J Intell Inf Syst (2017) 49:407–433
DOI 10.1007/s10844-017-0455-6

Materialized view selection using evolutionary algorithm for speeding up big data query processing

Rajib Goswami[1] · **D. K Bhattacharyya**[1] · **Malayananda Dutta**[2]

Received: 6 March 2015 / Revised: 14 February 2017 / Accepted: 1 March 2017 /
Published online: 16 March 2017

Abstract For speeding up query processing on Big Data, frequent sub-queries or views may be materialized such that the query processing cost is minimized with optimum cost of maintaining the materialized views and/or queries. Materializing frequent sub-queries and views means that resultant data set of the views reside in the memory of one or more nodes in the cluster, so that it reduces the MapReduce cost, submission and scheduling cost of Distributed File System jobs for query processing. We have defined materialized views as resultant data of frequent sub-queries and aggregation functions of a set of Big Data warehousing queries that are saved for enhancing query performance. 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. We applied Differential Evolution algorithm and NSGA-II to study their performances for developing a recommendation system for selecting views for materializing in Big Data warehousing.

Keywords Big data warehouse · Differential evolution algorithm · Hadoop · Hive · Materialized view · Multi-objective optimization · NSGA-II

1 Introduction

Views are derived relations from the base relations or tables used for increasing performance of data warehouse query processing. In data warehousing, historical data are analyzed by

✉ Rajib Goswami
rgos@tezu.ernet.in

D.K Bhattacharyya
dkb@tezu.ernet.in

Malayananda Dutta
malay@iiitg.ac.in

[1] Department of Computer Science and Engineering, Tezpur University, Tezpur, 784028, India

[2] Department of Computer Science and Engineering, Indian Institute of Information Technology Guwahati, Guwahati, 781001, India

making some complex queries. Some of these queries are triggered very frequently in On-Line Analytical Processing (OLAP) applications. Views are set of logical relational data derived from the base tables so that complex queries can be simplified by accessing these views. If the retrieved data of intermediate views of these queries are saved or materialized then instead of generating and retrieving data from the base tables for these derived relations again and again, they may be directly read from the materialized views. The materialized views can be indexed for further increasing the query performance. These materialized views are designed by aggregate functions on base tables, or as copies of frequently executed sub-queries of a set of frequent queries. Materialized views are to be refreshed or maintained for up to date data incurring some amount of processing cost. Materialization of the intermediate temporary views generated while processing queries means extra requirement of space. Therefore, it is necessary to select an optimum set of views to materialize to increase query processing performance, with optimized materialized view maintenance cost and materializing space cost. The selection of views for materializing to minimize query processing cost with minimized view maintenance cost and space requirement for saving the selected views is termed as the *materialized view selection problem* (Harinarayan et al. 1996; Gupta et al. 1997; Goswami et al. 2016).

1.1 Views, materialized views and materialized queries in Big-Data management technologies

Big Data management 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 (Dean and Ghemawat 2004). It imposes a programming model termed as *MapReduce* that 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 (Dean and Ghemawat 2004; White 2012). The distributed file system used in Big Data management by *Apache* (Roy and Fielding Laguna Beach 1999) is known as *Hadoop Distributed File System (HDFS)* (Foundation 2014a; White 2012). For Data Warehousing applications in HDFS, Hadoop (Foundