20].

4.2.1 The cost model and problem definition

The cost model to be used for handling materialized view selection problem for Big data system is different from cost models used in approaches used for conventional Client-Server architecture with RDBMS based data warehousing. The main reason behind this is that, in conventional RDBMS based system the data access pattern is mainly dominated by "seeks" and "seek time", whereas in Big data DFS or in

similar distributed framework, the data access pattern is mainly dominated by data transfer rate and MapReduce costs. The MapReduce paradigm is designed to analyze massive amount of data in batch fashion unlike the traditional RDBMS where data-set has been indexed to deliver low-latency seek and update time [77]. As for increased size of data, a bigger sized commodity hardware cluster may be used, therefore the performance of MapReduce functions are independent of size of the data or rows to be accessed.

The MapReduce overheads are composed of data transfer cost of transferring data into number of data nodes across the DFS cluster, running *job trackers* and *task trackers*, creation of Mapppers and Reducers and substantial overheads in job submission and scheduling. In Big data management DFS, block size and split size are fixed. Therefore, in Big data/MapReduce framework, a small number of large files are better than large number of small files [17]. This means that in case of Big data based data warehousing smaller number of bigger views are to be preferred for materializing. This criterion is not applicable in case of traditional RDBMS based data warehouse's materialized views. The different costs and benefits that are to be optimized for materializing views to enhance query processing in Big data warehousing are formally defined below.

4.2.1.1 Query processing cost

The total query processing cost of a set of frequent queries may be considered as the total MapReduce overheads for executing the set of queries. If the results of some sub-queries and aggregate functions used in these queries are materialized or saved, then in subsequent execution of the queries, MapReduce overheads of executing these sub-queries or views are saved. If a set of sub-queries of a set of frequent queries are processed and composed as views and materialized in Big data DFS, the query processing cost of the set of considered queries may be defined as Definition 19.

Definition 19. *For a set of n number of frequent queries $Q = \{q_1, q_2, \cdots, q_n\}$ on a data warehouse, where V is the set of m intermediate views generated by Q, if $V' \subseteq V$ is the set of views $V' = \{v_1, v_2, \cdots, v_p\}$ that are materialized, the total Big data query processing cost can be defined by the following expression.*

$$C^Q_{V'} = C^Q_{\varnothing} - \sum_{i=1}^{p} M_{v_i} \tag{4.1}$$

where $C^Q_{\varnothing} = \sum_{i=1}^{n} M_{q_i}$ is the total query processing MapReduce cost of Q without materializing any view, and $\sum_{i=1}^{p} M_{v_i}$ is the MapReduce cost of processing V'.

4.2.1.2 Materialized view maintenance cost

In analytical processing on DFS based Big data warehouse, there are generally very few occurrences of updating operations. But whenever there is a change in

the base data, the materialized views are to be updated. In case of in-memory query materialization, frequent refreshment is needed as in this case infrequent queries are to be discarded after each fixed period of time [19]. Materialized view maintenance means re-processing the aggregate functions and/or corresponding sub-queries and then updating the views in disks or solid state drives[1]. Thus there will be another set of DFS overheads. The materialized view maintenance cost may be defined as follows.

Definition 20. *For a set of materialized views* $V' = \{v_1, v_2, \cdots, v_p\}$ *for processing a set of queries* Q*, the materialized view maintenance cost may be expressed as*

$$U(V') = \sum_{i=1}^{p} U_{v_i} \tag{4.2}$$

where U_{v_i}*,* $i = 1, 2 \cdots, p$ *, are the maintenance MapReduce overheads for the set of materialized views* $v_i \in V'$*,* $i = 1, 2 \cdots p$*.*

4.2.1.3 Number of views to be materialized and storage space requirements

The storage space requirements for p number of materialized views V' ($|V'| = p$), can be defined as Definition 21 below. In Big data systems, smaller number of bigger tables are preferred (as discussed in Section 4.2.1). That is, $|V'| = p$ is to be minimized with maximized size of the tables.

Definition 21. *If* A_{v_i} *is the storage space required by* i*th materialized view, then the total space required for materializing* p *number of views is*

$$A_{V'} = \sum_{i=1}^{p} A_{v_i} \tag{4.3}$$

4.2.1.4 The materialized view selection problem

Considering the definitions 19, 20 and 21, the view selection for materializing in Big data based data warehousing can be stated as Definition 22 below.

Definition 22. *The view selection for materializing in Big data framework data warehousing for a given set of* n *frequent data warehouse queries* $Q = \{q_1, q_2, \cdots, q_n\}$*, where* V *is a set of* m *views generated while processing* Q*, a set of views* $V' = \{v_1, v_2, \cdots, v_p\}$*,* $V' \subseteq V$ *i.e.* $p \leq m$*, is to be selected such that it minimizes*

1. $C_{V'}^{Q}$ *defined by Equation 4.1,*

[1]Here in addition to *select, join, project* and aggregation functions, MapReduce overhead due to *update* is incurred.

2. *$U(V')$ defined by Equation 4.2 and*

3. *$|V'| = p$ with constraint on minimum value of $A_{V'}$ defined by Equation 4.3.*

In next section we define materialized view selection as a multi-objective optimization problem and present a discussion on applying Multi-Objective Evolutionary Algorithm (MOEA) for solving this problem.

4.3 View Selection in Big data Systems as Multi-Objective Optimization Problem

From the above definitions 19,20, 21 and problem statement in Section 4.2.1.4, for a given set of views, say V, the view selection problem is to find the set V', $V' \subseteq V$, to minimize -

$$\mathbf{Y} = \mathbf{F}(V') \equiv (C^Q_{V'}, U(V'), |V'|) \tag{4.4}$$

such that the constraint on amount of space, $A_{V'}$, for materializing V' is as specified by user.

4.3.1 Simple problem representation

Deb et al. in [60] suggest few important features that must be present in an multi-objective optimization problem for solving by randomized and evolutionary algorithm. According to [60], very importantly the problem should be easy to construct with known dimensions. In our problem definition it is assumed that a set of frequently processed queries are known and thereby the frequent temporary views or sub-queries and aggregate functions triggered on the data warehouse can be derived or known. In our definition this known set of views are defined as V, where the cardinality of V is m, i.e $|V| = m$, and the cardinality of selected views for materializing V', $|V'| = p$. As $p \leq m$, a solution vector may be defined as a string of bits of length m where each of the m dimension may be represented as a decision variable that may be either selected or not selected for materializing.

In our representation of the problem and solution, we have labeled each of the candidate views with a serial number starting from 1 to m for m dimensions of each solution vector. In solution string, the first bit represents the candidate view labeled as the first view, the second bit represents the view labeled as second view and so on. If a view is not selected then the corresponding bit i.e., the corresponding dimension in the candidate solution vector is set as 0 and otherwise, if the view is selected for materializing, its corresponding bit is set as 1.

For two solution strings, say S_0 and S_1 of length m, if $C^Q_{S_0}$ and $C^Q_{S_1}$ are the total query processing costs for a set of frequent query Q having m num-

ber of candidate views for materializing, $U(S_0)$ and $U(S_1)$ are the corresponding maintenance cost of the views if materialized, and if $num(S_0)$ and $num(S_1)$ are number of views selected in solution S_0 and S_1, then *iff* $C^Q_{S_0} \leq C^Q_{S_1}$ *and* $U(S_0) \leq U(S_1)$ *and* $num(S_0) \leq num(S_1)$, then *if* $C^Q_{S_0} < C^Q_{S_1}$ *or* $U(S_0) < U(S_1)$ *or* $num(S_0) < num(S_1)$, then the solution S_0 dominates solution S_1 which is expressed as $S_0 \prec S_1$. If S_0 does not dominate S_1 and S_1 also does not dominate S_0, expressed as $S_0 \not\prec S_1$ and $S_1 \not\prec S_0$, then S_0 and S_1 are two non-dominating solutions of the problem.

Definition 23. *The materialized view selection problem is the problem of finding the set of non-dominating solutions which is an approximation to the true Pareto front of the problem defined by Equation 4.4.*

4.3.2 Scalability

In [60], it has been suggested that the test problem for applying multi-objective optimization problem should be scalable. In our representation of the problem, the solution vectors are of dimension m, where m is the total number of candidate views. As each decision variable is expressed as a single dimension of a solution vector, the solution vector representation is linearly scalable with number of dimensions i.e., value of the variable m. For m number of decision variables of a solution vector, the size of the solution vector space will be 2^m. Thus with increasing dimension in decision vector space, the solution vector space increases exponentially. Due to this, stochastic, randomized or evolutionary algorithms are suitable for handling this problem.

4.3.3 Well defined objectives

For defining a problem as multi-objective optimization problem and solving it by evolutionary or randomized algorithm, most importantly the objectives should be distinct and well defined. In our problem definitions and by Equation 4.4, three objectives are clearly defined. With these three objectives a clearly visible Pareto-front or Pareto-optimal surface may be plotted for getting a clear idea of performance by a multi-objective optimization technique applied on this problem.

In Equation 4.1 and Equation 4.2, M_{v_i} and U_{v_i} for ith view v_i and M_{q_i} for ith query are independent variables. $|V'|$ cannot determine $C^Q_{V'}$ and $U(V')$, and $C^Q_{V'}$ cannot determine $|V'|$ and $U(V')$. Similarly $U(V')$ cannot determine $|V'|$ and $C^Q_{V'}$. Therefore, multi-objective optimization can be designed to introduce controllable hindrance to getting trapped in local optimum.

4.4 Multi-Objective Evolutionary Algorithm for View Selection to Materialize in Big data

It has been observed from our discussion presented in Chapter 3 that multi-objective Evolutionary Algorithms (MOEAs) are suitable for applying in materialized view selection problem. In solution representation for handling this problem, solution vectors have been defined as a string of bits. The single objective DE for binary encoded data as presented by Gong et al. in [31] has been customized and implemented for multi-objective optimization to handle materialized view selection problem for conventional data warehousing in Chapter 3. In basic MODE-BE, as implemented earlier in Chapter 3 for materialized view selection in conventional data warehousing, it has been observed that when the initial population, NP, is very large, there arises a memory issue because - before *mutation, cross-over* and *selection* are performed for all the NP solution vectors, the number of offspring solution vectors some times becomes too large to be handled. In basic MODE-BE, to control the population size in intermediate generations, the most elite solutions that maintain diversity in the population are retained discarding the other solution vector after the mutation, cross-over and selection are done for all NP solution vectors. In this improved version of MODE-BE for materialized view selection in Big data management framework, a threshold value is used to determine the maximum size of the population in intermediate generations such that whenever the offspring population size of a generation after mutation, cross-over and selection becomes more than this threshold value, the most elite solutions in terms of their Pareto rank and diversity measure in the population are retained for next generation discarding the rest of the offspring population generated. Selecting appropriate set of views for materializing out of a large number of solutions in the *first Pareto front* which may be positioned very closely in objective function space, is another issue in case of very large value of NP. Therefore in the proposed version of MODE-BE, a filtering criterion on maximum dissimilarity values among solutions is proposed based on the *Three Sigma Rule* [81]. The NSGA-II proposed by Deb et al. in [32] also has been implemented for comparative performance analysis between MODE-BE and NSGA-II in selecting views.

4.4.1 Multi-objective DE with binary encoded solutions for Big data view selection

For selecting materialized views for conventional data warehousing, the mutant vectors in multi-objective DE with binary encoded data generated by *forma basis* as discussed in [66,67] has been used in Chapter 3. In this work forma basis based multi-objective DE for binary encoded data (MODE-BE) has been used to design Algorithm 6 for selecting views to materialize in Big data framework based data warehousing.

In *DE/rand/1/bin* version of DE, the mutant vector for next generation

Algorithm 6: View selection for materializing by Multi-objective Differential Evolution using Binary Encoded Data in Big data based data warehouse.

Require: NP, g_{max}, F, CR, D, $C^Q_\varnothing$, $M_{v_{i=1,\cdots,m}}$, $U_{v_{i=1,\cdots,m}}$, $A_{v_{i=1,\cdots,m}}$, Γ , MinSpace

Ensure: A set of non-dominated solutions.

Generate NP random vectors $x_1, x_2, \cdots, x_{NP}$ of dimension D that satisfy the MinSpace constraint
$N \leftarrow NP$
$g \leftarrow 1$
repeat
  for $i = 1$ to N **do**
    select x_i and $x_{r_1}, x_{r_2}, x_{r_3}$, such that $x_i \neq x_{r_1} \neq x_{r_2} \neq x_{r_3}$
    for $j = 1$ to D **do**
      $v_{j,i} \leftarrow D_{\Psi_j}(x_{r_1}, round(F.D_{\Psi_j}(x_{r_2}, x_{r_3})))$
where $round(F.D_{\Psi_j}(x_{r_2,g}, x_{r_3,g})) =$

$$\begin{cases} 1, & \text{if random}[0,1] < F \wedge (D_{\Psi_j}(x_{r_2,g}, x_{r_3,g}) = 1) \\ 0, & \text{otherwise} \end{cases}$$

$$D_{\Psi_j}(x_{r_2}, x_{r_3}) = \begin{cases} 0, & \text{if } x_{j,r_2} = x_{j,r_3} \\ 1, & \text{otherwise} \end{cases}$$

      $j \leftarrow j + 1$
    end for
    $rand_i = random[1, D]$
    for $j = 1$ to D **do**

$$u_{j,i} \leftarrow \begin{cases} v_{j,i}, & \text{if}(random[0,1] \leq CR) \text{ or } j = rand_i \\ x_{j,i}, & \text{otherwise} \end{cases}$$

      $j \leftarrow j + 1$
    end for
    if $space(u_i) \geq MinSpace$ **then**
      Evaluate query processing cost $C^Q_{x_i}$ and $C^Q_{u_i}$, materialized view maintenance cost $U(x_i)$, $U(u_i)$ and number of non-zero elements in u_i, x_i
      if $u_i \prec x_i$ **then**
        $x_i \leftarrow u_i$
      else
        if $x_i \not\prec u_i$ **then**
          $NP \leftarrow NP + 1$
          $x_{NP} \leftarrow u_i$
        end if
      end if
    end if
    if $NP \geq \Gamma N$ **then**
      $i \leftarrow N + 1$
    else
      $i \leftarrow i + 1$
    end if
  end for
  if $NP > N$ **then**

Algorithm 7: View selection for materializing by Multi-objective Differential Evolution using Binary Encoded Data in Big data based data warehouse- (continued from previous page).

35: Compute and assign Pareto rank to each of the population vectors $x_{i=1,\cdots NP}$
36: Compute maximum distance of each solution vector to other solution vectors of the population, Max_i
37: Sort vectors $x_1, x_2, \cdots, x_{NP}$ in ascending order of Pareto-rank and descending order of Max_i
38: $NP \leftarrow N$
39: **end if**
40: $g \leftarrow g + 1$
41: **until** $(g < g_{max})$
42: Remove all dominated solutions from the population list $x_1, x_2, \cdots, x_{NP}$
43: **Return** The final set of solution population

$g+1$ for each target vector $x_{i,g}$, $i = 1, 2, \cdots, NP$, is generated as equation 4.5.

$$v_{i,g+1} = x_{r_1,g} + F.(x_{r_2,g} - x_{r_3,g}) \tag{4.5}$$

where $r_1, r_2, r_3 \in \{1, 2, \cdots, NP\}$, $r_1 \neq r_2 \neq r_3$, F is a real constant factor $\in [0, 1]$ and $F > 0$. By using forma basis [66, 67], Gong et al. in [31] expressed mutant vector defined by Equation 4.5 for binary encoded solution vector as Equation 4.6.

$$v_{j,i,g+1} = D_{\Psi_j}(x_{r_1,g}, F.D_{\Psi_j}(x_{r_2,g}, x_{r_3,g})) \tag{4.6}$$

Where, $x_{r_2,g}$ and $x_{r_3,g}$ are considered as two strings of bits of length D and each jth dimension difference between $x_{r_2,g}$ and $x_{r_3,g}$, $D_{\Psi_j}(x_{r_2,g}, x_{r_3,g})$ is represented by using *formae basis* [31] Ψ_j as Equation 4.7.

$$D_{\Psi_j}(\mathbf{x}, \mathbf{y}) = \begin{cases} 0, & \text{if } x_j = y_j \\ 1, & \text{otherwise} \end{cases} \tag{4.7}$$

To interpret the scaled difference $F.D_{\Psi_j}(x_{r_2,g}, x_{r_3,g})$ of jth dimension rounded to 1 or 0, Equation 4.8 is used.

$$F.D_{\Psi_j}(x_{r_2,g}, x_{r_3,g}) = \begin{cases} 1, & \text{if random}[0,1] < F \wedge (D_{\Psi_j}(x_{r_2,g}, x_{r_3,g}) = 1) \\ 0, & \text{otherwise} \end{cases} \tag{4.8}$$

In Algorithm 6, a set of initial solutions $x_1, x_2, \cdots, x_{NP}$ are generated for a given set of frequent queries Q that satisfy the space constraint. In each generation of a evolutionary process g, against each solution vector $x_i, i = 1, 2, \cdots, NP$, a mutant vector $v_{i,g+1}$ is generated as expressed by Equation 4.6 (and Equation 4.8). Then trial vector $u_{i,g+1}$ is formed by crossover. To adapt the problem of view selection to materialize in Big data, the query processing cost, the materialized view maintenance cost and the number of views in solution sets of the considered frequent queries Q, for each solution vector $x_{i,g}$ and trial vector $u_{i,g+1}$ are computed

by using equations (4.1),(4.2) and by counting number of selected views in $x_{i,g}$ and $u_{i,g+1}$. If $u_{i,g+1} \prec x_{i,g}$, then $x_{i,g+1}$ is set as $u_{i,g+1}$, else if $x_{i,g} \prec u_{i,g+1}$, then $u_{i,g+1}$ is discarded. Otherwise, in case $u_{i,g+1} \not\prec x_{i,g}$ and $x_{i,g} \not\prec u_{i,g+1}$, $u_{i,g+1}$ is appended to the population for next generation $g+1$. Thus the population may go on increasing. To control the population growth in each generation of DE, when the population size touches a limit, i.e when NP becomes ΓN, N being the initial population size (i.e initial NP), and Γ being a positive real constant, a technique is used to filter out NP elite solution population that maintains diversity in the solution population as discussed in Section 4.4.1.1. This evolutionary process is continued till it reaches a maximum number of generations specified, say g_{max}. The dominated solutions in the final population are then removed from the archive to return the non-dominated solutions of the problem.

4.4.1.1 Promoting elitism and diversity in solution population

For elitism and diversity in solution population, though the Pareto ranking has been used as suggested by Deb et al. in [32], the diversity of solutions are maintained in solution space unlike it is done by using Crowding distance in objective function space for NSGA-II. The diversity in solution space is preferred here for wide coverage of representations of views in the solution set which has already been discussed in Chapter 3. The Pareto ranking and diversity preservation used in this version of MODE-BE are as stated below.

Pareto ranking: To control the population size by keeping the diversity in solution population in the intermediate generations of DE in this approach, the diversity of solutions in solution space is promoted with necessary elitism. When the population size NP in a generation becomes ΓN, where N is the initial value of NP in that generation, the solutions of intermediate generations are ranked according to their Pareto dominance levels as discussed in [32]. The solution population is then sorted in ascending order of their Pareto ranks so that the most elite solutions from the list may be kept in the population for next generation. To ensure finding out the most elite NP solutions, Deb et al. [32] suggests $\Gamma = 2$.

Diversity in solution space: For each $i-th$ solution of the population, the maximum distance to other solution vectors in the population Max_i is measured. Since the solution vectors are represented as a string of bits and a particular bit as 1 or 0 does not mean any preference over each other, the *Simple Matching Co-efficient (SMC) distance* measure [72] is used for measuring distance between two solutions. The solution population sorted in ascending order of their Pareto ranks are then sorted again on descending order of their maximum distances to other solutions in the population. From the doubly sorted solution population, the top NP solutions are retained as next generation population. Here, to promote diversity of solutions, density in solution space is used instead of using crowding density measured by *crowding distance* of solutions in objective function value space.

4.4.1.2 Filtering Representative Solutions from Non-dominated Solutions Obtained

From large number of non-dominated solutions yielded by multi-objective optimization, finding significant representation set is useful for decision makers [82]. In selecting views for materialization in data warehouse, degree of diversity in solutions is important as solutions having larger representations from candidate views are preferable. Therefore, for filtering significant solutions from the obtained solutions, farthest distance approach may be incorporated. In our implementation, first the SMC based maximum distance from every solution vector S_i of n number of non dominated solutions to other solution vectors in the population, $Max_i, i = 1, 2, \cdots, n$, is computed. Then the mean μMax and standard deviation σMax of $Max_i, i = 1, 2, \cdots, n$, are computed. In the empirical sciences and in statistics the "three sigma rule of thumb" expresses a conventional heuristic that in a normal distribution, 68.27% of values lie within mean (μ) and one standard deviation(σ) i.e within $\mu \pm \sigma$, 95.45% values lie within mean and two standard deviation i.e within $\mu \pm 2\sigma$ and 99.73% values lie within mean and three standard deviation i.e within $\mu \pm 3\sigma$. Based on this "three sigma rule of thumb" popularly referred as "68-95-99.7 rule", for narrowing down the options of considering non dominated solutions, a solution S_i is discarded if $Max_i < (\mu Max + C.\sigma Max)$, where C is a real constant that may be specified as positive, negative or zero, depending on number of solutions we want to filter out from the population, preserving diversity in solution space.

4.4.2 Implementing NSGA-II for view selection

In [26], two implementations of multi-objective GA for materialized view selection have been presented. These two approaches used non-elitist multi-objective evolutionary algorithms for selecting views under storage space constraint. Deb et al. in [32] presented that elitism can speed-up the performances of the GA significantly and also can help retaining good solutions generated during intermediate generations. In multi-objective GAs for ensuring diversity in solution population, the concept of sharing parameter σ_{share} in objective space is used. But a parameter less diversity preservation mechanism is better. To address this issue NSGA-II was suggested [32].

To adapt NSGA-II for selecting views to materialize in Big data warehousing with distributed file system framework, I used the cost model and problem definition as discussed in Section 4.2 and 4.3.

Here, first a random solution population $\Psi_1, \Psi_2, \cdots, \Psi_{NP}$ are created using the same solution representation as used in MODE-BE. For generating ith random solution, initially all decision variables of Ψ_i are set as 0. A random integer in the range [0,D] is generated for deciding how many dimensions of Ψ_i is to be set as 1. This random number is the cardinality of the set of views selected V' in Ψ_i, expressed as $|V'| = p$ in Definition 22. Randomly these p number of

decision variables are set as 1 for the vector Ψ_i. As discussed in Section 4.2, in Big data framework based data warehousing (like Hive), smaller number of larger sized views are to be selected for materializing. Therefore, the solution vector Ψ_i is added to the list of initial population only if the total size of the set of p number of views of the set V' satisfies the minimum space criteria specified and Ψ_i is not already present in the solution population.

In subsequent generations, the usual binary tournament selection, recombination and mutation operators are applied to the NP solutions to create offspring. In each generation, against a selected solution from the NP solutions, one offspring is generated. For finding domination or non-domination between two solutions, say Ψ_i and Ψ_j, where V_i' is the set of non-zero dimensions of Ψ_i and V_j' is the set of non-zero dimensions of Ψ_j, the three objective functions (1) the query processing costs $C^Q_{V_i'}$, $C^Q_{V_j'}$, (2) maintenance costs $U(V_i')$, and $U(V_j')$ and (3) $|V_i'|$, $|V_j'|$ are evaluated. If the generated offspring dominates the selected solution vector, then the new offspring replaces the selected vector. Otherwise , if the newly generated offspring does not dominate the solution vector and if the selected solution also does not dominate the offspring, the offspring vector is added to the population. Thus a new offspring population of size N is created. Whenever N becomes $2NP$, the solution population in the list are ranked in their non-domination levels. The ranked solution population are sorted in ascending order of their non-domination ranks. The crowding distance among the solutions are then computed in objective function space and sorted in descending order of their crowding distances. The solutions are sorted in ascending order of ranks for providing higher priority for keeping the solutions of lower domination ranks in next generation, so that the most elite solutions are retained in subsequent generations. The solution population are sorted in descending order of objective function based crowding distances to preserve diversity among solution population in each generation.

4.4.2.1 Run time complexity of NSGA-II

The basic operations of NSGA-II based application's worst case complexities as presented in [32] are - (1) for non-dominated sorting is $O(M(2N)^2)$, (2) for crowding distance assignments is $O(M(2N)log(2N))$ and (3) for sorting on crowding distances is $O(2Nlog(2N))$. Here, M is the number of objectives and N is the number of solution population. Thus the over all complexity is dominated by $O(M(2N)^2)$. As our problem is defined with 3 objectives therefore it becomes $O(3(2N)^2)$. Thus the overall complexity is $O(N^2)$.

4.5 Experimentation and Observations

For experimental analysis, Multi-Objective DE for Binary Encoded solutions, MODE-BE, and NSGA-II as a recommending system, taking input from log-files

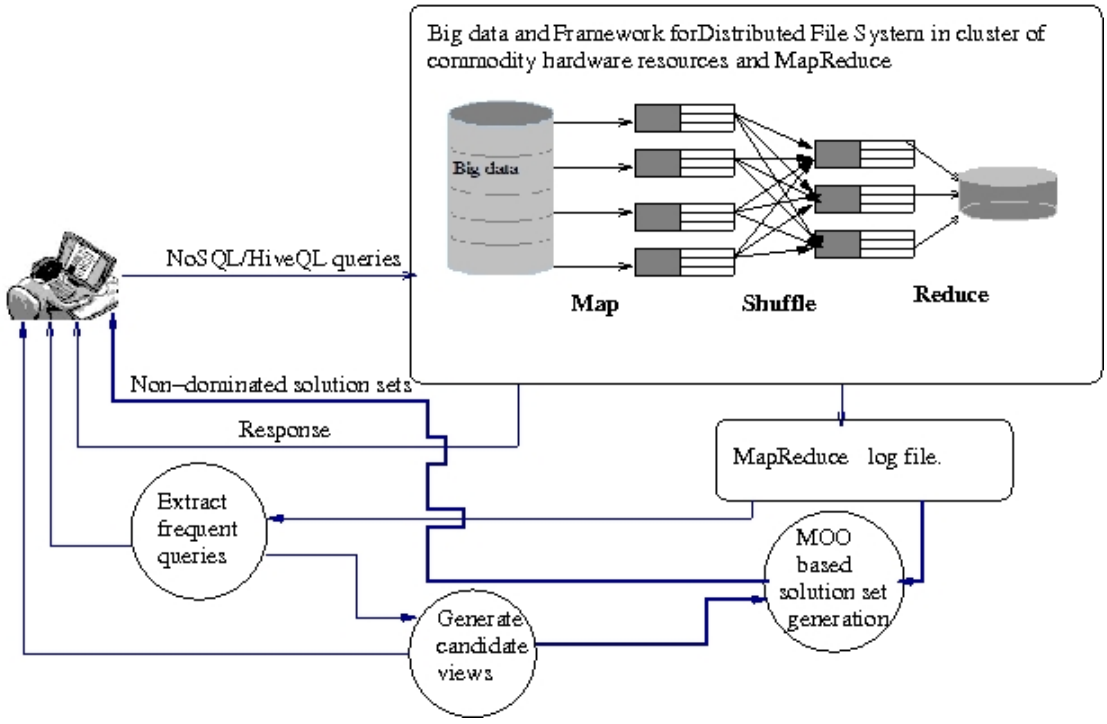

Figure 4-1: Test-bed for selecting non-dominated solution sets of materialized views

generated on processing HiveQL instructions, has been implemented. The recommended solution sets generated by both the implementations are analyzed. A set of HiveQL queries has been synthesized for triggering on data warehouse in a single node implementation of experimental HDFS. This set of queries considered as the set of frequent queries and are broken-up into some sub-queries or views which are considered as candidate views for our experimentation.

4.5.1 Experimental setup

In this experimental setup, Hortonworks Data Platform (HDP) version 2.0.6 has been used with Hortonworks Sandbox version 2.0 VMware for 64 bit CentOS operating system workstation 6.5-7.x virtual machine [83]. This is a single node implementation for experimenting HDFS. Hadoop version 2.2.0 and Hive version 0.12.0 of Apache [80] are used here, which let us manage data, perform ad-hoc queries and perform analysis of data warehouse in Hadoop cluster. For executing HiveQL queries, an HDP interactive interface to Hive^{TM} named Beeswax provided by HDP has been used. Using Beeswax we can type in HiveQL queries and have Hive evaluate them for us using a series of MapReduce jobs.

A block diagram of my test-bed is presented in Fig. 4-1. Though generally "Big data" means database of Tera byte/Peta byte size, HDP is designed for single node implementation for experimenting HDFS with smaller sized databases. In this experimentation Lahman Baseball Database of American Major League Baseball statistics from 1871 through 2011 [84] have been used as suggested by

HDP 2.0.6 for experimenting with Hadoop version 2.2.0.

For generating different cost function values for experimentation, few HiveQL queries on Lahman baseball database have been synthesized as listed below.

List of HiveQL test queries:

Q1. select a.yearid, a.playerid, b.average, c.totgs from batting_data a join (select yearid, playerid, avg(r) as average from batting_data group by yearid , playerid) b on a.playerid=b.playerid and a.yearid=b.yearid join (select yearid, playerid, sum(gs) as totgs from appearances_data group by yearid, playerid) c on a.playerid=c.playerid and a.yearid=c.yearid ;

Q2. select a.yearid, a.playerid, b.average, c.tot_batting from batting_data a join (select yearid, playerid, avg(r) as average from batting_data group by yearid, playerid) b on a.playerid=b.playerid and a.yearid=b.yearid left outer join (select yearid, playerid, sum(g_batting) as tot_batting from appearances_data group by yearid, playerid) c on a.playerid=c.playerid and a.yearid=c.yearid ;

Q3. select a.yearid, a.playerid, b.wild_pitches, c.tot_glf from fielding_data a join (select yearid, playerid, sum(wp) as wild_pitches from fielding_data group by yearid, playerid) b on a.yearid=b.yearid and a.playerid=b.playerid left outer join (select yearid, playerid, sum(glf) as tot_glf from fielding_of group by yearid, playerid) c on a.yearid=c.yearid and a.playerid=c.playerid ;

Q4. select a.yearid, a.teamid, a.playerid, b.average, c.wild_pitches from appearances_data a left outer join (select yearid, playerid, teamid, avg(r) as average from batting_data group by yearid, teamid, playerid) b on a.playerid=b.playerid and a.yearid=b.yearid and a.teamid=b.teamid left outer join (select yearid, playerid, sum(wp) as wild_pitches from fielding_data group by yearid, playerid) c on a.yearid=c.yearid and a.playerid=c.playerid ;

Q5. select a.lahmanid, a.playerid, a.hofid, b.yearid, b.totgs, c.totvotes from master_data a left outer join (select yearid, playerid, sum(gs) as totgs from appearances_data group by yearid, playerid) b on a.playerid=b.playerid left outer join (select hofid, yearid, sum(votes) as totvotes from hall_of_fame group by hofid, yearid) c on a.hofid=c.hofid and b.yearid=c.yearid;

Q6. select a.playerid, b.playedin, a.teamid, c.winner from appearances_data a join (select playerid, count(*) as playedin from appearances_data group by playerid) b on a.playerid=b.playerid left outer join (select teamidwinner, count(*) as winner from series_post group by teamidwinner) c on a.teamid=c.teamidwinner ;

Q7. select a.playerid, b.total_played, c.tot_fielding, d.tot_pitching from appearances_data a join (select playerid, sum(g_all) as total_played from appearances_data group by playerid) b on a.playerid=b.playerid left outer join (select playerid, sum(g) as tot_fielding from fielding_data group by playerid) c on

a.playerid=c.playerid left outer join (select playerid, sum(g) as tot_pitching from pitching_post group by playerid) d on a.playerid=d.playerid ;

Q8. select a.yearid, a.teamid, a.lgid, a.playerid, b.playedin, c.winner from appearances_data a join (select yearid, teamid, lgid, playerid, count(*) as playedin from appearances_data group by yearid, teamid, lgid, playerid) b on a.yearid=b.yearid and a.teamid=b.teamid and a.lgid=b.lgid and a.playerid=b.playerid left outer join (select teamidwinner , count(*) as winner from series_post group by teamidwinner) c on a.teamid=c.teamidwinner ;

Q9. select a.yearid, a.teamid, a.playerid, b.average, c.max_runs from appearances_data a left outer join (select yearid, playerid, teamid, avg(r) as average from batting_data group by yearid, teamid, playerid) b on a.playerid=b.playerid and a.yearid=b.yearid and a.teamid=b.teamid left outer join (select playerid , max(r) as max_runs from batting_data group by playerid) c on a.playerid=c.playerid;

Q10. select a.playerid, a.namefirst, a.namelast, b.max_runs, c.min_runs from master_data a join (select playerid , max(r) as max_runs from batting_data group by playerid) b on a.playerid=b.playerid join (select playerid, min(r) as min_runs from batting_data group by playerid) c on a.playerid=c.playerid ;

Q11. select a.playerid, a.hofid, b.runs_allowed, c.tot_votes from master_data a left outer join (select playerid, sum(r) as runs_allowed from pitching_post group by playerid) b on a.playerid=b.playerid left outer join (select hofid, sum(votes) as tot_votes from hall_of_fame group by hofid) c on a.hofid=c.hofid ;

Q12. select a.playerid, b.tot_leftfielding, c.namefirst, c.namelast from fielding_post a left outer join (select playerid, sum(g_lf) as tot_leftfielding from appearances_data group by playerid) b on a.playerid=b.playerid join (select playerid, namefirst, namelast from master_data) c on a.playerid=c.playerid ;

Q13. select a.playerid, b.tot_innouts, c.namefirst, c.namelast from fielding_post a join (select playerid, sum(innouts) as tot_innouts from fielding_post group by playerid) b on a.playerid=b.playerid join master_data c on a.playerid=c.playerid ;

Q14. select a.hofid, a.yearid, a.category, b.namefirst, b.namelast, c.batting from hall_of_fame a join master_data b on a.hofid=b.hofid join (select playerid, sum(g_batting) as batting from appearances_data group by playerid) c on b.playerid=c.playerid;

Q15. select a.playerid, a.bats, b.tot_runs, b.tot_hits from master_data a left outer join (select playerid, sum(r) as tot_runs, sum(h) as tot_hits from batting_post group by playerid) b on a.playerid=b.playerid ;

Q16. select a.playerid, b.tot_runs, c.tot_hits, c.bats, c.namefirst, c.namelast from batting_post a join (select playerid, sum(r) as tot_runs, sum(h) as tot_hits from batting_post group by playerid) b on a.playerid=b.playerid join master_data c on a.playerid=c.playerid ;

Q17. select a.playerid, b.tot_games, b.errors, c.throws, c.namefirst, c.namelast from fielding_post a join (select playerid, sum(g) as tot_games, sum(e) as errors from fielding_post group by playerid) b on a.playerid=b.playerid join master_data c on a.playerid=c.playerid ;

Q18. select a.playerid, a.glf, a.gcf, a.grf, b.tot_games, c.tot_outfielding from fielding_of a join (select playerid, sum(g) as tot_games from fielding_post group by playerid) b on a.playerid=b.playerid left outer join (select playerid, sum(g_of) as tot_outfielding from appearances_data group by playerid) c on a.playerid=c.playerid ;

Q19. select a.playerid, a.stint, a.yearid, a.teamid, a.lgid, b.r, b.h from batting_data a join (select * from batting_post where r>0) b on a.playerid=b.playerid and a.yearid=b.yearid and a.teamid=b.teamid and a.lgid=b.lgid ;

Q20. select a.playerid, a.yearid, a.teamid, a.lgid, b.g, b.r, b.h, b.rbi from appearances_data a join (select * from batting_post where r>0) b on a.playerid=b.playerid and a.yearid=b.yearid and a.teamid=b.teamid and a.lgid=b.lgid ;

The constituent views and aggregation functions that are to be considered as candidate views for materializing are extracted from these queries by a semantic analysis process. The queries and their constituent views are presented in Table 4.1. The queries and constituent views are triggered to HDP to get query processing costs, view processing costs and maintenance costs in terms of MapReduce CPU time along with the space requirements for the views. The HDP and Beeswax interface generates responses as well as log-files against HiveQL commands. These log-files contain all MapReduce split details and associated CPU costs along with the MapReduce jobs creation details. Different costs against these queries and views are extracted from log-files and stored in a database. The extracted costs of our queries in one instance of execution are presented in Table 4.2 and 4.3. The materialized view selection process takes input from this database for recommending optimum sets of views for materializing. All the selected frequent queries and candidate views are indexed and labeled to represent the solution vectors such that if the first dimension of the solution vector is 1, the first view is selected for materializing and if the second dimension of the solution vector is 0, the view labeled as second view is not to be selected for materializing and so on. This representation is used in many materialized view selection techniques [8,9,14,25]. The extracted costs of our queries in one instance of execution as we present in Table 4.2 and 4.3 are used as input to our multi-objective EA based view selection recommendation system.

4.5.2 Parameters used

In Differential Evolution (DE) algorithm based applications the main control parameters are the mutation scaling factor F, the solution population size NP and the cross-over parameter CR. In [28] it has been suggested that the value of NP

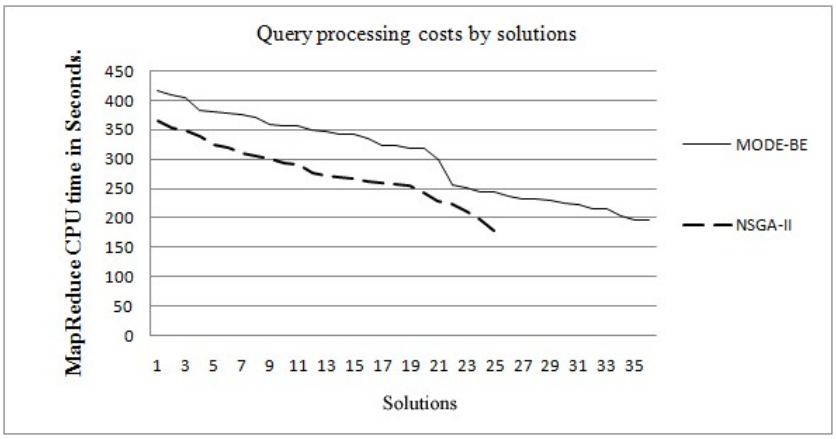

Figure 4-2: Processing MapReduce cost (in Seconds) by NSGA-II and MODE-BE generated non-dominated solutions.

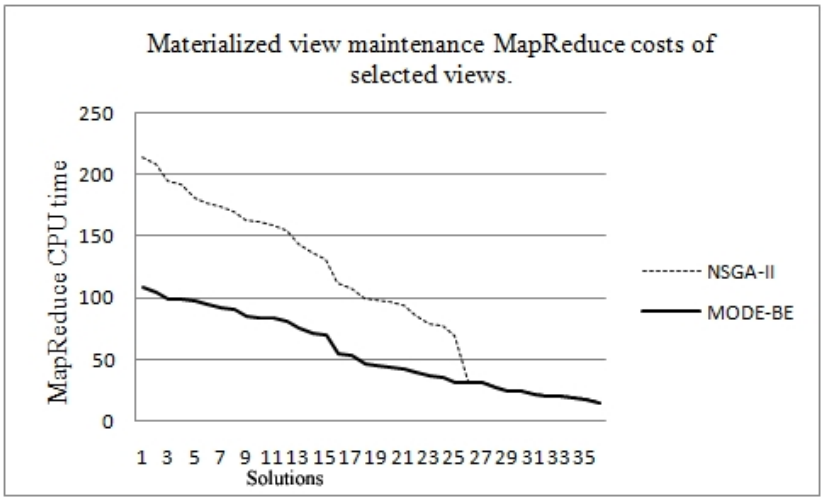

Figure 4-3: Materialized view maintenance MapReduce cost (in Seconds) by NSGA-II and MODE-BE generated non-dominated solutions.

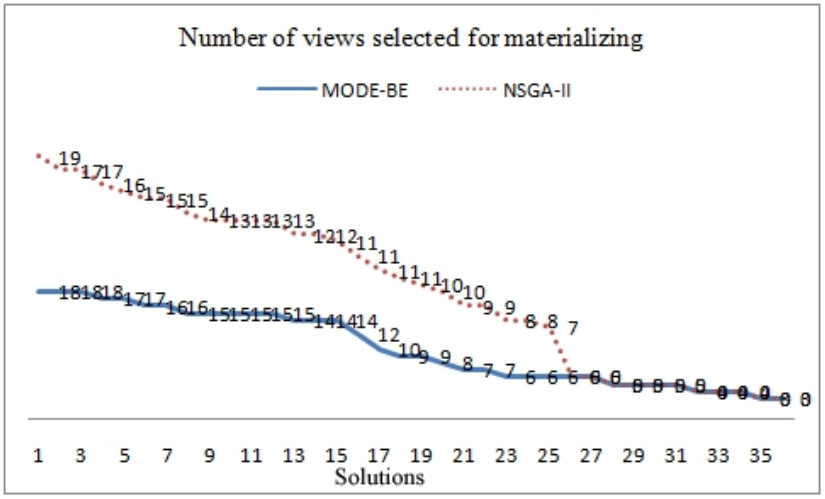

Figure 4-4: Number of views in solution sets for materializing.

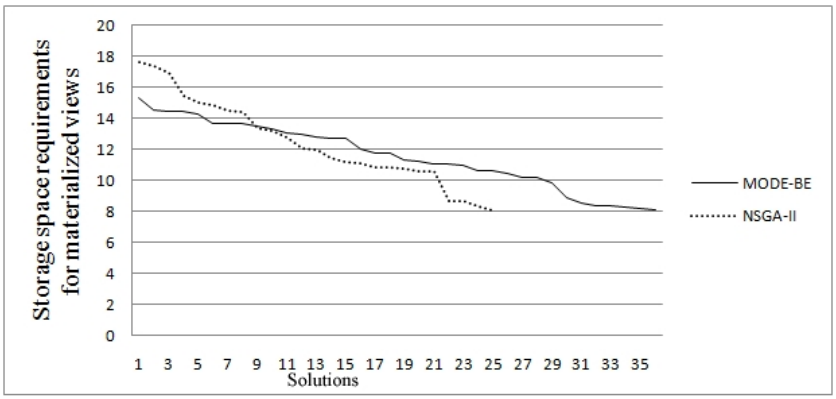

Figure 4-5: Space requirements by solution sets of views for materializing.

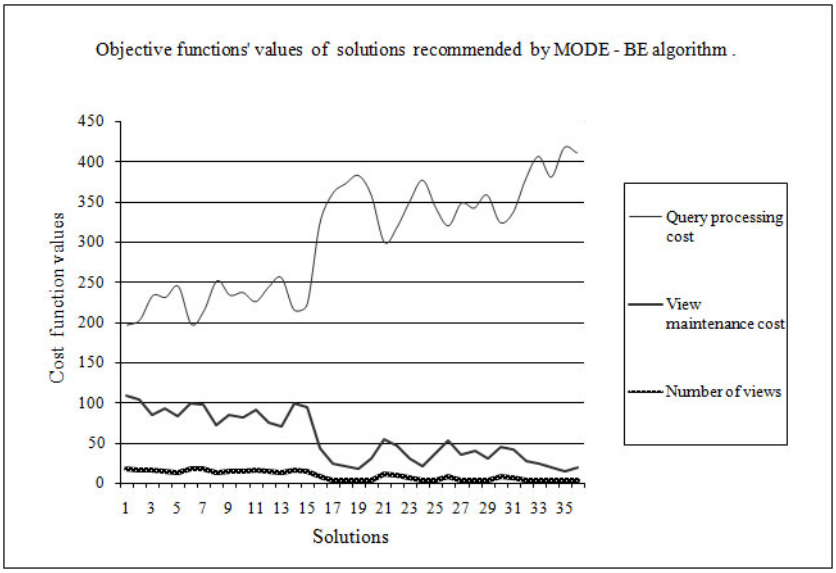

Figure 4-6: Objective functions' values by MODE-BE generated non-dominating solutions.

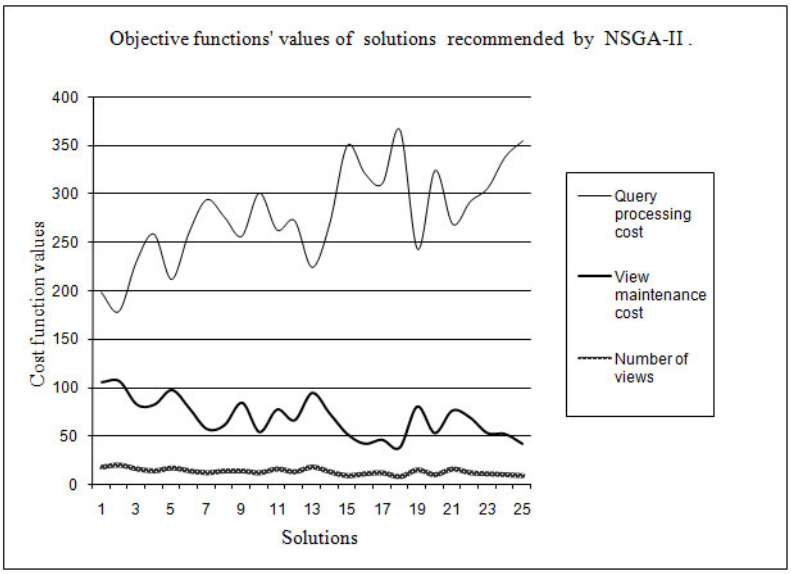

Figure 4-7: Objective functions' values by NSGA-II generated non-dominating solutions.

Table 4.1: Considered frequent HiveQL queries and constituent views

HiveQL queries	Constituent views
Q_1	v_1,v_2
Q_2	v_1,v_3
Q_3	v_4,v_5
Q_4	v_4,v_6
Q_5	v_2,v_7
Q_6	v_8,v_9
Q_7	v_{10},v_{11},v_{12}
Q_8	v_9,v_{13}
Q_9	v_6,v_{14}
Q_{10}	v_{14},v_{15}
Q_{11}	v_{16},v_{17}
Q_{12}	v_{18},v_{19}
Q_{13}	v_{20}
Q_{14}	v_{21}
Q_{15}	v_{22}
Q_{16}	v_{22}
Q_{17}	v_{23}
Q_{18}	v_{23},v_{24}
Q_{19}	v_{25}
Q_{20}	v_{25}

should be around 5 to 10 times the dimensionality of the problem. Therefore, for 25 candidate views, we may set values of NP between 125 to 250. In DE, a good choice of F is 0.5. The value of CR indicates number of inheritance by the mutant vector. According to [27, 73], for population size NP between 3 to 8 times of the dimensionality of the problem, the mutation scaling factor F=0.6 and cross-over ratio CR between 0.3 to 0.9 are good choices. In our multi-objective DE for binary represented decision variables (MODE-BE) we used the following parameters -

- the population size, NP=125,
- number of generations=50,
- selection scheme=*DE/rand/1/bin*,
- F=0.5,
- binary cross-over probability CR=0.3.

To compare the performance of MODE-BE and NSGA-II in materialized view selection problem for HDFS based data warehousing, we used following parameters with NSGA-II based system.

- the population size, NP=125,

Table 4.2: Query responding MapReduce costs of selected queries

HiveQL queries	HDFS MapReduce cost (in Seconds)
Q_1	45.05
Q_2	37.62
Q_3	37.35
Q_4	42.59
Q_5	25.51
Q_6	24
Q_7	25.48
Q_8	38.87
Q_9	29.77
Q_{10}	18.29
Q_{11}	22.57
Q_{12}	14.15
Q_{13}	12.68
Q_{14}	29.27
Q_{15}	13.83
Q_{16}	20.56
Q_{17}	22.04
Q_{18}	21.82
Q_{19}	2.39
Q_{20}	2.31

Table 4.3: Processing and maintenance MapReduce costs and space requirements of candidate views

Candidate view	Processing MapReduce cost (in Seconds)	Maintenance MapReduce cost (in Seconds)	Space (in MB)
v_1	11.52	4.023	2.2
v_2	16.34	4.234	2.236
v_3	19.43	4.034	2.151
v_4	14.16	2.652	2.2
v_5	8.37	6.051	0.267
v_6	13.85	13.042	2.9
v_7	6.84	11.521	0.0956
v_8	11.74	3.231	0.296
v_9	7.75	6.455	0.001
v_{10}	15.71	5.823	0.316
v_{11}	16.98	10.034	0.313
v_{12}	6.02	5.034	0.0252
v_{13}	20.12	4.611	3.252
v_{14}	11.05	6.810	0.3
v_{15}	13.76	5.430	0.294
v_{16}	6.61	4.517	0.026
v_{17}	10.73	2.220	0.021
v_{18}	10.11	4.315	0.297
v_{19}	1.84	5.412	0.519
v_{20}	7.28	3.021	0.061
v_{21}	16.38	5.011	0.314
v_{22}	8.07	9.025	0.075
v_{23}	6.99	7.25	0.07
v_{24}	11.58	9.312	0.299
v_{25}	2.3	5.812	0.487

- number of generations=50,
- Size of mating pool=125,
- tournament size=2,
- individual cross-over probability=1,
- individual mutation probability=1.

Two other problem specific parameters - maximum number of views that may be selected and minimum size of storage space to be used are also to be specified as well. These two parameters are mainly dependent on size of memory block size and split size of the HDFS. Generally HDFS block-size and split-size are of 64MB or 128MB. Therefore, as small files may use unnecessary MapReduce split and overhead, Hadoop works better with smaller number of larger files [17].

4.5.3 Results and observations

In my experimentation with the above mentioned experimental setup, parameters and data, it has been observed that,

- the NSGA-II based system converges more quickly than MODE-BE based recommendation system. But, as for preserving diversity among solutions in MODE-BE, distances among solutions in their solution vector space are used instead of crowding-distance in objective function space, the standard deviation between solutions generated by MODE-BE is 5.203402 whereas that of NSGA-II is 3.120897. The diversity in solution vector space is preferred because diversity preservation on objective function values may lead to loss of some significantly distinct solutions on the basis of constituent selected views in them. This may obviously happen because a scalar valued function with different vector parameters may result same scalar value.
- In our experimentation it can be observed that MODE-BE generates 37.04% more number of solutions than NSGA-II based system. More number of solutions with comparable quality of solutions may be useful for selecting most appropriate solutions depending on user application requirements. For filtering significant solutions from the obtained solutions, distances in solution vector space for each solution to all other solutions yielded may be computed and then based on the mean (μ) distances and their standard deviation (σ), filtration criteria may be applied [25].
- From the query processing costs in terms of MapReduce time, as plotted in Figure 4-2, it can be observed that solutions generated by MODE-BE are more costly in case of query processing and responding than of NSGA-II generated solutions. But, the MapReduce time for maintaining the materialized views are less in case of MODE-BE.

- From Figure 4-4 and 4-5 it can be observed that MODE-BE results are slightly better based on the Hadoop framework's basic criterion that lesser number of bigger views or tables are to be considered for materializing. For our experimental data presented here, we found that the minimum size of storage space requirement is slightly more in case of MODE-BE.
- Again, by applying Mann-Whitney U test on both MODE-BE and NSGA-II generated solutions at 5% level of significance, i.e at $\alpha = 0.05$, we cannot reject the null hypothesis that the solution vectors generated by both the systems are from the same population.

4.6 Discussion

I have presented here my study on view selection techniques for materializing in distributed commodity hardware file system data warehousing. The main contribution here is establishing the view selection for materializing problem in distributed commodity hardware file system as a multi-objective optimization problem and study of performances by multi-objective Differential Evolution algorithm and NSGA-II for solving this problem. As it is an NP-hard problem, I used multi-objective evolutionary algorithm based approach for designing a recommendation system for selecting views for materializing.

A prototype of view selection framework has been designed that uses log-files generated by single node implementation of HDFS based data warehousing framework and Hive 0.12.0 queries. As the approach is a generic one, it may be implemented easily for any kind of similar framework that uses distributed cluster of commodity hardware.

Though, in this experimental framework, the cost functions, number of views in different solution sets and space costs of each individual solutions are computed by the recommending system, the semantic analysis process for extracting and composing candidate views is yet to be developed. At present the candidate views are fed to the HDP by an offline process for computing different associated costs.

The present popular version of Hive^{TM} does not support materialized views. The work presented here leads towards efficient Big data query processing by materializing selected views recommended by the system in cluster of distributed commodity hardware file system. There may be alternative technologies for implementing the materialized view selection and/or selection of response of queries for in-memory materialization like Discardable In-Memory Materialized Query (DIMMQ), Spark's Resilient Distributed Data-set (RDD)[19] etc.. We believe that community like Hadoop may find many more such technologies and will suggest bench-mark framework for unbiased evaluation of different techniques for efficient Big data query processing.

In this chapter the solution quality of multi-objective DE and NSGA-II in

view selection for materializing in Big data DFS framework have been analyzed based on the cost function values by the yielded solutions. The performances of non-deterministic multi-objective optimization techniques may be studied in different ways. In next chapter another widely used stochastic method for hard optimization problems called Simulated Annealing (SA) has been customized for solving this problem and comparative performances of MODE-BE, NSGA-II and multi-objective SA have been reported while applying in this problem based on measures on how the obtained solutions converge towards estimated true Pareto front, number of obtained solutions that are not dominated by solutions generated by other algorithms and distribution of solutions on the found Pareto front.

Chapter 5

Multi-Objective Simulated Annealing Algorithm in Big data View Selection for Materializing

5.1 Introduction

The pioneering non-deterministic optimization based approach used in selecting views for materialization in data warehouse was a Genetic Algorithmic (GA) approach by Zhang et al. [38]. Derakhshan et al. in [8] introduce an approach for materialized view selection using Simulated Annealing (SA) with Multiple View Processing Plan (MVPP) [7] of frequent data warehouse queries as input. This approach was later modified by applying Parallel Simulated Annealing (PSA) [9]. In [14], Multi-Objective Simulated Annealing (MOSA) [61,85] and Archived Multi-Objective Simulated Annealing (AMOSA) [86] are applied by us in materialized view selection problem using MVPP based representation of the problem. This chapter presents how multi-objective SA based techniques may be applied in selecting sub-query results or views in MapReduce based query processing framework. A comparative performance analysis of this technique with respect to MODE-BE and NSGA-II based techniques in view selection problem in this paradigm also has been presented in this chapter.

5.1.1 Motivation

Derakhshan et al. in [8] showed that by using SA, the cost of a selected set of materialized views is up to 70% less than the GA based technique in [37] and heuristic algorithmic approach in [7] both of which use MVPP graph as input. Derakhshan et al. in [9] present that PSA in conjunction with MVPP graph outperforms heuristic method [7] and basic SA based technique [8] considering the cost of obtained set of views. SA and GA are both stochastic methods widely used for handling hard optimization problems. The key difference between SA and GA

is that SA creates a new solution by modifying only one solution with a local move in the solution space but GA creates solutions by combining two different solutions from the solution population. Lahtinen et al. in [87] compare several algorithms including SA and GA considering a tree cost minimization problem by normalizing the execution time given to different algorithms and presented that in same amount of execution time, SA consistently gave better solutions than GA. In [88] it has been reported that though GA gives slightly better solutions than SA, SA achieved its solutions much quicker.

5.1.2 SA for multi-objective optimization

Though SA has been applied in diverse type of single objective optimization problem, there have been very few attempts in extending it for multi-objective optimization problem [30]. In [8,9], to handle the materialized view selection problem as a single objective optimization problem for solving it by SA, query processing cost, materialized view maintenance cost and space cost are combined. In fundamental SA of statistical mechanics, if $\delta E(x', x)$ is the associated energy difference of newly generated solution state x' and current solution state x in an annealing process, x' is accepted as next state or move in the process, if a random value in the range [0,1] is less than $e^{-\delta E(x',x)/T}$, where T is the system temperature. In higher temperature (T), there will be higher probability that an inferior move will be selected. In the annealing process, the temperature is decreased from high to sufficiently low very gradually ensuring sufficient time at each temperature i.e to explore more regions in solution space for better solution in each temperature by large number of iterations. In single objective SA, $\delta E(x', x)$ is used as cost function difference between two points in the solution space. It has been proved that in SA, if annealed sufficiently slow, it converges to the global optimum [30]. To extend SA for multi-objective optimization, few Pareto-dominance based Multi-Objective SA (MOSA) techniques have been developed. In MOSA presented in [61,85], acceptance criterion between the current solution and newly generated solution is defined in terms of the difference between number of solutions dominated by them. A new version of MOSA referred as Archived Multi-Objective SA (AMOSA) presented by Bandyopadhyay et al. in [30] incorporates a novel concept of amount of dominance instead of number of solutions dominated, as acceptance criterion to determine the acceptance of a newly generated solution. As multi-objective optimization may yield a large number of Pareto optimum solutions, in AMOSA, to limit the size of the *Archive* of non-dominated solutions with reduced loss of diversity, clustering is used. After obtaining the clusters, the solution whose average distance to other members in the cluster of solutions is the minimum, is considered as the representative member of each cluster. Then from each of the clusters the representative solutions are added into the Archive for subsequent processing or iterations.

5.1.3 Contribution

In this work, the basic Archived Multi-Objective Simulated Annealing (AMOSA) algorithm designed by Bandyopadhyay et al. in [30] is customized for applying in materialized view selection problem in MapReduce based distributed file system framework. This work is an extension of our earlier implementation of MOSA and AMOSA for handling materialized view selection using MVPP graph in conventional relational database management system [14].

In this implementation of AMOSA, the diversity of solutions are maintained in solution space by computing maximum dissimilarity value of each solution to other solutions in the archive while controlling the size of solution population. The solutions are sorted in descending order of maximum dissimilarity values of each solution and the top few solutions from the sorted list are picked up for adding into the archive during the annealing process or iterations. In fundamental AMOSA [30] the diversity of solutions are maintained in objective function space by selecting representative solutions that are having minimum distances to other solutions in each cluster in objective function space. As in the proposed version the diversity of solutions are maintained in solution space, more diversity of solutions with respect to constituent views may be achieved. The complexity of using the proposed customized version of AMOSA for selecting views is less than the original AMOSA because the run time complexity of single-linkage clustering is more than that of computing the maximum distance computation of each solution to other solutions in the archive.

A *Hadoop version 2.2.0* based framework [83] is designed for generating log-files for synthesizing data regarding MapReduce costs of queries and constituent views by triggering random Hive [80] queries. The AMOSA in this materialized view selection problem is designed for binary encoded data and therefore the solutions are represented as binary strings. The performance of this version of AMOSA for materialized view selection, referred as AMOSA-MVS, is compared with NSGA-II [29] and multi-objective DE with binary encoded data, MODE-BE [25], in terms of *Convergence* [32], *Purity* [30] and *Spacing* measures [89] of respective obtained solutions in materialized view selection.

5.2 Dominance based Energy Functions in Multi-Objective Simulated Annealing Algorithm

In Simulated Annealing algorithm based optimization, a new solution x', generated by perturbing a current solution x, is accepted as a better proposal with probability

$$\mathbb{P} = min(1, exp\{-\delta E(x', x)/T\}) \tag{5.1}$$

where $\delta E(x', x) \equiv E(x') - E(x)$, and $E(x)$ and $E(x')$ are *energy functions* on x and x' respectively at system temperature parameter T to obtain optimum solution state x that optimizes $E(x)$ [90]. In single objective optimization, a new proposal x' can be either better or worse than x, depending on sign of $\delta E(x', x)$. In case of multi-objective optimization however, x' is preferably accepted if it is not dominated by other solutions, because when x' is better for one objective function, it may be worse for other objective functions. Therefore to adapt SA for multi-objective optimization, in initial works, objective functions are combined as a composite objective by weighted sum of the objective functions as Equation 5.2 below.

$$E(x) = \sum_{i=1}^{M} w_i f_i(x) \tag{5.2}$$

But in case of multi-objective optimization where the *true* Pareto front, that is, the complete set of non-dominated solutions of the problem is not known, relative weight of the objectives can be decided by estimated Pareto front only. In such case where the complete Pareto front is not known, there is no clear cut mechanism to choose weight w_i in advance [61]. In this weighted sum composite objective function based SA process, non-dominated solutions found so far are appended to an archive. This archive is used for finding other non dominated solutions by subsequent iterations. Though proof of convergence can be provided for multi-objective SA using scalar objective function with fixed weights as in Equation 5.2, it is not clearly defined how proofs of convergence can be obtained with changing weights for promoting exploration on the non-dominated front [61, 85].

In multi-objective optimization, the objective is to find the set of all non-dominating solutions based on some objective functions. Therefore, The dominance relations among solutions obtained and generated can be used to determine acceptability of a solution. In recent works [61, 85, 86] for adapting SA for multi-objective optimization, relative dominance measure of non-dominated solution front obtained so far on other solutions are used to define the energy function.

5.2.1 Energy function in terms of number of dominating solutions

For a known Pareto front, if a particular solution x is on the front, then value of energy function $E(x)$ is 0. In other words when a solution is not dominated by any solutions, energy function value is 0. When the distance of x from the true Pareto front increases, a greater portion of the front will dominate x and therefore solution x may be defined as having higher energy. If the energy function $E(x)$ for the SA process is defined using this concept, the weighting (w_i) of objective functions is not needed. Smith et al. in [61], therefore, presented a multi-objective optimization technique using SA based on this energy function, which is termed as Multi-objective Simulated Annealing (MOSA).

To define the energy function applying dominance by Pareto front, the

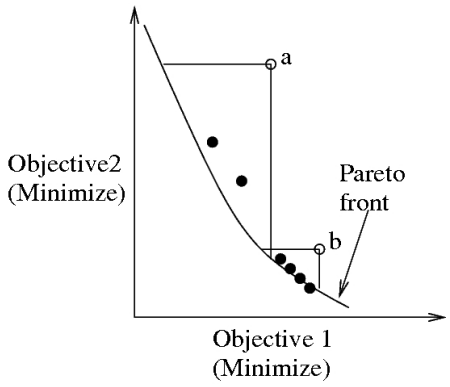

Figure 5-1: Number of dominating solutions in estimated Pareto front

true Pareto front is to be known during the optimization process. But as the true Pareto front is not available during the process, in MOSA [61] Smith et al. suggest to define the energy function by currently available Pareto front termed as current estimate of Pareto front. The current estimate of Pareto front $\tilde{\mathcal{F}}$ is defined as the union of currently available non dominated solution front $\mathcal{F}$, the current solution x and the newly generated solution x' expressed as Equation 5.3 below.

$$\tilde{\mathcal{F}} = \mathcal{F} \cup \{x\} \cup \{x'\} \tag{5.3}$$

Now, if $\tilde{\mathcal{F}}_x$ is the (set of) elements from (the set) $\tilde{\mathcal{F}}$ that dominate x, and $\tilde{\mathcal{F}}_{x'}$ is the elements from $\tilde{\mathcal{F}}$ that dominate x', then in MOSA, the energy difference between the proposed and current solution, x' and x, is defined as Equation 5.4.

$$\delta E(x', x) = \frac{(|\tilde{\mathcal{F}}_{x'}| - |\tilde{\mathcal{F}}_x|)}{|\tilde{\mathcal{F}}|} \tag{5.4}$$

In Equation 5.4, when the estimated front $\tilde{\mathcal{F}}$ is a non dominating set, the energy difference $\delta E(x', x)$ is zero. If $x' \prec x$ then the current solution x and the new proposed solution x' are included in $\tilde{\mathcal{F}}$ means $\delta E(x', x) < 0$. The value of the difference $(|\tilde{\mathcal{F}}_{x'}| - |\tilde{\mathcal{F}}_x|)$ is divided by $|\tilde{\mathcal{F}}|$ makes $\delta E(x', x)$ always less than unity and this strengthens the probability function. Therefore, it is ensured that new proposals moving in the estimated front towards the true Pareto front will be always accepted [61]. This energy function (5.4) is built on the concept that distant solutions from the true Pareto front are dominated by more number of solutions, and hence less probability of getting accepted. But in some cases like depicted in Figure 5-1 it is evident that - as the function is based on cardinality of $\tilde{\mathcal{F}}$, $\tilde{\mathcal{F}}_{x'}$, $\tilde{\mathcal{F}}_x$ and an estimated front only, solutions dominated by fewer elements will be of lower energy, and hence they are more likely to be accepted. Thus another advantage of this energy function is that sparsely populated regions of the front also will be explored.

Smith et al. present empirical evidence of convergence by MOSA using energy function expressed by Equation 5.4 in [61].

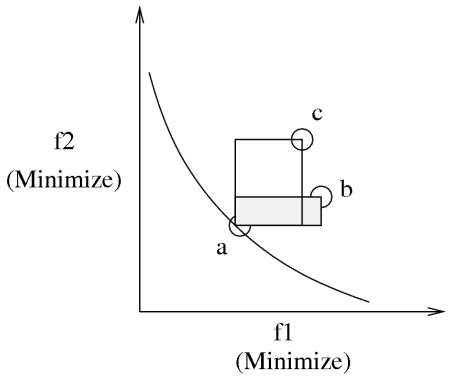

Figure 5-2: Domination during initializing an archive of non dominated solutions.

5.2.2 Energy function in terms of amount of domination

Bandyopadhyay et al. propose *amount of domination* for computing the acceptance probability of solutions in AMOSA algorithm [30]. In multi-objective SA process, initially there may be just one solution in the front. In this case, dominance measured by number of dominating solutions in the archived front for accepting newly generated solutions is not possible. In this case amount of domination for making decision on acceptance may be a better choice as seen in Figure 5-2. When -(i) *current-point* dominates *new-point* and k number of points from the Archive also dominate the new-point, or (ii) current-point and new-point are non-dominating to each other but new-point is dominated by $k(k \geq 1)$ points in the front, or (iii) new-point dominates current-point but $k(k \geq 1)$ points in the front dominate the new-point, then the amount of domination between the newly generated (solution) point and the current (solution) point is applicable for using as acceptance probability function instead of number of dominating points.

5.2.2.1 Amount of domination

In AMOSA, the concept of amount of domination is used for computing the acceptance probability of a new solution [30]. The amount of domination between two elements is defined as the amount (or volume) of space between them in the objective function space. For two solutions a and b of an optimization problem in objective function space of three objectives $f1$, $f2$ and $f3$ may be depicted as the shaded volume in Figure 5-3.

Using this concept, for given two solutions a and b, and $\mathbf{M}$ = number of objectives, Bandyopadhyay et al. in [30] defined the amount of domination as Equation 5.5.

$$\Delta dom_{a,b} = \prod_{i=1, f_i(a) \neq f_i(b)}^{M} \frac{(|f_i(a) - f_i(b)|)}{R_i} \tag{5.5}$$

Where R_i is the range of the i-th objective. The value of R_i may not be known a priori. In that case the solutions already obtained and kept in the Archive along

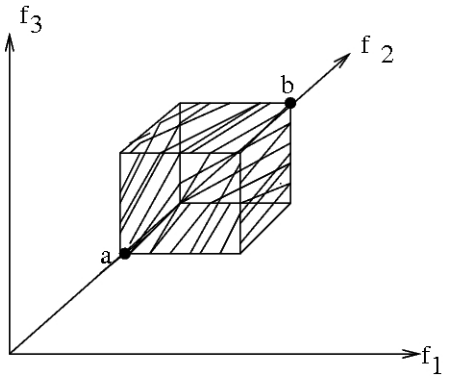

Figure 5-3: Amount of domination

with the current and the new proposed solution may be used for computing it. Thus in AMOSA $\Delta dom_{x',x}$ is used for computing the probability of acceptance between current solution x and the new proposal x'.

5.2.2.2 Accepting newly generated solutions

During the annealing process, at a system temperature T, a new solution or new point is generated by perturbing a randomly selected solution from the already obtained archive of non-dominating solutions termed as current point, and their domination status are checked with respect to the Archive. In multi-objective SA even if the new point is a dominated solution either by the current point or one or more points from the Archive, the point is accepted as current point for next iteration with probability within 0 and 1. Domination amount based probability functions are used for selecting current point for next iteration. The AMOSA in [30] suggests these probabilities as in Equation 5.6 and 5.8 below in respective cases ensuring probability between 0 and 1.

When the current point dominates the new point and $k(k \geq 0)$ points of the Archive dominate the new point, the new point is selected as the current point with probability in Equation 5.6.

$$Probability = \frac{1}{1 + exp(\Delta dom_{average} * T)} \tag{5.6}$$

where

$$\Delta dom_{average} = \frac{((\sum_{i=1}^{k} \Delta dom_{i,newpoint}) + \Delta dom_{newpoint,currentpoint})}{k+1} \tag{5.7}$$

In case current point and new point are non dominating with respect to each other but the new point is dominated by $k(k \geq 1)$ points in the Archive, then the new point is selected as the current point with probability in Equation 5.6, where $\Delta dom_{average} = \sum_{i=1}^{k} (\Delta dom_{i,newpoint})/k$. Here the current point may or may not be in the Archive.

In above two cases there will be probability that a new point is accepted as current point even though it is not in the non dominating Archive. If the current point is not in the Archive and new point dominates the current point, then it may also happen that $k(k \geq 1)$ points in the Archive dominate the new point. In this case the minimum of the difference of domination amounts between the new point and the dominating k points of the Archive, Δdom_{min} is computed. The point out of the k dominating points from the Archive that corresponds to the minimum dominance is selected as the current point with probability in Equation 5.8. Otherwise the new point is selected as the current point.

$$Probability = \frac{1}{1 + exp(-\Delta dom_{min})} \tag{5.8}$$

Thus, at the beginning of AMOSA process, one solution is selected as current point from the set of already found Archive of non dominating solutions at temperature $T = T_{max}$. The value of T is reduced at a very slow rate till it reaches T_{min} in the process. At every epoch of T, for specified number of iterations, a current point is selected randomly from the Archive and is perturbed to generate a new solution as new point and checked it's domination status with respect to the current point and other solutions already kept in the Archive. Other than the three types of domination status discussed above, there may be some other cases as well. In these cases, as stated below, solutions are added to the non dominating Archive.

When the new point is non dominating with respect to the current point as well as other solutions in the Archive, the new point is selected as the current point for next iteration and appended to the Archive. But if the new point is non dominating with each other with respect to the current point and the new point dominates $k(k > 1)$ points in the Archive, the new point is selected as the current point and added to the Archive, and the k dominated points of the Archive are removed.

If the new point dominates the current point where the current point is a member of the Archive and the new point is non dominating with respect to other points in the Archive (except the current point), then the current point is removed from the Archive. The new point is then accepted as the current point and added to the Archive.

There may be a case that the new point dominates the current point and the current point may or may not be in the Archive, but the new point also dominates $k(k \geq 1)$ other points in the Archive. In this case, all the dominated points in Archive are removed, and then, the new point is selected as the current point and appended to the Archive.

While continuing the AMOSA process from $T = T_{max}$ to $T = T_{min}$, where T is reduced as $T = \alpha T$, α being the cooling rate after every epoch of T, it may happen that the size of the Archive may go on increasing. To keep the size of the Archive within limit during the process, AMOSA suggests a clustering based

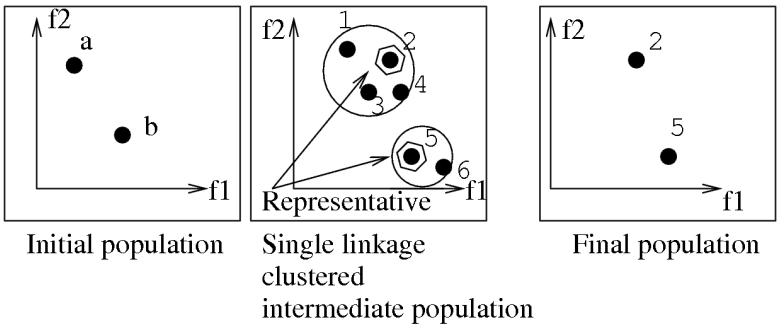

Figure 5-4: Clustering to reduce solution population while minimizing objective functions $f1$ and $f2$

technique.

5.2.2.3 Clustering for maintaining diversity of solutions in intermediate generations

In AMOSA, for controlling the number of solution population in the Archive enforcing diversity among the solutions, Bandyopadhyay et al. in [30] suggested clustering of the solutions in the Archive and then selecting representative solutions from each of these clusters. This may be illustrated by Figure 5-4 for minimizing two objective functions $f1$ and $f2$. AMOSA uses two user defined parameters for controlling the size of solution population in the Archive. One is SL for *soft limit* of population and the other one is HL for *hard limit* of population in the Archive. At the beginning of the AMOSA process, $\gamma \times SL$ number of solutions are randomly generated where $\gamma > 1$. Each of these random solutions are then refined by perturbing for fixed number of times to get a solution that dominates the previous one. Upon finding a solution that dominates the original solution, it is added to a list. After continuing this for all $\gamma \times SL$ number of solutions, all the dominated solutions are discarded and non dominating solutions are kept in an Archive. If the size of this Archive is more than HL, then the solution population in the Archive is clustered to select and keep only HL number of solutions in the Archive. The main AMOSA process now starts with this Archive.

During the main AMOSA process, which starts with HL number of non dominating solutions in the Archive, whenever the size of the Archive becomes SL (SL¿HL), the solutions are clustered to make group of HL number of clusters. From each of these HL number of clusters, the solution or point within each cluster whose average distance to other member of the cluster is the minimum is considered as the representative member of the cluster. These HL number of representative solutions are selected and kept in the Archive discarding the other solutions. In next iteration of the process, one solution from the Archive is picked up as current point to continue the process as discussed above. In [30], Bandyopadhyay et al. use *Single linkage* clustering algorithm [91] where every point of the population is first linked to another point having the shortest distance between them to form clusters and then the distance between any two clusters corresponding to the shortest link

between them are grouped to form clusters. This clustering process goes on till it reaches the number of clusters desired to obtain in the process. The complexity of Single linkage clustering is found to be of $O(SL^2 \times log(SL))$ [30, 92].

5.3 Representation of the View Selection Problem for Applying AMOSA

In case of Big data DFS framework, the data warehouse is basically very few *Key-Value* paired tables or semi-structured tables with or without *primary key* based relations or indices where queries are triggered as shown in Figure 5-5 for processing. These queries are composed of some sub-queries and aggregation functions. For analytical processing some queries are triggered very frequently on this Big data which share some sub-queries or sub-expressions and/or aggregation functions that are considered as the candidate views for materializing to speedup the total query processing for an application. In DFS the data access pattern is mainly dominated by data transfer rate and MapReduce costs unlike RDBMS based warehousing where the data access pattern is mainly dominated by *seeks* and *seek time*. It has been presented (in Chapter 4) that if the results of some sub-queries and aggregate functions used in these queries are materialized or saved, then in subsequent execution of the queries, MapReduce overheads of executing these sub-queries or views are not to be incurred. In DFS based Big data management, block size and split size are fixed. Therefore smaller number of bigger views are preferred for materializing.

5.3.1 Representing view selection problem for Big data

The materialized view selection problem for Big data has been stated (in Chapter 4) as - for a set of n number of frequent queries $Q = \{q_1, q_2, \cdots, q_n\}$ on a data warehouse, where V is the set of m intermediate views generated by Q, if $V' \subseteq V$ is the set of views $V' = \{v_1, v_2, \cdots, v_p\}$ that are materialized, then if $\sum_{i=1}^{p} M_{v_i}$ is the MapReduce cost of processing V', $C_{\varnothing}^{Q} = \sum_{i=1}^{n} M_{q_i}$ is the total query processing cost of Q without materializing, and U_{v_i}, $i = 1, 2 \cdots, p$ are the maintenance MapReduce overheads for the set of materialized views $v_i \in V'$, $i = 1, 2 \cdots p$, the following objective functions are to be minimized-

$$C_{V'}^{Q} = C_{\varnothing}^{Q} - \sum_{i=1}^{p} M_{v_i} \tag{5.9}$$

$$U(V') = \sum_{i=1}^{p} U_{v_i} \tag{5.10}$$

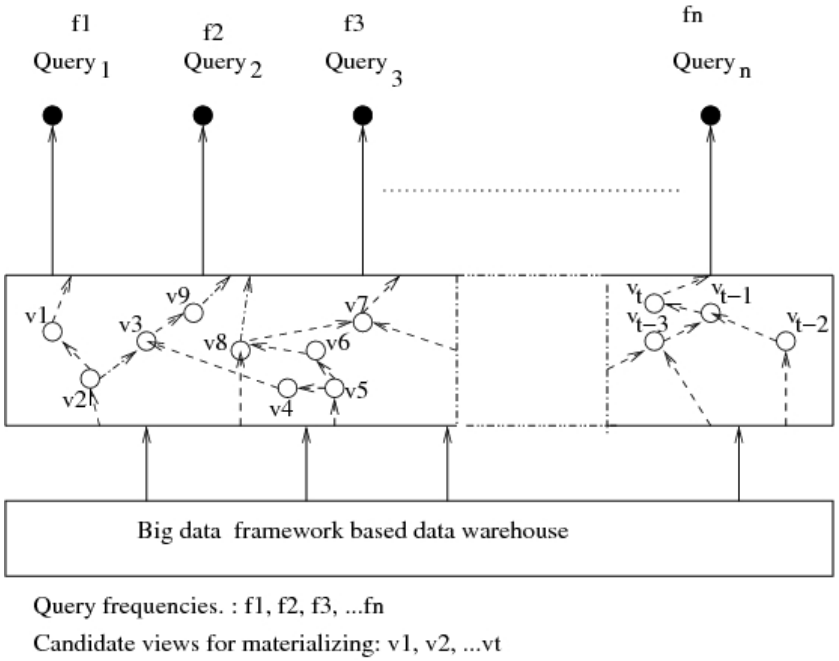

Figure 5-5: Big data query responding

and

$$|V'| = p \tag{5.11}$$

with a constraint on the total space required for materializing p number of views $A_{V'} = \sum_{i=1}^{p} A_{v_i}$, A_{v_i} being the storage space required by ith materialized view.

From the objective functions defined by Equations 5.9, 5.10 and 5.11, and the associated space constraint, it is evident that the objective function values of this optimization problem depend on the set of views selected from the candidate set of views for materializing. Therefore for applying AMOSA, by using this notion of the problem, a solution is to be represented as a set of views.

5.3.2 Solution representation

Perturbing an already obtained solution for generating a new solution for applying AMOSA means removing one or more views from the current set of views, or adding one or more views, or both adding and removing few views from a solution set of views. Therefore, to make solution generation and perturbation easy, instead of representing solutions as sets of different cardinality, they may be represented as vectors with number of dimensions as the total number of candidate views and each dimension value representing presence or absence of a view. Thus in this vector representation of solution, a solution may be represented as a string of bits. Perturbation can be performed by just changing one or more dimensions of the vector, i.e just changing values of bits from 1 to 0 or 0 to 1. Selected number of views as expressed by Equation 5.11, may be measured by counting the number of dimensions where the corresponding bit in the string is 1.

5.4 AMOSA for Materialized View Selection

The AMOSA as presented in [30] has been customized for handling the materialized view selection problem and is referred hence forth as AMOSA for Materialized View Selection (AMOSA-MVS). In AMOSA-MVS algorithm, the solution vectors are represented as string of bits. The diversity of the solution population is maintained in the solution space instead of the objective function space to yield a limited number of non dominated solutions having maximum dissimilarity with respect to combination of views selected. Instead of using *Single-Linkage* clustering for controlling solution population by maintaining their diversity, AMOSA-MVS uses dissimilarity based sorting of solutions which is discussed in Section 5.4.2. The details of AMOSA-MVS is discussed in sub-sections of this section.

5.4.1 Initializing the archive of solutions

At the beginning of the process, a solution vector is randomly generated. Then a list of $\gamma \times SL$, $(\gamma > 1)$ number of solutions are generated by changing random number of dimensions of the initial solution vector. For doing this, a random integer i is generated in the range $[1, D]$, where D is the number of dimensions of the solution vector. For i iterations, again a random integer j is generated in the range $[1, D]$ and the j-th dimension of the solution vector, i.e the j-th bit of the solution vector, is changed from 0 to 1 or 1 to 0. This process of generating new solutions continued till $\gamma \times SL$ number of different solution vectors are generated. Each of these solutions are then perturbed again for a specific number of iterations to generate a new solution vector that dominates the original solution. Whenever during the iteration process a new dominating solution is found, the iteration stops and the original solution is replaced by the newly generated dominating solution. If a new dominating solution is not found even after the specified number of iterations, the original solution remains in the list. Now from the final list of solutions, all the dominated solutions are removed and all the non dominated solutions are stored in an Archive. A hard limit, HL, is defined as the maximum number of non-dominated and sufficiently different solutions that are to be yielded by the algorithm at the end. Therefore, at the beginning of the annealing process, the Archive size is restricted to HL. For doing this, the solutions in the Archive are sorted on their dissimilarity measure with respect to other solutions in the Archive and HL number of most dissimilar solutions are filtered out to keep in the Archive as discussed in Section 5.4.2 below.

5.4.2 Enforcing diversity in solution space for restricting the archive size

Any two solution vectors in the Archive may be very close in objective function space whereas they may be far from each other in their solution vector space. Therefore, if diversity among solution vectors of the Archive is enforced in objective

Algorithm 8: Initialization of solution population *Archive*

Require: SL,HL,D,γ,iterations, space constraint, $C^Q_{\emptyset}$, $M_{v_{i=1,\cdots,m}}$, $U_{v_{i=1,\cdots,m}}$, $A_{v_{i=1,\cdots,m}}$
Ensure: *Archive* of maximum HL number of non dominated solutions
1: $Curr\_Sol \leftarrow$ (Randomly generated string of D number of bits)
2: $Archive\_Size = 0$
3: **while** $Archive\_Size < (\gamma \times SL$ -1) **do**
4: $New\_Sol \leftarrow$ perturb($Curr\_Sol$)
5: **if** New_Sol satisfies the space constraint and $New\_Sol \notin Archive$ **then**
6: $Archive\_Size = Archive\_Size + 1$
7: reallocate *Archive* for size $Archive\_Size$
8: $Archive[Archive\_Size\text{-}1] \leftarrow New\_Sol$
9: **end if**
10: **end while**
11: **for** $i = 0$ to ($Archive\_Size$-1) **do**
12: $Curr\_Sol \leftarrow Archive[i]$
13: **for** $j = 1$ to iterations **do**
14: $New\_Sol \leftarrow$ perturb($Curr\_Sol$)
15: **if** $New\_Sol \notin Archive$ **then**
16: compute $C^Q_{New\_Sol}$, $C^Q_{Curr\_Sol}$, U_{New_Sol}, U_{Curr_Sol}, $|New\_Sol|$,$|Curr\_Sol|$
17: **if** ($New\_Sol \prec Curr\_Sol$) and (New_Sol satisfies the space constraint) **then**
18: $Curr\_Sol \leftarrow New\_Sol$
19: break
20: **end if**
21: **end if**
22: **end for**
23: $Archive[i] \leftarrow Curr\_Sol$
24: **end for**
25: **for** $i = 0$ to ($Archive\_Size$-1) **do**
26: $Curr\_Sol \leftarrow Archive$[i]
27: **for** $j = 0$ to ($Archive\_Size$-1) **do**
28: **if** $i \neq j$ **then**
29: $New\_Sol \leftarrow Archive$[j]
30: Compute $C^Q_{New\_Sol}$, $C^Q_{Curr\_Sol}$, U_{New_Sol}, U_{Curr_Sol}, $|New\_Sol|$,$|Curr\_Sol|$
31: **if** ($New\_Sol \prec Curr\_Sol$) **then**
32: Remove $Curr\_Sol$ from *Archive*
33: $Archive\_Size \leftarrow (Archive\_Size$ -1)
34: reallocate *Archive* for size $Archive\_Size$
35: break
36: **end if**
37: **end if**
38: **end for**
39: **end for**

Algorithm 9: Initialization of solution population *Archive* - (continued)

40: **if** $Archive\_Size > HL$ **then**
41: **for** $i = 0$ to ($Archive\_Size$-1) **do**
42: MaxDist[i] ← Maximum distance w.r.t $Archive[i]$ and all other solutions in *Archive*
43: **end for**
44: Sort *Archive* w.r.t corresponding MaxDist in descending order
45: keep only top HL solutions discarding others by re-allocating the *Archive*
46: $Archive\_Size \leftarrow HL$
47: **end if**
48: **Return** *Archive*

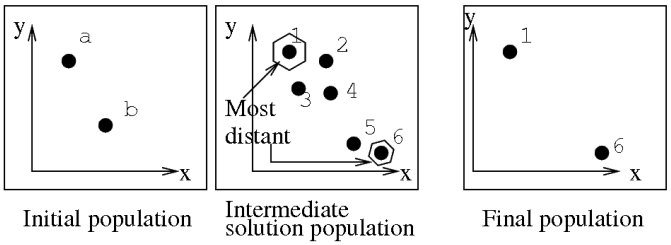

Figure 5-6: Restricting solution population in AMOSA-MVS

function space for reducing or restricting the solution population to HL number of solutions, some solutions with a very different set of views may be lost in the process. Therefore, in AMOSA-MVS the diversity of solution vectors are maintained in solution space instead of objective function space while controlling the solution population in the Archive.

For restricting the Archive size to HL by enforcing diversity in solution space, the distance from each solution of the Archive to each of the other solutions in the Archive are measured for finding the maximum distance value of each solution in the Archive with respect to other solutions. Upon finding the maximum distance value of each solution, say Max_i for ith solution in the Archive, the solution vectors are sorted in descending order of their maximum distance Max_i. From the sorted list of solutions the top HL number of solutions are retained in the Archive discarding the rest of the solutions as depicted in Figure 5-6 for two dimensional solution vectors. During the main AMOSA process, the Archive size is allowed to increase up to the limit SL, where $SL > HL$, and then the Archive size is reduced to HL by this method. But during initialization time, as suggested in [30], if the size of the initially generated and filtered Archive size is more than HL, the size of the Archive is reduced by the said method even if the size is less than SL.

5.4.3 The main process of AMOSA-MVS

From the initial Archive, a solution vector is randomly chosen as current point, say it is denoted as *current_point*, for the initial temperature $T = T_{max}$. A new point,

say it is denoted as *new_point*, is generated by altering some of the dimensions of the *current_point* randomly. That is, a random number of views that are present in the current solution vector are dropped and equal number of views that are not selected in the current solution are added to generate a new solution vector *new_point*. In case of conventional RDBMS based data warehousing applications the total query processing cost, materialized view maintenance cost and space cost for the considered MVPP DAG are to be evaluated for both the solution vectors. Similarly, in case of Big data warehousing with DFS processing framework, the total query processing cost as in Equation 5.9, materialized view maintenance cost as in Equation 5.10, number of views selected as defined by Equation 5.11 and total space requirement by the views selected are computed for both the solutions. Thus by computing these cost functions, the domination status between the *current_point* and the *new_point* is checked. Bandyopadhyay et al. applied 5.6 and 5.8 as probability functions to ensure probability within 0 and 1 for accepting a dominated solution as a solution for keeping in next generation [30].

If the *current_point* dominates the *new_point*, expressed as *current_point* $\prec$ *new_point*, and $k(k \geq 0)$ number of solution vectors of the Archive dominate the *new_point*, then the *new_ point* is selected as *current_point* with probability defined by Equation 5.6. The average domination $\Delta dom_{average}$ is computed as in Equation 5.7. For computing $\Delta dom_{average}$, the amount of domination between the two solution vectors *current_point* and *new_point* is computed as follows. For three objectives of materialized view selection in case of DFS and MapReduce frame work say, $(i.)$ $f_1(current\_point)$ and $f_1(new\_point)$ are the objective functions for total query processing cost of the *current_point* and the *new_point*, $(ii.)$ $f_2(current\_point)$ and $f_2(new\_point)$ are the objective functions for view maintenance cost of solutions *current_point* and *new_point* and $(iii.)$ $f_3(current\_point)$ and $f_3(new\_point)$ are the objective functions for number of views selected in solutions *current_point* and *new_point* respectively, the amount of domination may be computed as :

$$\begin{aligned}\Delta dom_{current\_point,new\_point} = (|f_1(current\_point) - f_1(new\_point)|)/R_1 \\ \times(|f_2(current\_point) - f_2(new\_point)|)/R_2 \\ \times(|f_3(current\_point) - f_3(new\_point)|)/R_3,\end{aligned}$$

where the ranges R_1, R_2 and R_3 are the difference between maximum objective function values and minimum objective function values found for solutions in the Archive as well as the *current_point* and the *new_point* for the respective objective functions f_1, f_2 and f_3. When k increases, $\Delta dom_{average}$ also increases, because dominating points farther away from the *current_point* means increase in the amount of domination. Therefore Bandyopadhyay et al. in [30] suggest using $\Delta dom_{average}$ for the probability value in this case since dominating points farther away from the *current_point* also contribute to the probability.

If the *current_point* and the *new_point* are non dominating with respect to each other but the *new_point* is dominated by $k(k \geq 1)$ points of the Archive, then the *new_point* is selected as the *current_point* with probability expressed by Equation 5.6. Here $\Delta dom_{average} = \sum_{i=1}^{k}(\Delta dom_{i,new\_point})/k$. In case the *current_point*

and the *new_point* are non dominating and the *new_point* and the solutions in the Archive are also non dominating with them, the *new_point* is selected as the *current_point* and the solution is added to the Archive. The moment Archive size becomes more than SL, the number of solutions in the Archive are reduced to HL as discussed in Section 5.4.2. But in case the *current_point* and the *new_point* are non dominating with respect to each other and the *new_point* dominates $k(k > 1)$ solutions in the Archive, the *new_point* is added to the Archive and selected as *current_point* and all the k dominated solutions in the Archive are removed.

In case the *new_point* dominates the *current_point*, and $k(k \geq 1)$ number of solutions in the Archive dominate the *new_point*, the solution in the Archive which corresponds to the minimum difference of amount of domination between the *new_point* and the *current_point* is selected as the *current_point* with probability expressed by Equation 5.8. In this case obviously the *current_point* is not in the Archive. If the probability is not equal to which is evaluated by Equation 5.8, the *new_point* is selected as the *current_point*. But if the *new_point* dominates the *current_point* and the *new_point* neither dominates any other solutions in the Archive nor any solution in the Archive dominates the *new_point*, then the *new_point* is accepted as the *current_point* and added to the Archive and if the *current_point* is in the Archive, (as it may happen here,) it is to be removed. Here, if the *current_point* was not in the Archive, number of solutions may become more than SL after appending the *new_point*. After appending If number of solutions in the Archive becomes more than SL, then some of the solutions are removed from the Archive maintaining diversity among the solutions as discussed in Section 5.4.2. When the *new_point* dominates the *current_point* as well as $k(k \geq 1)$ number of solutions in the Archive, all the k dominated solutions of the Archive are removed and the *new_point* is selected as *current_point* for next iteration and added to the Archive.

This process is executed for a specified number of iterations in each temperature T. T is then reduced by a cooling rate α , i.e, T is decremented as $T = \alpha \times T$. The process continues till it reaches a specified minimum temperature T_{min}. At T_{min}, the process stops and the Archive contains the final set of non dominated solutions.

5.4.4 Parameter selection

The parameters for SA based algorithms are selected based on their applications, i.e, they depend on the dimensions, objectives and required performance level of the problem. The parameters for AMOSA that are to be optimized are - initial temperature T_{max}, the terminating temperature of the annealing process T_{min}, cooling schedule $\alpha(0 < \alpha < 1)$, and number of iterations in each temperature T. By observing the performance analysis and suggestions in [61] and [30], the value of α is chosen in the range [0.5, 0.9]. The maximum temperature T_{max}, and minimum temperature T_{min} are preferred 200 and around 10^{-5} respectively for comparing this algorithm's performance with the performances of evolutionary algorithms like NSGA-II with around 100 initial population or Archive size. The number

Algorithm 10: Archived Multi-Objective Simulated Annealing for Materialized View Selection (AMOSA-MVS)

Require: T_{max},T_{min}, HL, SL, iterations, α, D, $C^Q_{\varnothing}$, $M_{v_{i=1,\cdots,m}}$, $U_{v_{i=1,\cdots,m}}$, $A_{v_{i=1,\cdots,m}}$, space constraint

Ensure: *Archive* of mutually non-dominated solutions

Initialize *Archive* as an array of maximum HL number of distinct solution strings of D bits
$Archive\_Size \leftarrow$ length_of($Archive$)
$Curr\_Sol \leftarrow$ a randomly selected solution from the *Archive*
$T \leftarrow T_{max}$
while $T > T_{min}$ **do**
  for $i = 0$ to $i <$ iterations **do**
    repeat
      $New\_Sol \leftarrow$ perturb($Curr\_Sol$)
    until New_Sol satisfies space constraint
    Compute $C^Q_{New\_Sol}$, $C^Q_{Curr\_Sol}$, U_{New_Sol}, U_{Curr_Sol}, number_of_views(New_Sol), number_of_views(Curr_Sol)
    if ($Curr\_Sol \prec New\_Sol$) and ($k(k \geq 0)$ solutions in $Archive \prec New\_Sol$) **then**
      if random(0,1)$< \frac{1}{1+exp(\Delta dom_{average} \times T)}$ **then**
        $Curr\_Sol \leftarrow New\_Sol$
      end if
    else if ($Curr\_Sol \not\prec New\_Sol$) and ($New\_Sol \not\prec Curr\_Sol$) **then**
      if $k(k \geq 1)$ solutions in $Archive \prec New\_Sol$ **then**
        if random(0,1)$< \frac{1}{1+exp(\Delta dom_{average} \times T)}$ **then**
          $Curr\_Sol \leftarrow New\_Sol$
        end if
      else if $New\_Sol \prec k(k \geq 1)$ solutions in *Archive* **then**
        $Curr\_Sol \leftarrow New\_Sol$
        $Archive\_Size \leftarrow Archive\_Size + 1$
        re-allocate *Archive* for size $Archive\_Size$
        $Archive[(Archive\_Size\text{-}1)] \leftarrow Curr\_Sol$
        Remove all k dominated solutions from *Archive* by re-allocating
        $Archive\_Size \leftarrow Archive\_Size$ - k
      else
        $Curr\_Sol \leftarrow New\_Sol$
        $Archive\_Size \leftarrow Archive\_Size + 1$
        re-allocate *Archive* for size $Archive\_Size$
        $Archive[(Archive\_Size\text{-}1)] \leftarrow Curr\_Sol$
        if $Archive\_Size > SL$ **then**
          for $i = 0$ to ($Archive\_Size$-1) **do**
            $MaxDist$[i]$\leftarrow$ Maximum distance w.r.t $Archive[i]$ and all other solutions in *Archive*
          end for
          Sort *Archive* w.r.t corresponding $MaxDist$ in descending order
          keep only top HL solutions discarding others by re-allocating the *Archive*

Algorithm 11: *Continued- second page* - Archived Multi-Objective Simulated Annealing for Materialized View Selection (AMOSA-MVS).

38: $Archive\_Size \leftarrow HL$
39: **end if**
40: **end if**
41: **else if** $New\_Sol \prec Curr\_Sol$ **then**
42: **if** $k(k \geq 1)$ solutions in $Archive \prec New\_Sol$ **then**
43: **if** random(0,1)$< \frac{1}{1+exp(-\Delta dom_{min})}$ **then**
44: $Curr\_Sol \leftarrow$ solution in $Archive$ corresponding to Δdom_{min}
45: **else**
46: $Curr\_Sol \leftarrow New\_Sol$
47: **end if**
48: **else if** $New\_Sol \prec k(k \geq 1)$ solutions in $Archive$ **then**
49: $Curr\_Sol \leftarrow New\_Sol$
50: $Archive\_Size \leftarrow Archive\_Size$-$k$
51: Remove k dominated solutions and re-allocate $Archive$
52: $Archive\_Size \leftarrow Archive\_Size$+1
53: re-allocate $Archive$ for size $Archive\_Size$
54: $Archive[(Archive\_Size\text{-}1)] \leftarrow Curr\_Sol$
55: **else**
56: **if** $Curr\_Sol \in Archive$ **then**
57: $Archive\_Size \leftarrow Archive\_Size$-1
58: Remove $Curr\_Sol$ from $Archive$ and re-allocate
59: **end if**
60: $Archive\_Size \leftarrow Archive\_Size + 1$
61: re-allocate $Archive$ for size $Archive\_Size$
62: $Curr\_Sol \leftarrow New\_Sol$
63: $Archive[(Archive\_Size\text{-}1)] \leftarrow Curr\_Sol$
64: **if** $Archive\_Size > SL$ **then**
65: **for** $i = 0$ to $(Archive\_Size\text{-}1)$ **do**
66: $MaxDist$[i]$\leftarrow$ Maximum distance w.r.t $Archive[i]$ and all other solutions in $Archive$
67: **end for**
68: Sort $Archive$ w.r.t corresponding $MaxDist$ in descending order
69: keep only top HL solutions discarding others by re-allocating the $Archive$
70: $Archive\_Size \leftarrow HL$
71: **end if**
72: **end if**
73: **end if**
74: **end for**
75: T=$\alpha \times T$
76: **end while**
77: **if** $Archive\_Size > SL$ **then**
78: **for** $i = 0$ to $(Archive\_Size\text{-}1)$ **do**
79: $MaxDist$[i]$\leftarrow$ Maximum distance w.r.t $Archive[i]$ and all other solutions in $Archive$

Algorithm 12: *Continued- third page* - Archived Multi-Objective Simulated Annealing for Materialized View Selection (AMOSA-MVS).

end for
Sort *Archive* w.r.t corresponding $MaxDist$ in descending order
keep only top HL solutions discarding others by re-allocating the *Archive*
$Archive\_Size \leftarrow HL$
end if
Return *Archive*

of iterations in each temperature of 100 to 500 are good choice for analyzing performances with other evolutionary algorithms and SA based algorithms because similar parameter values have been used for comparing by other algorithms [27,30, 60, 61]. In popular evolutionary algorithms and multi-objective SA using binary encoded solution vectors of three objectives optimization on standard test problem, 10 to 20 dimensions are mostly used [27, 30, 60, 61]. In case of view selection problem, the number of dimensions or number of candidate views for selection may be of much higher than these standard test problems. But the cost functions or objective functions are of less complex than the standard test problems that have been used in [27, 30, 60, 61]. Therefore, higher dimensional solution vectors are used in this application for comparing performances among the algorithms, as presented in Section 5.5.

5.4.5 Complexity analysis

The time complexities of different AMOSA-MVS processes are -

- Initializing the Archive at the beginning of the AMOSA process is $O(SL)$. Where SL is the maximum limit, termed as the soft limit, on maximum number of solutions that may be present in the archive of solutions and HL is the hard limit on number of solution population to which the number of solutions are reduced when it becomes more than SL.
- To check the domination status between two solutions with M objectives is $O(M)$.
- Domination status checking process between a solution and all solutions already in the Archive is $O(M \times SL)$.
- Computing maximum distance (Max_i) from each solution to other solutions in the Archive is $O(SL^2)$.
- Sorting the solutions in the Archive on maximum distance values (Max_i) of the solutions is $O(SLlogSL)$.

The computation of maximum distance of each solution to other solutions in the Archive and then the solutions are to be sorted on this maximum distance values assigned to them in following 3 cases [30].

Case 1. When non dominated solutions in the initial Archive is more than HL.

Case 2. After each $(SL - HL)$ number of iterations.

Case 3. At the end of the annealing process, if the size of the Archive becomes more than HL.

Therefore maximum number of times the computation of maximum distance measure of each solution and sorting of solutions on maximum distance assigned to each solution are to be done equals to $(Total\ iterations/(SL - HL)) + 2$. Therefore the complexity of maximum distance measure computation and sorting of solutions based on this is:

$$O((\frac{Total\ iterations}{(SL - HL)}) \times (SL^2 + SLlogSL)).$$

Thus the total complexity becomes

$$O((Total\ iterations) \times (SL + M + M \times SL) + (\frac{Total\ iterations}{(SL - HL)} \times (SL^2 + SLlogSL))).$$

Let $HL = N$. Then for $SL = \beta \times HL$, where $\beta \geq 2$, $SL = \beta N$. Therefore, the complexity can be expressed as

$$\begin{aligned} &O((Total\ iterations) \times (\beta N + M + M \times \beta N + \frac{1}{\beta N - N} \times (\beta^2 N^2 + \beta N log \beta N))) \\ &= O((Total\ iterations) \times (\beta N + M + M \times \beta N + \frac{\beta N}{\beta N - N} \times (\beta N + log \beta N))) \\ &= O((Total\ iterations) \times (\beta N + M + M \times \beta N + \frac{\beta}{\beta - 1} \times (\beta N + log \beta N))) \\ &= O((Total\ iterations) \times (N + M + M \times N + (N + logN))) \end{aligned}$$

As N and M are less than MN, the overall complexity may be expressed as

$$O((Total\ iterations) \times (MN + logN)) \tag{5.12}$$

In materialized view selection problem for Big data warehouse with MapReduce and DFS framework, three objectives are considered i.e $M = 3$. Therefore in this application the overall complexity is

$$O((Total\ iterations) \times (N + logN)) \tag{5.13}$$

The complexity of NSGA-II and Multi-objective DE for Binary Encoded Solutions for materialized view selection have been found to be $O((Total\ iterations)N^2)$. The complexity of generalized AMOSA is $O((Total\ iterations) \times N \times (M + logN))$. Therefore if Single-Linkage clustering based AMOSA presented in [30] is used for

materialized view selection, the complexity will be $O((Total\ iterations) \times (N \times logN))$.

5.4.6 Convergence

Though there have been some convergence proofs for multi-objective evolutionary algorithms [63, 93], in case of Simulated Annealing (SA) based non-deterministic global multi-objective optimizers the proof of convergence is yet to be well developed. In [64], it has been proved that with a suitable choice of the acceptance probabilities, SA for multi-objective optimization yields asymptotic convergence. But this proof is based on transforming multi-objective optimization by combining the objectives as a weighted sum of objectives and this composite objective is then used as the energy to be minimized by scalar SA optimizer.

Table 5.1: Convergence (γ)

Number of queries	Number of candidate views	AMOSA-MVS γ	MODE-BE γ	NSGA-II γ
109	51	0.6549	0.2229	0.2874
60	51	0.8877	0.2732	0.1235
50	51	0.5967	0.7592	0.1342
20	25	1.6296	0.5317	1.5023
	Average	0.942225	0.44675	0.51185

The convergence measure γ presented in [32] is used for measuring the extent of convergence by an algorithm to a known set of Pareto optimal solutions. Smaller γ value means lesser distance from the true Pareto front. But the non-deterministic multi-objective optimization techniques are applied on those problems, where the true Pareto optimal solutions are not known. Therefore the convergence measure γ may be computed with respect to a set of uniformly spaced solutions from a set of Pareto optimal solutions obtained by other accepted algorithms for comparative analysis. In this case the measure γ reflects the relative convergence quality only. While experimenting with the setup and data sets presented in Section 5.5 and with algorithm specific parameters in Subsection 5.5.2.1 for AMOSA-MVS, MODE-BE and NSGA-II in selection of views to materialize in data warehouse by binary encoded solution representation, the convergence measure γ are evaluated as presented in Table 5.1. The convergence measures presented in Table 5.1 are evaluated by considering the results where the maximum number of solutions remain non dominated with respect to all non dominated solutions obtained by other algorithms while executing the implementations for 20 times. It has been observed from Table 5.1 that the average value of γ for AMOSA-MVS is 0.942225 with 0.5967 being the minimum and 1.6296 being the maximum, MODE-BE is 0.44675 with 0.2229 being the minimum and 0.7592 being the maximum and that of NSGA-II is 0.51185 with 0.1235 being the minimum and 1.5023 being the maximum. The convergence measure γ by Binary-coded NSGA-II for test problems ZDT3, ZDT4 and ZDT6 suggested by

Zitzler et al. in [75] are found to be 0.043411, 3.227636 and 7.806798 respectively by Deb et al. [32] where 30 bits of decision variables are used in solution strings of these problems with two objective functions. By observing these empirical data, the AMOSA-MVS in selecting views for materializing is accepted as asymptotic to the true Pareto front.

5.5 Experimental Results and Performance Analysis

The comparative performance of AMOSA-MVS with respect to MODE-BE and NSGA-II has been reported in this section with experimental setup, test data sets, parameters, performance comparison metrics and obtained results in following sub-sections.

5.5.1 Experimental setup and test data sets

In this work, Hortonworks Data Platform (HDP) version 2.0.6 with Hortonworks Sandbox version 2.0 VMware for 64 bit CentOS operating system workstation of version 6.5-7.x virtual machine [83] have been used. Hadoop version 2.2.0 and Hive version 0.12.0 of Apache [80] have been used for executing HiveQL queries to generate log files for extracting query processing, view processing, view maintenance MapReduce CPU time and the size of candidate views for materializing. HiveQL queries on Lahman baseball database [84] and associated intermediate views, as presented in Chapter 4 have been used for our experimentation. Other than the 20 queries and 25 associated views presented in Chapter 4 and presented as *data set 4* in Section 5.5.1.1 below, which was implemented for a recommendation system for selecting views to materialize using real-life situation log-file containing different Map-Reduce and space costs against queries and views implemented in real-life Lahman baseball database [84], another set of data, referred as *data set 1* in Section 5.5.1.1 is generated for 109 queries and 51 candidate views for experimenting with higher dimensional data. Out of the data set 1, two other data sets have been generated for 50 and 60 queries sharing the 51 candidate views of data set 1.

The MapReduce CPU cost of processing the queries, materialized view creation and maintenance of the materialized views and sizes of the candidate views for materializing are extracted from the system and related log-file. The test queries are designed such a way that the shared sub-queries of selection, join and projection and other aggregation functions for considering as candidate materialized views are easily distinguishable. These test-data are then entered manually as input to the three different materialized view selection systems that uses AMOSA-MVS, Multi-Objective Differential Evolution Algorithm using Binary Encoded Data (MODE-BE), and NSGA-II for analyzing their performances.

5.5.1.1 Test data sets

The different MapReduce costs and view sizes for queries and shared views that have been used as input to the materialized view selection programs are presented below. Here labels $q_1, q_2 \cdots q_{109}$ are used to represent the queries and $v_1, v_2 \cdots v_{51}$ are the labels representing the sub-queries or views. The MapReduce CPU time costs are presented as pairs of labels representing queries or views and corresponding costs in Seconds. Similarly the view sizes are presented in terms of Mega Bytes(MB).
Data set 1: This data set is generated for 109 queries and 51 candidate views synthesized for experimenting with higher dimensional data in the experimental setup.

- **Query processing MapReduce CPU time in Seconds:** {q_1, 15.5},{q_2, 6}, {q_3, 14.57}, {q_4, 18.42}, {q_5, 19.92}, {q_6, 27.08}, {q_7, 29.67}, {q_8, 36.02}, {q_9, 27.73 }, {q_{10}, 30.65}, {q_{11}, 27.78}, {q_{12}, 30.01 }, {q_{13}, 24.24 }, {q_{14},23.49 }, {q_{15}, 21.22}, {q_{16},27.97 }, {q_{17}, 31.51 }, {q_{18}, 27.08}, {q_{19}, 24.25}, {q_{20}, 24.25 }, {q_{21}, 22.57 }, {q_{22}, 24.6}, {q_{23}, 23.26}, {q_{24}, 24.09}, {q_{25}, 25.88}, {q_{26}, 25.18}, {q_{27}, 28.74}, {q_{28}, 28.74}, {q_{29}, 33.54}, {q_{30}, 29.17}, {q_{31}, 25.46}, {q_{32}, 24.79}, {q_{33}, 17.41}, {q_{34}, 14.49}, {q_{35}, 12.54}, {q_{36}, 18.14}, {q_{37}, 14.21}, {q_{38}, 10.67}, {q_{39}, 6.67 }, {q_{40}, 10.72}, {q_{41}, 6.01}, {q_{42}, 14.3}, {q_{43}, 13.15}, {q_{44}, 10.09}, {q_{45}, 9.15}, {q_{46}, 11.28}, {q_{47}, 10.41}, {q_{48}, 10.41}, {q_{49}, 10.41}, {q_{50}, 10.77}, {q_{51}, 10.46}, {q_{52}, 7.58}, {q_{53}, 10.77}, {q_{54}, 13.71}, {q_{55}, 17.01}, {q_{56}, 5.55}, {q_{57}, 15.84}, {q_{58}, 10.66}, {q_{59}, 12}, {q_{60}, 11.36}, {q_{61}, 8.52}, {q_{62}, 10.89}, {q_{63}, 8.52}, {q_{64}, 10.39}, {q_{65}, 10.39}, {q_{66}, 11.55}, {q_{67}, 10.39}, {q_{68}, 11.12}, {q_{69}, 10.18}, {q_{70}, 6.64}, {q_{71}, 15.53}, {q_{72}, 10.42}, {q_{73}, 12.79}, {q_{74}, 10.42}, {q_{75}, 5.58}, {q_{76}, 2.46}, {q_{77}, 13.23}, {q_{78}, 7.82}, {q_{79}, 11.29}, {q_{80}, 11.36}, {q_{81}, 7.15}, {q_{82}, 11.53}, {q_{83}, 16.34}, {q_{84}, 10.43}, {q_{85}, 7.8}, {q_{86}, 8.96}, {q_{87}, 12.48}, {q_{88}, 22.56}, {q_{89}, 16.72}, {q_{90}, 13.93}, {q_{91}, 10.14}, {q_{92}, 10.14}, {q_{93}, 10.14}, {q_{94}, 10.89}, {q_{95}, 13.33}, {q_{96}, 5.78}, {q_{97}, 7.65}, {q_{98}, 5.78}, {q_{99}, 9.57}, {q_{100}, 13.71}, {q_{101}, 21.3}, {q_{102}, 13.34}, {q_{103}, 13.34}, {q_{104}, 14.24}, {q_{105}, 7.8}, {q_{106}, 10.28}, {q_{107}, 14.84}, {q_{108}, 7.94}, {q_{109}, 16.18}.

- **View processing MapReduce CPU time in Seconds:** {v_1, 3.79},{v_2, 4.79}, {v_3, 3.9}, {v_4, 3.54}, {v_5, 1.87}, {v_6, 5.11}, {v_7, 2.36}, {v_8, 4.38}, {v_9, 3.47}, {v_{10}, 4.48}, {v_{11}, 2.84}, {v_{12}, 1.78}, {v_{13}, 1.16}, {v_{14}, 4.79}, {v_{15}, 3.59}, {v_{16}, 3.72}, {v_{17}, 0.51}, {v_{18}, 3.68}, {v_{19}, 4.19}, {v_{20}, 3.55}, {v_{21}, 3.56}, {v_{22}, 2.18}, {v_{23}, 0.56}, {v_{24}, 1.8}, {v_{25}, 1.48}, {v_{26}, 4.01}, {v_{27}, 1.82}, {v_{28}, 3.59}, {v_{29}, 0.84}, {v_{30}, 3.5}, {v_{31}, 5.55}, {v_{32}, 0.49}, {v_{33}, 3.42}, {v_{34}, 3.96}, {v_{35}, 0.67}, {v_{36}, 1.95}, {v_{37}, 2.7}, {v_{38}, 4.01}, {v_{39}, 1.56}, {v_{40}, 3.59}, {v_{41}, 4.84}, {v_{42}, 1.65}, {v_{43}, 2.97}, {v_{44}, 2.63}, {v_{45}, 2.62}, {v_{46}, 3.24}, {v_{47}, 1.35}, {v_{48}, 2.23}, {v_{49}, 1.9}, {v_{50}, 2.39}, {v_{51}, 1.7}.

- **View maintenance MapReduce CPU time in Seconds:** {v_1, 4.7},{v_2, 5.69}, {v_3, 4.02}, {v_4, 3.7}, {v_5, 2.04}, {v_6, 5.52}, {v_7, 3.07}, {v_8, 4.48}, {v_9, 3.72}, {v_{10}, 4.71}, {v_{11}, 2.94}, {v_{12}, 2.55}, {v_{13}, 1.38}, {v_{14}, 5.15}, {v_{15}, 4.03}, {v_{16}, 3.91}, {v_{17}, 0.61}, {v_{18}, 4.32}, {v_{19}, 4.43}, {v_{20}, 4.51},

$\{v_{21}, 3.91\}$, $\{v_{22}, 2.93\}$, $\{v_{23}, 1.5\}$, $\{v_{24}, 2.31\}$, $\{v_{25}, 1.71\}$, $\{v_{26}, 4.78\}$, $\{v_{27}, 2.33\}$, $\{v_{28}, 4.27\}$, $\{v_{29}, 1.24\}$, $\{v_{30}, 3.55\}$, $\{v_{31}, 6.01\}$, $\{v_{32}, 1.08\}$, $\{v_{33}, 3.6\}$, $\{v_{34}, 4.54\}$, $\{v_{35}, 1.41\}$, $\{v_{36}, 2.3\}$, $\{v_{37}, 3.69\}$, $\{v_{38}, 4.46\}$, $\{v_{39}, 2.02\}$, $\{v_{40}, 3.82\}$, $\{v_{41}, 5.51\}$, $\{v_{42}, 2.2\}$, $\{v_{43}, 3.97\}$, $\{v_{44}, 3.47\}$, $\{v_{45}, 3.62\}$, $\{v_{46}, 3.68\}$, $\{v_{47}, 1.98\}$, $\{v_{48}, 2.78\}$, $\{v_{49}, 2.68\}$, $\{v_{50}, 2.57\}$, $\{v_{51}, 1.82\}$.

- **Size of candidate views for materializing in MBs:** $\{v_1, 4.93\}$,$\{v_2, 9.78\}$, $\{v_3, 5.85\}$, $\{v_4, 4.83\}$, $\{v_5, 2.22\}$, $\{v_6, 9.69\}$, $\{v_7, 3.55\}$, $\{v_8, 5.69\}$, $\{v_9, 5.67\}$, $\{v_{10}, 9.38\}$, $\{v_{11}, 3.96\}$, $\{v_{12}, 3.87\}$, $\{v_{13}, 2.42\}$, $\{v_{14}, 6.17\}$, $\{v_{15}, 7.03\}$, $\{v_{16}, 5.05\}$, $\{v_{17}, 0.82\}$, $\{v_{18}, 7.31\}$, $\{v_{19}, 5.19\}$, $\{v_{20}, 5.17\}$, $\{v_{21}, 7.34\}$, $\{v_{22}, 3.67\}$, $\{v_{23}, 2.71\}$, $\{v_{24}, 3.65\}$, $\{v_{25}, 1.78\}$, $\{v_{26}, 9.35\}$, $\{v_{27}, 4.16\}$, $\{v_{28}, 6.74\}$, $\{v_{29}, 2.4\}$, $\{v_{30}, 6.97\}$, $\{v_{31}, 7.73\}$, $\{v_{32}, 1.91\}$, $\{v_{33}, 6.54\}$, $\{v_{34}, 6.84\}$, $\{v_{35}, 2.19\}$, $\{v_{36}, 4.38\}$, $\{v_{37}, 4.66\}$, $\{v_{38}, 7.62\}$, $\{v_{39}, 2.36\}$, $\{v_{40}, 6.84\}$, $\{v_{41}, 9.37\}$, $\{v_{42}, 3.35\}$, $\{v_{43}, 4.81\}$, $\{v_{44}, 5.34\}$, $\{v_{45}, 7.17\}$, $\{v_{46}, 4.32\}$, $\{v_{47}, 3.37\}$, $\{v_{48}, 5.11\}$, $\{v_{49}, 4.97\}$, $\{v_{50}, 3.14\}$, $\{v_{51}, 3.24\}$.

- **Candidate view and number of queries that access them:** $\{v_1, 43\}$,$\{v_2, 17\}$, $\{v_3, 8\}$, $\{v_4, 15\}$, $\{v_5, 17\}$, $\{v_6, 8\}$, $\{v_7, 7\}$, $\{v_8, 6\}$, $\{v_9, 5\}$, $\{v_{10}, 7\}$, $\{v_{11}, 5\}$, $\{v_{12}, 25\}$, $\{v_{13}, 5\}$, $\{v_{14}, 28\}$, $\{v_{15}, 14\}$, $\{v_{16}, 7\}$, $\{v_{17}, 7\}$, $\{v_{18}, 12\}$, $\{v_{19}, 7\}$, $\{v_{20}, 4\}$, $\{v_{21}, 3\}$, $\{v_{22}, 20\}$, $\{v_{23}, 1\}$, $\{v_{24}, 16\}$, $\{v_{25}, 17\}$, $\{v_{26}, 8\}$, $\{v_{27}, 1\}$, $\{v_{28}, 5\}$, $\{v_{29}, 10\}$, $\{v_{30}, 8\}$, $\{v_{31}, 14\}$, $\{v_{32}, 13\}$, $\{v_{33}, 4\}$, $\{v_{34}, 16\}$, $\{v_{35}, 12\}$, $\{v_{36}, 6\}$, $\{v_{37}, 1\}$, $\{v_{38}, 13\}$, $\{v_{39}, 5\}$, $\{v_{40}, 12\}$, $\{v_{41}, 45\}$, $\{v_{42}, 3\}$, $\{v_{43}, 7\}$, $\{v_{44}, 6\}$, $\{v_{45}, 1\}$, $\{v_{46}, 1\}$, $\{v_{47}, 18\}$, $\{v_{48}, 1\}$, $\{v_{49}, 10\}$, $\{v_{50}, 4\}$, $\{v_{51}, 6\}$.

Data sets 2 and 3: Data sets 2 and 3 are not separately generated but extracted from data set 1. In data set 2, cost function values of 60 queries from data set 1 was considered which share the 51 views considered in data set 1. Similarly in data set 3 the cost function values of 50 queries from data set 1 was considered which share the 51 views.

Data set 4: The 20 queries and 25 associated views presented in Chapter 4 which was implemented for a recommendation system for selecting views to materialize, using real-life situation log-file containing different Map-Reduce and space costs against queries and views designed and implemented in real-life Lahman baseball database [84] is referred here as data set 4.

- **Query processing MapReduce CPU time in Seconds:** $\{q_1, 45.05\}$,$\{q_2, 37.62\}$, $\{q_3, 37.35\}$, $\{q_4, 42.59\}$, $\{q_5, 25.51\}$, $\{q_6, 24\}$, $\{q_7, 25.48\}$, $\{q_8, 38.87\}$, $\{q_9, 29.77\}$, $\{q_{10}, 18.29\}$, $\{q_{11}, 22.57\}$, $\{q_{12}, 14.15\}$, $\{q_{13}, 12.68\}$, $\{q_{14}, 29.27\}$, $\{q_{15}, 13.83\}$, $\{q_{16}, 20.56\}$, $\{q_{17}, 22.04\}$, $\{q_{18}, 21.82\}$, $\{q_{19}, 2.39\}$, $\{q_{20}, 2.31\}$.

- **View processing MapReduce CPU time in Seconds:** $\{v_1, 11.52\}$,$\{v_2, 16.34\}$, $\{v_3, 19.43\}$, $\{v_4, 14.16\}$, $\{v_5, 8.37\}$, $\{v_6, 13.85\}$, $\{v_7, 6.84\}$, $\{v_8, 11.74\}$, $\{v_9, 7.75\}$, $\{v_{10}, 15.71\}$, $\{v_{11}, 16.98\}$, $\{v_{12}, 6.02\}$, $\{v_{13}, 20.12\}$, $\{v_{14}, 11.05\}$, $\{v_{15}, 13.76\}$, $\{v_{16}, 6.61\}$, $\{v_{17}, 10.73\}$, $\{v_{18}, 10.11\}$, $\{v_{19}, 1.84\}$, $\{v_{20}, 7.28\}$, $\{v_{21}, 16.38\}$, $\{v_{22}, 8.07\}$, $\{v_{23}, 6.99\}$, $\{v_{24}, 11.58\}$, $\{v_{25}, 2.3\}$.

- **View maintenance MapReduce CPU time in Seconds:** $\{v_1, 4.02\}$,$\{v_2, 8.23\}$, $\{v_3, 4.03\}$, $\{v_4, 2.6\}$, $\{v_5, 6\}$, $\{v_6, 13.04\}$, $\{v_7, 11.5\}$, $\{v_8, 3.2\}$, $\{v_9, 6.45\}$, $\{v_{10}, 5.8\}$, $\{v_{11}, 10.03\}$, $\{v_{12}, 5.03\}$, $\{v_{13}, 4.6\}$, $\{v_{14}, 6.8\}$, $\{v_{15}, 5.4\}$, $\{v_{16}, 4.5\}$, $\{v_{17}, 2.2\}$, $\{v_{18}, 4.3\}$, $\{v_{19}, 5.4\}$, $\{v_{20}, 3\}$, $\{v_{21}, 5\}$, $\{v_{22}, 9\}$, $\{v_{23}, 7.2\}$, $\{v_{24}, 9.3\}$, $\{v_{25}, 5.8\}$.

- **Size of candidate views for materializing in MBs:** $\{v_1, 2.2\}$,$\{v_2, 2.236\}$, $\{v_3, 2.151\}$, $\{v_4, 2.2\}$, $\{v_5, 0.267\}$, $\{v_6, 2.9\}$, $\{v_7, 0.0956\}$, $\{v_8, 0.296\}$, $\{v_9, 0.001\}$, $\{v_{10}, 0.316\}$, $\{v_{11}, 0.313\}$, $\{v_{12}, 0.0252\}$, $\{v_{13}, 3.252\}$, $\{v_{14}, 0.3\}$, $\{v_{15}, 0.294\}$, $\{v_{16}, 0.026\}$, $\{v_{17}, 0.021\}$, $\{v_{18}, 0.297\}$, $\{v_{19}, 0.519\}$, $\{v_{20}, 0.061\}$, $\{v_{21}, 0.314\}$, $\{v_{22}, 0.075\}$, $\{v_{23}, 0.07\}$, $\{v_{24}, 0.299\}$, $\{v_{25}, 0.487\}$.

- **Candidate view and number of queries that access them:** $\{v_1, 2\}$,$\{v_2, 2\}$, $\{v_3, 1\}$, $\{v_4, 2\}$, $\{v_5, 1\}$, $\{v_6, 2\}$, $\{v_7, 1\}$, $\{v_8, 1\}$, $\{v_9, 2\}$, $\{v_{10}, 1\}$, $\{v_{11}, 1\}$, $\{v_{12}, 1\}$, $\{v_{13}, 1\}$, $\{v_{14}, 2\}$, $\{v_{15}, 1\}$, $\{v_{16}, 1\}$, $\{v_{17}, 1\}$, $\{v_{18}, 1\}$, $\{v_{19}, 1\}$, $\{v_{20}, 1\}$, $\{v_{21}, 1\}$, $\{v_{22}, 2\}$, $\{v_{23}, 2\}$, $\{v_{24}, 1\}$, $\{v_{25}, 2\}$.

5.5.2 Experimentation and results

The data sets presented in Section 5.5.1 was applied as input in implementations of AMOSA-MVS, Multi-Objective Differential Evolution Algorithm using Binary Encoded data (MODE-BE), and NSGA-II based system for materialized view selection. These 3 implementations are executed for several times (i.e, 20 times) using the data sets presented in Section 5.5.1 and the result sets for maximum Purity [94](see Section 5.5.3), i.e, where the maximum number of solutions remain non dominated considering the all non dominated solutions obtained by other algorithms, are considered for analysis.

5.5.2.1 Parameters used and obtained solutions

In the set of experimentation with AMOSA-MVS algorithm, the maximum size of the Archive i.e SL was fixed at 150 and when the *archive size* becomes more than 150, some of the solutions in the Archive are discarded to reduce the number of solutions to 75 as discussed in Section 5.4.2, i.e, the HL value is defined as 75. The initial value of temperature T_{max} was set as 200 and that of T_{min} used as 0.0000002. The value of α was set as 0.8. The number of iterations in every temperature in the annealing process was set as 300. For data set 1,2 and 3 the constraint on minimum size for the set of materialized views was set as 25MB. For data set 4, the constraint of minimum size of materialized views was set as 3MB.

While experimenting with MODE-BE algorithm based materialized view selection, the amplification factor F was set as 0.7 and cross-over ratio CR was set as 0.6. the population size NP used as 100 and is allowed to grow up to 200 after which the population size is controlled or reduced by preserving diversity in solution space as discussed in Chapter 4. That is, the value of Γ was set as 2. Here for 100 number of generations the process was set to run. In case of NSGA-II

based view selection, the CR value was set as 0.5 for 100 generations with initial 100 solution population in the archive which was allowed to increase maximum up to double of this size in intermediate generations and after which the population was controlled as suggested in [32].

The objective function values obtained by the yielded solutions that are considered for performance analysis are presented in tables 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12 and 5.13.

5.5.3 Comparison measures

In case of multi-objective optimization by randomized algorithms, the performance of algorithms may be analyzed in different ways. The main measures used for performance analysis of non-deterministic multi-objective optimization techniques are (1) the measure on how the obtained solutions converge towards the true Pareto front, (2) the number or fraction of obtained solutions that are non dominating with respect to the known Pareto front and (3) the distribution of solutions in the Pareto front or in the solution space. In case of selection of materialized views, the goal is to find sets of views as solutions such that these solution sets converge to the true sets of solutions that minimizes query processing costs, materialized view maintenance costs and number of materialized views with respect to size of the views. Thus the obtained solutions should be nearest to the true solutions and they should be well represented from the actual complete set of non-dominated solutions. By a single measuring parameter these can not be measured. Therefore two types of measures have been used to evaluate the quality of obtained solutions. One for measuring the fraction of solutions that remain non-dominated with respect to all solutions by other algorithms i.e the Purity [94] of solutions. And the second type of measure for measuring the extent of Convergence [32] of the solution set to an already known set of Pareto optimal solutions with uniformity of the Spacing between the solutions over the non dominated front [32, 89, 94]. These measures are defined below.

Purity: The Purity measure is used to compare the solutions obtained by different multi-objective optimization techniques by calculating the fraction of solutions from one particular technique that remains non-dominating by considering the all non-dominated solutions obtained by all other techniques that are considered for comparing [94]. The Purity value near 1 indicates better performance and the value near 0 means poorer performance. If the solutions obtained by an algorithm yields Purity value 1, then the the algorithm may be considered as the fittest for the application, because all the solutions it has produced are not dominated by solutions produced by any other algorithm so far.

Convergence measure γ: The Convergence measure denoted by γ measures the extent of convergence to a known set of Pareto-optimal solutions. For computing Convergence as suggested in [32], first the set of non dominating solutions obtained by already used algorithms for the application considered, H, are found and then for each solution obtained with an algorithm, the minimum Euclidean distance of it from chosen solutions of H on the Pareto-optimal front are computed and the average of these minimum distances is used as the Convergence measure γ. The

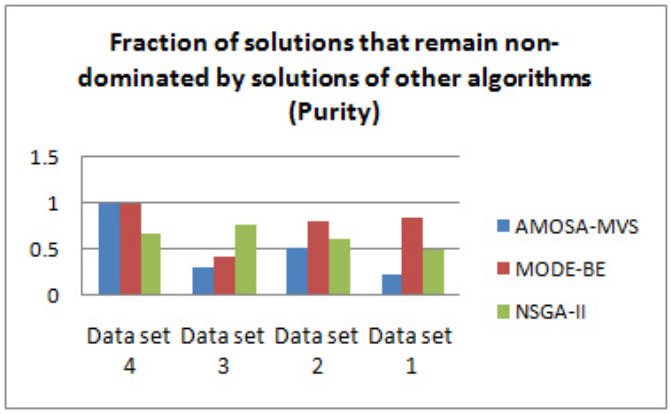

Figure 5-7: Purity

lower the value γ, better is the convergence of the solution set obtained to the true Pareto optimal front.

Spacing and Minimal Spacing: Other than Convergence and Purity measure, the algorithms are also analyzed for how the solutions are distributed over the known true Pareto front. The multi-objective optimization techniques are basically aimed at getting a set of solutions that spans the entire Pareto-optimal region. For measuring the span of solutions, Schott [89] proposed the measure of Spacing, S, to reflect the uniformity of the solutions over a non-dominated front. The *spaceing*, S is computed as expressed below.

$$S = \sqrt{\frac{1}{|Q|}\sum_{i=1}^{|Q|}(d_i - d)^2} \tag{5.14}$$

where $d_i = min_{k \in Q \text{ and } k \neq i} \sum_{m=1}^{M} |f_m^i - f_m^k|$ and f_m^i (or f_m^k) is the mth objective value of the ith (or kth) solution in the final non-dominated solution set Q , d is the mean value of all d_is. A value of S near 0 indicates that the solutions are uniformly distributed over the Pareto optimal front. But in cases where the complete true Pareto front is not known or only a segment of the front is considered for computing S, this measure may be unable to indicate the actual spaces in between the solutions in the front. Therefore in [94] a modified measure named *Minimal Spacing*, S_m is proposed, where $|Q|$ is replaced by $|Q| - 1$ as actually $|Q| - 1$ number of distances are considered for measuring. Again there may be diverse objective function values. Therefore, the term $|f_m^i - f_m^k|$ is divided by $|F_m^{max} - F_m^{min}|$ to normalize the objective function values, where F_m^{max} and F_m^{min} are the maximum and minimum objective function values respectively of mth objective. Bigger value of S_m reflects that solutions are not uniformly distributed over the known Pareto-optimal front. If uniformly distributed larger number of solutions are desired then smaller value of S or S_m indicates better performance. But if fewer number widely spread solutions in objective function space are to be extracted, then bigger value of S or S_m is better.

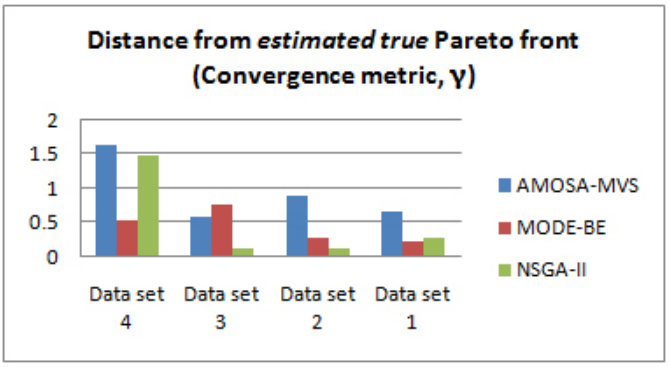

Figure 5-8: Convergence metric (γ)

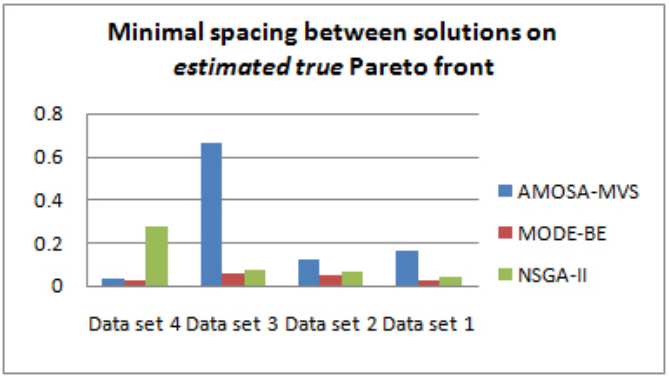

Figure 5-9: Minimal spacing between solutions on estimated true Pareto front

Table 5.6: MODE-BE generated solutions' objective function values considering 109 number of queries and 51 views. Number of initial solutions = 2487.

Sol. Sl.No.	Query processing cost (in Seconds)	View maintenance cost (in Seconds)	Number of views selected	Total view size (in MB)	[D]ominated/ [N]on-dominated solutions by other algorithms.
1.	124.18	129.14	37	197.29	N
2.	157.17	128.42	36	198.11	N
3.	183.88	125.4	34	190.34	N
4.	184.42	122.69	35	189.55	N
5.	187.08	119.02	31	183.86	N
6.	198.37	115.19	32	174.03	N
7.	210.8	116.56	31	180.42	N
8.	218.33	115.71	31	178.79	N
9.	221.56	113.33	33	177.55	N
10.	222.01	115.74	30	176.14	N
11.	222.38	114.53	31	174.12	N
12.	228.63	112.56	31	171.76	N
13.	231.92	110.89	33	166.37	N
14.	236.77	112.18	30	171.57	N
15.	241.26	109.98	30	165.6	N

Continued on next page

Table 5.6 – continued from previous page

Sol. Sl.No.	Query processing cost (in Seconds)	View maintenance cost (in Seconds)	Number of views selected	Total view size (in MB)	[D]ominated/ [N]on-dominated solutions by other algorithms.
16.	249.65	109.66	31	170.59	N
17.	249.85	109.11	30	166.57	N
18.	249.9	108.84	32	165.52	N
19.	257.02	109.35	29	164.1	N
20.	261.99	108.45	29	162.42	N
21.	263.33	107.8	32	163.85	N
22.	263.45	105.26	28	161.17	N
23.	266.97	103.92	29	154.94	N
24.	270.28	105.19	28	162.12	N
25.	274.26	105.65	27	159.88	N
26.	278.1	103.35	29	155.57	N
27.	281.55	102.64	27	156.07	N
28.	292.13	100.61	27	152.91	N
29.	297.25	100.22	28	153.51	N
30.	302.08	101.52	26	152.26	N
31.	316.87	98.97	28	152.79	N
32.	318.36	94.96	27	145.31	N
33.	323.61	95.23	26	145.26	N
34.	326.78	98.42	24	152.73	N
35.	333.24	91.53	26	139.08	N
36.	346.76	91.44	24	136.66	N
37.	353.44	89.87	24	135.95	N
38.	356.48	90.74	23	134.15	N
39.	369.15	87.63	26	133.5	N
40.	377.52	87.56	24	130.95	N
41.	396.8	88.14	23	139.75	N
42.	397.96	85.1	24	127.15	N
43.	401.15	90.47	22	137.36	N
44.	428.66	83.45	25	122.88	N
45.	431.08	86.95	22	132.9	N
46.	452.92	82.09	23	121.62	D
47.	456.55	84.32	22	133.2	D
48.	457.13	80.24	21	124.78	N
49.	472.32	79.54	22	122.16	N
50.	474.28	79.11	22	121.92	N
51.	476.61	76.94	23	114.38	N
52.	503.05	76.42	21	117.27	N
53.	504.1	82.68	20	125.49	N
54.	516.88	75.84	20	111.89	N
55.	524.4	75.13	20	113.64	N
56.	531.57	72.93	18	107.14	N

Continued on next page

Table 5.6 – continued from previous page

Sol. Sl.No.	Query processing cost (in Seconds)	View maintenance cost (in Seconds)	Number of views selected	Total view size (in MB)	[D]ominated/ [N]on-dominated solutions by other algorithms.
57.	536.57	72.44	19	108.49	N
58.	559.96	66.24	19	98.33	N
59.	607.53	69	18	107.94	N
60.	651.5	64.4	19	94.18	N
61.	658.18	66.41	16	101.53	D
62.	685.91	61.46	17	94.89	D
63.	687.89	66.02	16	100.6	D
64.	692.07	60.16	16	88.21	D
65.	726.05	57.54	14	84.09	N
66.	818.85	54.89	18	82.55	D
67.	820.6	52.69	16	78.33	D
68.	937.27	50.62	16	77.36	D
69.	973.71	47.87	16	68.83	D
70.	1002.54	45.97	15	70.61	D
71.	1019.05	42.71	13	60.75	N

Table 5.10: NSGA-II generated solutions' objective function values considering 109 number of queries and 51 views. Number of initial solutions = 5015

Sol. Sl.No.	Query processing cost (in Seconds)	View maintenance cost (in Seconds)	Number of views selected	Total view size (in MB)	[D]ominated/ [N]on-dominated solutions by other algorithms.
1.	1381.99	19.7	7	27.97	N
2.	1368.75	32.87	10	47.94	N
3.	1210.3	33.7	11	53.5	N
4.	1192.64	39.48	12	62.24	D
5.	1150.16	42.49	12	63.57	N
6.	1084.72	37.52	13	58.24	N
7.	1069.11	40.09	13	57.43	N
8.	1065.4	42.12	13	65.62	N
9.	1062.05	42.62	13	66.32	N
10.	1052.05	42.56	15	62.7	N
11.	1012.29	44.57	13	62.63	N
12.	976.22	42.95	14	63	N
13.	930.36	44.91	13	70.37	N
14.	887.53	47.94	15	71.02	N
15.	799.51	49.87	15	70.94	N
16.	764.3	50.62	13	76.96	N
17.	754.63	54.65	16	75.65	N
18.	754.48	58.22	17	84.04	D

Continued on next page

Table 5.10 – continued from previous page

Sol. Sl.No.	Query processing cost (in Seconds)	View maintenance cost (in Seconds)	Number of views selected	Total view size (in MB)	[D]ominated/ [N]on-dominated solutions by other algorithms.
19.	747.36	56	18	83.64	N
20.	717.48	58.53	15	90.62	N
21.	665.06	60.95	15	90.41	N
22.	652.81	59.36	16	86.89	N
23.	652.53	62.82	18	92.73	N
24.	637.06	64.5	16	97.38	N
25.	602.46	65.13	18	95.32	N
26.	589.43	69.81	18	105.91	N
27.	581.57	66.74	19	99.33	D
28.	581.47	68.12	19	98.77	D
29.	574.57	73.17	19	110.35	D
30.	573.82	74.17	21	110.29	D
31.	551.6	76.48	20	116.26	D
32.	546.57	75.64	22	112.31	D
33.	487.46	76.62	21	113.72	N
34.	451.3	81.34	22	121.95	N
35.	433.36	83.4	23	124.17	N
36.	409.57	90.34	25	140.37	D
37.	408.61	91.93	24	142.79	D
38.	407.1	91.34	26	138.07	D
39.	399.05	93.69	24	141.87	D
40.	394.19	92.21	25	139.69	D
41.	387.94	92.81	25	145.43	D
42.	386.78	94.15	25	146.02	D
43.	383.79	95.01	25	148.79	D
44.	378.85	95.41	25	148.53	D
45.	374.03	94.56	26	140.62	D
46.	369.82	93.72	28	146.09	D
47.	365.19	95.41	26	147.02	D
48.	361.56	99.62	27	153.32	D
49.	334.83	100.82	27	156.76	D
50.	303.06	103.27	28	161.23	D
51.	296.74	112.6	30	175.8	D
52.	280.38	109.85	31	168.71	D
53.	263.57	118.31	33	176.32	D
54.	237.34	121.23	35	187.5	D
55.	184.3	138.94	40	209.53	D

Table 5.2: AMOSA-MVS generated solutions' objective function values considering 109 number of queries and 51 views. Number of initial solutions considered= 3975.

Sol. Sl.No.	Query proc. cost (in Seconds)	View maint. cost (in Seconds)	Number of views selected	Total view size (in MB)	[D]ominated/ [N]on-dominated solutions by other algorithms.
1.	661.84	100.6	32	148.64	D
2.	680.5	84.13	27	128.54	D
3.	751.26	83.52	27	129.21	D
4.	754.83	82.91	26	128.39	D
5.	814.62	78.98	26	122.37	D
6.	898.3	78.62	26	119.17	D
7.	972.63	77.4	25	119.02	D
8.	973.35	77.33	25	119.05	D
9.	1004.71	72.62	24	109.67	D
10.	1033.51	70.31	23	106.02	D
11.	1159.04	36.01	10	52.27	N
12.	1226.68	35.73	10	54.13	N
13.	1267.19	32.56	10	54.13	N
14.	1442.95	29.11	10	46.72	D

5.5.4 Comparative analysis

The Purity, Convergence and Minimal Spacing values obtained by AMOSA-MVS, MODE-BE and NSGA-II algorithms in materialized view selection for data warehousing in our experimentation using the data sets in Section 5.5.1 above are presented in Tables 5.14, 5.1 and 5.15 respectively. Though all the algorithms show acceptable values of performance measures, MODE-BE results are found to be consistently well performing among these three techniques in our experimental setup. In case of materialized view selection problem, for large dimensional problem, i.e, with large number of queries and views it is not possible to find the true Pareto front beforehand. In our comparison metrics, only the segment of the front that has been obtained by the considered algorithms have been used. MODE-BE and NSGA-II both are evolutionary algorithms where cross-over and mutations among solutions are done for generating new candidate solution. Whereas in case of AMOSA-MVS new candidate solutions are generated by perturbing one or more dimensions of one solution vector at a time during large number of iterations in the annealing process. Therefore, randomized function based generation of solutions by perturbing values of dimensions of solution vectors from randomly selected solution vector as in case of AMOSA-MVS may produce distant solutions in objective function space.

In this set of experimentation, though in higher dimensional test data it has been observed that MODE-BE and NSGA-II generated solutions are of

Table 5.3: AMOSA-MVS generated solutions' objective function values considering 60 number of queries and 51 views. Number of initial solutions considered= 3975.

Sol. Sl.No.	Query processing cost (in Seconds)	View maintenance cost (in Seconds)	Number of views selected	Total view size (in MB)	[D]ominated/ [N]on-dominated solutions by other algorithms.
1.	471.93	56.18	18	84.31	D
2.	771.74	36.57	11	53.04	D
3.	702.44	36.39	11	52.49	D
4.	816.32	29.95	8	41.2	N
5.	863.62	28.99	8	43.23	N
6.	925.63	25.84	8	38.14	N

Table 5.4: AMOSA-MVS generated solutions' objective function values considering 50 number of queries and 51 views. Number of initial solutions considered= 3975.

Sol. Sl.No.	Query processing cost (in Seconds)	View maintenance cost (in Seconds)	Number of views selected	Total view size (in MB)	[D]ominated/ [N]on-dominated solutions by other algorithms.
1.	93.96	104.54	33	152.42	D
2.	717.62	21.58	6	32.05	N
3.	162.2	99.01	31	143.8	D

higher Purity, AMOSA-MVS produced highest Purity value in smaller dimensional problem (as in Table 5.14). In Table 5.1, the Convergence measures are presented. Here it is observed that AMOSA-MVS produced acceptable convergence measure γ despite measuring it with respect to large number of non-dominated solutions obtained from MODE-BE and NSGA-II. Overall it is observed that MODE-BE algorithm converges very well empirically in this application.

From Table 5.15, it is observed that in higher dimensional cases, AMOSA-MVS based solutions are of largest value of minimal spacing indicating least uniformity in their distribution in the Pareto front. But in case of one data set, the AMOSA-MVS and MODE-BE generated solutions' minimal spacing values are less than that of NSGA-II generated solutions. In most cases, as MODE-BE and NSGA-II yield comparatively larger number of solutions in the Pareto front than that of AMOSA-MVS (as seen in Tables 5.2 to 5.13), there is possibility of getting smaller minimal spacing value. By looking at these comparison metrics, it has been observed that AMOSA-MVS with its less computational complexity (as discussed in Section 5.4.5), yields comparable quality of solutions with respect to MODE-BE and NSGA-II for selecting views to materialize in data warehouses which use Big data framework with Distributed File System architecture.

Table 5.5: AMOSA-MVS generated solutions' objective function values considering 20 number of queries and 25 views. Number of initial solutions considered= 4000.

Sol. Sl.No.	Query processing cost (in Seconds)	View maintenance cost (in Seconds)	Number of views selected	Total view size (in MB)	[D]ominated/ [N]on-dominated solutions by other algorithms.
1.	323.11	56.17	11	8.98	N
2.	331.91	53.82	10	9.47	N
3.	338.61	49.72	10	8.98	N
4.	343.65	50.62	9	9.17	N
5.	348.29	48.82	9	9.15	N
6.	352.59	42.52	9	8.91	N
7.	359.87	39.52	8	8.85	N

5.6 Discussion

In this endeavor, AMOSA algorithm has been applied for materialized view selection problem considering Data warehouse query performance, built on Big data or Big-table based Distributed File System Architecture termed as Hadoop Distributed File System frame work. The fundamental AMOSA algorithm designed by Bandyopadhyay et al. in [30] has been adapted with modification for controlling solution population by maintaining diversity in solution space using distance based measure for filtering solutions during annealing process unlike the Single-linkage clustering used in the original AMOSA algorithm. This version of AMOSA for materialized view selection problem is termed as AMOSA-MVS.

By sorting the solution vectors on minimum distances of each solution vectors to all other solutions in the Archive in solution space and discarding solutions with smaller minimum distances for maintaining diversity instead of using clustering based technique as used in original AMOSA algorithm, the complexity of AMOSA-MVS algorithm is remarkably reduced and yet the solutions yielded by this algorithm found to be of comparable quality with respect to other similar randomized algorithms with higher computational complexities. In case of higher dimensional problem, it is not possible to find the true Pareto front beforehand. Therefore, when randomized algorithms are used for such problems, at different instances, an algorithm may produce different sets of solutions and for that a different comparison metrics of Purity, Convergence and Minimal spacing may be found. In our experimentation randomly maximum 109 queries with 51 sub-queries, that may be converted to views for materializing, are used with their MapReduce CPU time cost for experimenting with higher dimensional data. Another real-life data set of 20 HiveQL queries with 25 associated views, implemented for a real-life situation data warehousing for generating cost function values is used in this Multi-Objective Simulated Annealing Algorithm based recommendation system.

Table 5.7: MODE-BE generated solutions' objective function values considering 60 number of queries and 51 views. Number of initial solutions considered= 2545.

Sol. Sl.No.	Query processing cost (in Seconds)	View maintenance cost (in Seconds)	Number of views selected	Total view size (in MB)	[D]ominated/ [N]on-dominated solutions by other algorithms.
1.	6.12	73.78	19	109.8	N
2.	29.21	73.89	18	110.36	N
3.	34.8	69.36	18	102.62	N
4.	37.51	68.16	21	100.97	N
5.	39.75	71.84	17	109.17	N
6.	48.51	69.11	18	104.28	N
7.	48.8	64.94	19	97.41	N
8.	52.02	65.52	17	97.78	N
9.	100.02	64.75	19	96.47	N
10.	104.22	62.86	17	93.44	N
11.	119.62	58.92	17	89.04	N
12.	165.7	56.57	17	84.76	N
13.	191.88	60	16	90.35	N
14.	208.3	59.39	16	92.99	D
15.	220.11	52.44	15	76.05	N
16.	297.03	49.99	16	77	N
17.	319.98	48.83	16	75.52	N
18.	387.58	46.96	14	69.9	D
19.	483.87	41.67	15	62.06	D
20.	614.9	40.43	12	64.37	D

In [30], Bandyopadhyay et al. mention that the main time consuming procedure in basic AMOSA algorithm is the clustering part. Therefore, in AMOSA-MVS algorithm, this clustering part is replaced with a simpler method. The AMOSA-MVS, MODE-BE and NSGA-II algorithm with test data generated by processing queries in a stand-alone version of testing platform with Hadoop and Hive system have been used for implementing multi-objective optimization technique in selecting views to materialize. Though by randomly generated experimental data, the actual convergence properties and quality of solutions can not be established, yet by looking at the performance comparison measures with respect to that of already established test problems and algorithms, AMOSA-MVS is found to be acceptable for selecting views to materialize in data warehouses.

Table 5.8: MODE-BE generated solutions' objective function values considering 50 number of queries and 51 views. Number of initial solutions considered=2520

Sol. Sl.No.	Query processing cost (in Seconds)	View maintenance cost (in Seconds)	Number of views selected	Total view size (in MB)	[D]ominated/ [N]on-dominated solutions by other algorithms.
1.	13.71	57.04	16	86.32	N
2.	90.06	53.42	15	76	N
3.	159.49	49.74	15	73.42	D
4.	174.71	53.5	14	83.07	D
5.	268.38	47.56	15	70.4	D

Table 5.9: MODE-BE generated solutions' objective function values considering 20 number of queries and 25 views. Number of initial solutions considered=2899

Sol. Sl.No.	Query processing cost (in Seconds)	View maintenance cost (in Seconds)	Number of views selected	Total view size (in MB)	[D]ominated/ [N]on-dominated solutions by other algorithms.
1.	371.95	21.08	5	8.81	N
2.	382.68	18.88	4	8.79	N
3.	388.25	17.85	5	6.89	N
4.	391.38	17.05	4	6.66	N
5.	398.98	15.65	4	6.87	N
6.	402.11	14.85	3	6.64	N
7.	404.63	12.85	4	6.57	N
8.	414.42	13.03	3	4.46	N
9.	415.36	10.65	3	6.55	N
10.	424.06	8.82	3	4.42	N
11.	425.15	10.83	2	4.44	N
12.	434.79	6.62	2	4.4	N

Table 5.11: NSGA-II generated solutions' objective function values considering 60 number of queries and 51 views. Number of initial solutions considered= 5040.

Sol. Sl.No.	Query processing cost (in Seconds)	View maintenance cost (in Seconds)	Number of views selected	Total view size (in MB)	[D]ominated/ [N]on-dominated solutions by other algorithms.
1.	686	30.7	10	47.89	N
2.	584.45	43.09	10	62.83	N
3.	523.03	35.4	11	52.82	N
4.	495.22	39.29	12	60.01	N
5.	479.38	39.89	13	61.19	N
6.	420	41.92	12	58.93	N
7.	413.07	43.75	12	63.49	N
8.	378.45	45.36	12	69.1	N
9.	369.85	42.58	14	60.19	N
10.	339.75	45.94	16	65.98	N
11.	329.02	46.77	13	67.02	N
12.	322.63	48.63	14	74.45	N
13.	308.72	48.87	16	71.99	N
14.	297.78	51.35	14	71.94	N
15.	286.02	55.85	14	79.29	N
16.	280.79	56.76	14	86.24	N
17.	266.41	57.02	13	84.58	N
18.	239.17	50.57	15	72.39	N
19.	238.87	54.45	17	81.11	D
20.	238.63	56.36	15	78.11	D
21.	237.75	56.53	18	83.6	D
22.	233.13	56.75	17	84.44	D
23.	193.99	57.67	15	89.47	N
24.	193.7	60.09	19	86.9	D
25.	174.62	60.39	16	87.25	N
26.	158.69	61.84	16	93.95	N
27.	143.28	68.07	17	102.38	D
28.	133.9	62.73	18	86.48	D
29.	131.95	72.27	17	108.78	D
30.	131.47	60.86	19	91.46	N
31.	117.43	69.84	18	99.18	D
32.	108.25	65.67	19	96.8	D
33.	102.41	66.87	19	101.17	D
34.	99.9	69.36	21	101.77	D
35.	98.1	70.89	19	99.64	D
36.	89.86	70.21	21	102.73	D
37.	55.69	75.8	18	112.73	D
38.	37.27	71.56	20	103.01	D
39.	34.02	72.86	19	103.75	N
40.	2.09	73.15	21	106.13	N

Table 5.12: NSGA-II generated solutions' objective function values considering 50 number of queries and 51 views. Number of initial solutions considered= 5163.

Sol. Sl.No.	Query processing cost (in Seconds)	View maintenance cost (in Seconds)	Number of views selected	Total view size (in MB)	[D]ominated/ [N]on-dominated solutions by other algorithms.
1.	461.65	34.04	10	52.1	N
2.	339.74	37.03	11	53.47	N
3.	332.99	41.71	12	63.57	N
4.	301.09	42.52	12	64.59	N
5.	267.7	43.24	12	67.44	N
6.	260.28	44.74	13	69.56	N
7.	254.56	45.56	13	68.09	N
8.	178.62	46.11	12	68.37	N
9.	134.05	47.84	14	71.77	N
10.	127.65	52.36	15	78.04	N
11.	114.93	54.65	14	86.42	N
12.	99.36	56.87	14	86.36	N
13.	59.13	55.53	15	79.87	N
14.	58.02	54.28	16	83.25	N
15.	56.82	59.29	17	83.77	D
16.	46.69	59.69	17	88.96	D
17.	40.15	60.16	17	92.36	D
18.	28.94	60.36	16	88.54	D
19.	14.48	61.28	18	91.83	D
20.	4.54	63.18	17	95.69	N
21.	2.56	64.34	17	97.86	N

Table 5.13: NSGA-II generated solutions' objective function values considering 20 number of queries and 25 views. Number of initial solutions considered=4888

Sol. Sl.No.	Query processing cost (in Seconds)	View maintenance cost (in Seconds)	Number of views selected	Total view size (in MB)	[D]ominated/ [N]on-dominated solutions by other algorithms.
1.	231.74	92.68	14	14.91	N
2.	197.1	122.43	20	16.16	N
3.	454.45	13.9	2	3.55	D

Table 5.14: Purity

Number of queries, shared views	AMOSA-MVS	MODE-BE	NSGA-II
109, 51	0.2143	0.8451	0.4909
60, 51	0.5	0.8	0.6
50, 51	0.3	0.4	0.7619
20, 25	1	1	0.6667

Table 5.15: Minimal Spacing

Number of queries, shared views	AMOSA-MVS	MODE-BE	NSGA-II
109, 51	0.1655	0.0228	0.0410
60, 51	0.1247	0.0497	0.0630
50, 51	0.6665	0.0592	0.0737
20, 25	0.0296	0.0211	0.2782

Chapter 6

Conclusion and Future Direction

This dissertation makes four important contributions to the body of knowledge on materializing views in data warehouses by selecting optimum set of views with respect to over all query efficiency, materialized view maintenance and memory space constraint of data warehouses. In this chapter, we summarize the main contributions made in this dissertation and provide directions for future works.

6.1 Conclusions

Following conclusions are drawn from the contributions in this dissertation.

- In Chapter 2, a comprehensive report on the approaches introduced to select views for materializing in data warehouses have been presented with associated issues and challenges that have been identified in the study. It has been observed that the scalability due to exponential explosion of solution space with dimension of data warehouses is a big issue with deterministic and heuristic algorithm based solution search methods applied in view selection. Evolutionary and stochastic methods like Genetic Algorithm (GA) and Simulated Annealing (SA) algorithms search solutions in a multi-dimensional fashion and can provide effective search performance in finding an optimum set of views for materializing near the global optimum. But in these approaches, the solution quality depends on different parameters and values that are specified. Soft-computing approaches in the view selection problem use clustering and associative rule mining on a (frequent) query versus view matrix. The quality of the quasi-optimum solutions discovered by these techniques depends on the pre-defined clustering parameters. Analysis of large number of complex queries for finding frequent significant sub-queries, aggregation functions and views that may be defined as candidate solution set of views is also a big issue. Defining generalized cost function representation of this optimization problem is another issue. In most of the existing approaches in materialized view selection, all the associated costs that are

to be minimized are summed up as a single cost for minimizing, ignoring the trade-offs between them. The existing models define views as some derived functions or relations on some normalized relational model based tables or relations. These models do not support semi-structured or un-structured databases with very little indexing capabilities as used in Big data framework based data warehousing.

- When an optimization problem with multiple non-dominating objectives is converted into single objective, it ignores that different solutions may offer trade-offs between the objectives. In Chapter 3, 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. Multi-objective Differential Evolution (MODE) algorithm has been patched up for binary encoded solution representation of the problem for utilizing conventional multiple view processing plan as input. NSGA-II also has been applied with equivalent parameters in this problem and it has been observed that the solutions yielded by both NSGA-II and multi-objective DE algorithms are distributed in similar curve in the objective function space. But it has been observed that the solution quality of solutions obtained by this approach in view selection for materialization are somewhat better than that of NSGA-II with respect to convergence property and total cost function values.

- In Big data framework, frequent sub-queries or views may be materialized for speeding up MapReduce computing paradigm based query processing. Materializing frequent sub-queries and views means that the views reside in the memory of one or more nodes in the cluster of commodity hardware save MapReduce costs by reducing repetitions of submission and scheduling cost of Distributed File System jobs for query processing. In Chapter 4, materialized views are defined as resultant data of frequent sub-queries and aggregation functions of a set of Big data warehousing queries. The problem is defined as a multi-objective optimization problem for minimizing the total query processing MapReduce cost, MapReduce cost for maintaining the materialized views and the number of views selected for materializing with maximized total size of the views selected while selecting views for materializing. The patched-up Multi-Objective DE used in Chapter 3 is modified and applied here. The NSGA-II has been implemented to study comparative performances for developing a recommendation system for selecting views for materializing in Big data warehousing. It is observed that the diversity of solutions generated by MODE-BE in solution space is more than that of NSGA-II generated solutions. The diversity in solution vector space is preferred because diversity preservation on objective function values may lead to loss of some significantly distinct solutions on the basis of constituent selected views in them. In our experimentation it is observed that MODE-BE generates 37.04% more number of solutions than NSGA-II based system. More number of solutions may be useful for selecting most appropriate solutions. But by applying Mann-Whitney U test on both MODE-BE and NSGA-II generated solutions at 5% level of significance, it cannot be

rejected the null hypothesis that the solution vectors generated by both the systems are from the same population.

- Finally Chapter 5 discusses how multi-objective Simulated Annealing based techniques may be applied in selecting sub-query results or views in MapReduce based query processing framework for materializing. A comparative performance analysis of this technique and common EA based techniques in view selection problem in this paradigm is presented. The original AMOSA algorithm of total run time complexity $O((Total\ iterations) \times (N \times logN))$ has been customized for materialized view selection application with overall run time complexity of $O((Total\ iterations) \times (N + logN))$. The customized algorithm produced acceptable convergence measure γ despite measuring it with respect to large number of non-dominated solutions obtained from MODE-BE and NSGA-II applied in this problem. But overall it has been observed that the MODE-BE algorithm converges the best empirically in this application. In this version of AMOSA, termed as AMOSA-MVS, while maintaining diversity in solution space in intermediate generations, distance based measure is used for filtering solutions in stead of the Single-linkage clustering used in the original version. Solutions yielded by this technique is found to be of comparable quality with respect to other similar randomized algorithms despite its much lower computational complexity.

6.2 Future Directions

Few of the possible directions for future research in this area may be outlined as below.

- 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 [76] 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. The future focus should be on developing an analytical model for big and complex view processing environment.

- Presently the view materialization problem has been defined from the perspective of homogeneous data warehousing system where the data warehousing is considered either as a conventional RDBMS based data warehouse or a Big data system. But with the advent new computing technologies like mobile computing to Big data distributed computing at heterogeneous cluster of commodity hardware, the query execution performance improvement by view materialization will involve optimization of many other parameters

with large number of different types of resource constraints. Therefore, for optimized materialization of views in this scenario, objective functions are to be defined for distributed data warehouse, spread over heterogeneous data nodes with heterogeneous data organization, resources and constraints.

- Extraction of candidate views from query workload on data warehouses is presently done by offline syntactic analysis of the query execution log files in the system. The basic assumption in most of the works on this problem is that the frequent analytical queries and the intermediate views generated on a data warehouse in a specific period of time will reoccur as frequent queries on the data warehouse in future. But, in present business scenario of complex strategic decision making process in very highly competitive business world, it is not always true. Therefore, integrated view selection and materialization system with on-line analysis of triggered queries is to be designed for dynamic and incremental updating of pool of candidate views for materializing.

- Theme based partitioning of analytical processing on data warehouse for enhanced query performance is another potential area of research. Works are to be done for analysis of queries for finding different themes of analytical processing. Data mining techniques specifically frequent item-set mining techniques may be useful for partitioning or dynamic partitioning of the candidate views for materializing on different identified themes. Finding candidate views and finally finding the solution set of views for materializing by user specified criteria or theme and configuration is another research direction.

- The existing data warehousing system in Apache Hadoop Distributed File System (HDFS) converts user's SQL (i.e, HiveQL) query statement into *abstract syntax tree* (AST) which is then converted to physical operator tree for execution by syntactic analysis. Query optimization is done on this physical operator tree. On the other hand query processing plans are generated by semantic information obtained from semantic analysis of queries. Down stream query performance optimizations like in case of selecting optimum set of views for materializing is mainly done by looking at these semantic information based query execution plans. Due to difference in semantic information based query execution plan and physical query tree, the query performance optimization may not be achieved at times. This is another major issue to be addressed in this area of research.

Bibliography

[1] Inmon, W. H. *Building the Data Warehouse*, John Wiley and Sons, Inc., New York, NY, USA, 2005, 4 edn.

[2] Han, J. & Kamber, M. *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, An imprint of Elsevier, 500 Sansome Street, Suite 400, San Francisco, CA 94111, 2006, 2 edn.

[3] Gupta, A. & Mumick, I. Maintenance of materialized views: Problems, techniques, and applications. *IEEE Data Engineering Bulletin* **18** (2), 3–18, 1995.

[4] Harinarayan, V. *et al.* Implementing data cubes efficiently. In *Proceedings of ACM SIGMOD International Conference on Management of Data*. 205–216, ACM SIGMOD, Montreal, Canada, 1996.

[5] Hobbs, L. *Oracle Materialized Views and Query Rewrite*, Oracle Corporation, World Headquarters, 500 Oracle Parkway, Redwood Shores, CA 94065, U.S.A, 2005, 1 edn.

[6] Nadeua, T. & Teorey, T. Achieving scalability in olap materialized view selection. In *Proceedings of ACM Fifth International Workshop on Data Warehousing and OLAP, DOLAP-02*. 28–34, ACM, McLean, Virginia, USA, 2002.

[7] Yang, J. *et al.* Algorithm for materialized view design in data warehousing environment. In *Proceedings of VLDB 1997*. 136–145, Athens, Greece, 1997.

[8] Derakhshan, R. *et al.* Simulated annealing for materialized view selection in data warehousing environment. In *Proceedings of 24th IASTED international conference on Database and applications*. 89–94, Innsbruck, Austria, 2006.

[9] Derakhshan, R. *et al.* Parallel simulated annealing for materialized view selection in data warehousing environments. In *Proceedings of ICA3PP 2008 International Conference on Algorithms and Architecture 2008*, vol. 5022 of *LNCS*. 121–132, Springer-Verlag, Berlin, Heidelberg, Cyprus, 2008.

[10] Aouiche, K. *et al.* Clustering-based materialized view selection in data warehouses. In Manolopoulos, Y. *et al.* (eds.) *Proceeding of Tenth East-European Conference Advances in Database and Information Systems, ADBIS 2006*, vol. 4152 of *LNCS*. 81–95, Springer-Verlag, Berlin, Heidelberg, Thessaloniki, Hellas, 2006.

[11] Aouiche, K. & Darmont, J. Data mining-based materialized view and index selection in data warehouses. *Journal of Intelligent Information System* **33**, 65–93, 2009.

[12] Das, A. & Bhattacharyya, D. K. Density-based view materialization. In Pal, S. K. *et al.* (eds.) *Proceedings of First International Conference on Pattern Recognition and Machine Intelligence, PReMI 2005*, vol. 3776 of *LNCS*. 589–594, Springer-Verlag, Berlin, Heidelberg, ISI, Kolkata, India, 2005.

[13] Kumar, T. V. *et al.* Mining queries for constructing materialized views in a data warehouse. In Wyld, D. C. *et al.* (eds.) *Advances in Computer Science, Engineering and Application, Proceedings of the Second International Conference on Computer Science, Engineering and Applications (ICCSEA 2012), Volume 2*, vol. 167 of *Advances in Intelligent and Soft Computing*. 149–159, Springer, Berlin Heidelberg, New Delhi, India, 2012.

[14] Goswami, R. *et al.* Selection of views for materializing in data warehouse using mosa and amosa. In Wyld, D. C. *et al.* (eds.) *Advances in Computer Science, Engineering and Applications, Proceedings of the Second International Conference on Computer Science, Engineering and Applications (ICCSEA 2012), Volume 1*, vol. 166 of *Advances in Intelligent and Soft Computing*. 619–628, Springer, Berlin Heidelberg, New Delhi, India, 2012.

[15] Tpc-h benchmark specification, 2008. URL http://www.tpc.org/hspec.html.

[16] Dean, J. & Ghemawat, S. Mapreduce: simplified data processing on large clusters. *Communications of the ACM* **51** (1), 107–113, 2008.

[17] White, T. *Hadoop:The Definitive Guide*, O'Reilly, O'Reilly Media, Inc., 1005 Gravenstein Highway North, sebastopol, CA 95472, 2012, 3 edn.

[18] Capriolo, E. *et al.* *Programming Hive*, O'Reilly, O'Reilly Media, Inc., 1005 Gravenstein Highway North, sebastopol, CA 95472, 2012, 1 edn.

[19] Hyde, J. Discardable in-memory, materialized query for hadoop, 2015. URL http://hadoopsummit.org/san-jose/schedule/.

[20] Hagleitner, G. Cost-based optimization in hive, 2015. URL https://cwiki.apache.org/confluence/display/Hive/Cost-based+optimization+in+Hive.

[21] Gupta, H. *et al.* Index selection for olap. In *Proceedings of the Thirteenth International Conference on Data Engineering, ICDE'97*. 208–219, IEEE Computer Society, Washington, DC, USA, 1997.

[22] Gupta, H. & Mumick, I. Selection of views to materialize under a maintenance cost constraint. In Beeri, C. & Bruneman, P. (eds.) *Proceedings of International Conference on Database Theory, ICDT 1999*, vol. 1540 of *LNCS*. 453–470, Springer, Heidelberg, Jerusalem, Israel, 1999.

[23] Miettinen, K. *Nonlinear Multiobjective Optimization*. International Series in Operations Research & Management Science, Springer US, 1999. URL https://books.google.co.in/books?id=ha_zLdNtXSMC.

[24] Serna-Encinas, M. T. & Hoya-Montano, J. A. Algorithm for selection of materialized views: based on a costs model. In *Proceedings of eighth International conference on Current Trends in Computer Science*. 18–24, Morella, Maxico, 2007.

[25] Goswami, R. *et al.* Multiobjective differential evolution algorithm using binary encoded data in selecting views for materializing in data warehouse. In Panigrahi, B. K. *et al.* (eds.) *Swarm, Evolutionary, and Memetic Computing*, vol. 8298 of *LNCS*. 95–106, Springer, 2013.

[26] Lawrence, M. Multiobjective genetic algorithms for materialized view selection in olap data warehouses. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. 699–706, ACM, 2006.

[27] Tusar, T. & Filipic, B. Differential evolution versus genetic algorithms in multiobjective optimization. In *Proceedings of the 4th international conference on Evolutionary multi-criterion optimization*, vol. 4403 of *LNCS*. 257–271, Springer-Verlag, 2007.

[28] Storn, R. & Price, K. Differential evolution- a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization* **11**, 341–359, 1997.

[29] Srinivas, N. & Deb, K. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computing* **2** (3), 221–248, 1995.

[30] Bandyopadhyay, S. *et al.* A simulated annealing-based multiobjective optimization algorithm: Amosa. *IEEE Transactions on Evolutionary Computation* **12** (3), 269–283, 2008.

[31] Gong, T. & Tuson, A. Differential evolution for binary encoding. In *11th Online World Conference on Soft Computing in Industrial Applications (WSC11)*, 2006.

[32] Deb, K. *et al.* A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation* **6** (2), 182–197, 2002.

[33] Gupta, H. *Selection and maintenance of views in a data warehouse*. Phd. thesis, 1999.

[34] Wagner, N. & Agrawal, V. Using an evolutionary algorithm to solve the weighted view materialisation problem for data warehouses. *Int. J. Intelligent Information and Database Systems* **7** (2), 163–179, 2013.

[35] Mohania, M. *et al.* Advances and research directions in data warehousing technology. *Australian Journal of Information Systems* **7** (1), 41–59, 1999.

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

[37] Zhang, C. & Yang, J. Genetic algorithm for materialized view selection in data warehouse environments. In Mohania, M. & Tjoa, A. M. (eds.) *Proceedings of DataWarehousing and Knowledge Discovery, First International Conference, DaWak 1999*, vol. 1676 of *LNCS*. 116–125, Springer, Heidelberg, Florence, Italy, 1999.

[38] Zhang, C. *et al.* An evolutionary approach to materialized views selection in a data warehouse environment. *IEEE Transactions on Systems and Cybernetics Part C: Applications and Reviews* **31** (3), 282–294, 2001.

[39] Lee, M. & Hammer, J. Speeding up materialized view selection in data warehouses using a randomized algorithm. *International Journal of Cooperative Information System* **10**, 327–353, 2001.

[40] Qingzhou, Z. *et al.* An efficient ma-based materialized views selection algorithm. In *Proceedings of the 2009 IITA International Conference on Control, Automation and Systems Engineering*. 315–318, Zhangjiajie, China, 2009.

[41] Sun, X. & Wang, Z. An efficient materialized views selection algorithm based on pso. In *Proceedings of International Workshop on Intelligent Systems and Applications 2009*. 1–4, Wuhan, China, 2009.

[42] Rizzi, S. & Saltarelli, E. View materialization vs. indexing: Balancing space constraints in data warehouse design. In Eder, J. & Missikoff, M. (eds.) *Advanced Information Systems Engineering, 15th International Conference, CAiSE 2003 Proceedings*, vol. 2681 of *LNCS*. 502–519, Springer-Verlag, Berlin Heidelberg, Klagenfurt/Velden, Austria, 2003.

[43] Smith, J. R. *et al.* A wavelet framework for adapting data cube views for olap. *IEEE Transactions on Knowledge and Data Engineering* **16** (5), 552–565, 2004.

[44] Sismanis, Y. *et al.* Dwarf: Shrinking the petacube. In *ACM SIGMOD international conference on management of data (SIGMOD 2002)*. 464–475, ACM, Madison, USA, 2002.

[45] Vijay Kumar, T. Answering query-based selection of materialised views. *Int. J. Information and Decision Sciences* **5** (1), 103–116, 2013.

[46] Horng, J. T. *et al.* Materialized view selection using genetic algorithms in a data warehouse system. In *Proceedings of 1999 congress on Evolutionary Computation*. 2221–2227, IEEE CEC, Washington D.C, USA, 1999.

[47] Loureiro, J. & Belo, O. An evolutionary approach to the selection and allocation of distributed cubes. In *10th International Database Engineering and Applications Symposium, IDEAS'06*. 243–0248, IEEE, 2006.

[48] Yuhang, Z. *et al.* Materialized view selection algorithm – cssa_vsp. In *2010 Second International Conference on Computational Intelligence and Natural Computing Proceedings (CINC)*, vol. 1. 68–71, IEEE, 2010.

[49] Moscato, P. On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Tech. Rep., California Institute of Technology, Pasadena, USA, 1989.

[50] Sedighizadeh, D. & Ellips, M. Particle swarm optimization methods, taxonomy and applications. *International Journal of Computer Theory and Engineering* **1** (5), 486–502, 2009.

[51] Maniezzo, V. *et al.* Ants for data warehouse logical design. In *Proceedings of the 4th Metaheuristics International Conference, Porto*. 249–254, Citeseer, 2001.

[52] Gu, J.-H. *et al.* Application of ant colony system to materialized views selection. *Jisuanji Yingyong/ Journal of Computer Applications* **27** (11), 2763–2765, 2007.

[53] Song, X. & Gao, L. An ant colony based algorithm for optimal selection of materialized view. In *2010 International Conference on Intelligent Computing and Integrated Systems (ICISS)*. 534–536, IEEE, 2010.

[54] Ankerst, M. *et al.* Optics: ordering points to identify the clustering structure. *ACM SIGMOD Record* **28** (2), 49–60, 1999.

[55] Hsu, W. W. *et al.* I/o reference behavior of production database workloads and the tpc benchmarks an analysis at the logical level. *ACM Transactions on Database Systems (TODS)* **26** (1), 96–143, 2001.

[56] Madavan, N. K. Multiobjective optimization using a pareto differential evolution approach. In *Proceedings of the 2002 Congress on Evolutionary Computation, 2002, CEC'02*, vol. 2. 1145–1150, IEEE, 2002.

[57] Xue, F. *et al.* Pareto-based multi-objective differential evolution. In *The 2003 Congress on Evolutionary Computation, 2003, CEC'03*, vol. 2. 862–869, IEEE, 2003.

[58] Iorio, A. W. & Li, X. Incorporating directional information within a differential evolution algorithm for multi-objective optimization. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. 691–698, ACM, 2006.

[59] Kukkonen, S. & Lampinen, J. Gde3: The third evolution step of generalized differential evolution. In *The 2005 IEEE Congress on Evolutionary Computation, 2005*, vol. 1. 443–450, IEEE, 2005.

[60] Deb, K. *et al.* Scalable test problems for evolutionary multi-objective optimization. Tech. Rep., Institute fur Technische Informatik und Kommunikationsetze, Zurich, Switzerland, 2001.

[61] Smith, K. I. *et al.* Dominance-based multiobjective simulated annealing. *IEEE Transactions on Evolutionary Computation* **12** (3), 323–283, 2008.

[62] Radcliffe, N. J. The algebra of genetic algorithms. *Annals of Mathematics and Artificial Intelligence* **10** (4), 339–384, 1994.

[63] Villalobos-Arias, M. *et al.* Asymptotic convergence of metaheuristics for multiobjective optimization problems. *Soft Computing* **10** (11), 1001–1005, 2006.

[64] Villalobos-Arias, M. *et al.* Asymptotic convergence of a simulated annealing algorithm for multiobjective optimization problems. *Mathematical Methods of Operations Research* **64** (2), 353–362, 2006.

[65] Mezura-Montes, E. *et al.* Multi-objective optimization using differential evolution: A survey of the state-of-the-art. *Advances in Differential Evolution* **143**, 173–196, 2008.

[66] Radcliffe, N. J. Equivalenc class analysis of genetic algorithms. *Complex Systems* 183–205, 1991.

[67] Radcliffe, N. J. Forma analysis and random respectful recombination. In *Proceedings of The International Conference on Genetic Algorithms - ICGA 1991*. 222–229, San Marco CA: Morgan Kaufmann, 1991.

[68] Radcliffe, N. J. Non-linear genetic representations. In *Parallel Problem Solving from Nature 2*. 261–270, 1992.

[69] Tuson, A. *No optimization without representation : a knowledge based systems view of evolutionary/neighbourhood search optimization*. Ph.d. thesis, 1999.

[70] Abbass, H. A. & Sarker, R. The pareto differential evolution algorithm. *International Journal of Artificial Intelligence Tools* **11** (4), 531–552, 2002.

[71] Kukkonen, S. & Lampinen, J. An extension of generalized differential evolution for multi-objective optimization with constraints. In *Parallel Problem Solving from Nature-PPSN VIII*. 752–761, Springer, 2004.

[72] Sokal, R. & Michener, C. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin* **38**, 1409–1438, 1958.

[73] Das, S. & Suganthan, P. N. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation* **15** (1), 4–31, 2011.

[74] Cruz, J. R. & Pereira, A. The elitist non-homogeneous genetic algorithm: Almost sure convergence. *Statistics & Probability Letters* **83** (10), 2179–2185, 2013.

[75] Zitzler, E. *et al.* Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation* **8** (2), 173–195, 2000.

[76] Philip Chen, C. & Zhang, C. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences* **275**, 314–347, 2014.

[77] Dean, J. & Ghemawat, S. Mapreduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December*, 2004. URL https://www.usenix.org/legacy/publications/library/proceedings/osdi04/tech/fullpapers/dean/dean_html/index.html.

[78] Fielding, R. T. Certificate of incorporation of the apache software foundation, 1999. URL http://www.apache.org/foundation/records/certificate.html.

[79] Foundation, T. A. S. Apache hadoop, 2014. URL http://hadoop.apache.org/.

[80] Foundation, T. A. S. Apache hive tm, 2014. URL http://hive.apache.org.

[81] Pukelsheim, F. *The three sigma rule*. Report, Inst. für Math., 1992. URL https://books.google.co.in/books?id=Q1jPGwAACAAJ.

[82] Benson, H. P. & Sayin, S. Towards finding global representations of the efficient set in multiple objective mathematical programming. *Naval Research Logistics (NRL)* **44** (1), 47–67, 1997.

[83] Inc., H. Hortonworks data platform, 2014. URL http://hortonworks.com/hdp/.

[84] Lahman, S. Baseball archive, 2014. URL http://seanlahman.com/files/database/lahman591-csv.zip.

[85] Smith, K. *et al.* Dominance measures for multi-objective simulated annealing. In *Congress on Evolutionary Computation, CEC2004*, vol. 1. 23–30, IEEE, 2004.

[86] Bandyopadhyay, S. *et al.* A simulated annealing-based multiobjective optimization algorithm: Amosa. *IEEE Transactions on Evolutionary Computation* **12** (3), 269–283, 2008.

[87] Lahtinen, J. *et al.* Empirical comparison of stochastic algorithms , 1996.

[88] Mann, J. W. & Smith, G. D. A comparison of heuristics for telecommunications traffic routing. In Rayward-Smith, V. J. *et al.* (eds.) *Modern Heuristic Search Methods*. 235–254, John Wiley and sons Ltd, 1996.

[89] Schott, J. R. *Fault tolerent design using single and multi-criteria genetic algorithms*. Ph.d. dissertation, 1995.

[90] Kirkpatrick, S. *et al.* Optimization by simulated annealing. *Science* **220**, 671–680, 1983.

[91] Jain, A. & Dubes, R. *Algorithms for Clustering Data*, Englewood Cliffs, NJ:Prentice-Hall, 1988.

[92] Toussaint, G. Pattern recognition and geometrical complexity. In *Proceedings of 5th International Conference on Pattern Recognition*. 1324–1347, Miami Beach, FL, 1980.

[93] Rudolph, G. On a multi-objective evolutionary algorithm and its convergence to the pareto set. In *The 1998 IEEE International Conference on Evolutionary Computation Proceedings, 1998, IEEE World Congress on Computational Intelligence*. 511–516, IEEE, 1998.

[94] Bandyopadhyay, S. *et al.* A simulated annealing-based multiobjective optimization algorithm: Amosa. *IEEE Transactions on Syst.,Man, Cybern.-B* **34** (5), 2088–2099, 2004.

Publications based on the Thesis Works

Journals

1. **Goswami, R.**, Bhattacharyya, D.K., Dutta, M. and Kalita J.K., "Approaches and Issues in View Selection for Materializing in Data Warehouse", *International Journal of Business Information Systems, InderScience*, Vol. 21, No. 1, pp. 17-47, 2016, DOI: 10.1504/IJBIS.2016.073379.

2. **Goswami, R.**, Bhattacharyya, D.K. and Dutta, M., "Materialized View Selection Using Evolutionary Algorithm for Speeding up Big data Query Processing", *Journal of Intelligent Information Systems, Springer*, 2015 [under review].

Book Chapters

3. **Goswami, R.**, Bhattacharyya, D.K. and Dutta, M., "Selection of views for materializing in data warehouse using MOSA and AMOSA," *in Advances in Computer Science, Engineering & Applications, AISC, Springer, Heidelberg*, Print ISBN: 978-3-642-30156-8, In: Wyld, D.C., Zizka, J., and Nagamalai, D., Editors. Vol. 166, pp. 619-628, 2012, DOI:10.1007/978-3-642-30157-5_62.

4. **Goswami, R.**, Bhattacharyya, D.K. and Dutta, M., "Multiobjective Differential Evolution Algorithm Using Binary Encoded Data in Selecting Views for Materializing in Data Warehouse", *Swarm, Evolutionary, and Memetic Computing, LNCS, Springer*, Vol.8298, pp 95-106, 2013, Print ISBN: 978-3-319-03755-4, DOI:10.1007/978-3-319-03756-1_9.

Conference proceedings

5. **Goswami, R.**, Bhattacharyya, D.K. and Dutta, M., "Approached and Issues in View Selection for Materializing in Data Warehouse", *in the proc. of National Conference on Trends in Machine Intelligence (NCTMI11)*, ISBN:978-

81-8487-140-1, pp. 76-82, March 23-25, 2011, Tezpur University, Tezpur -784028, India.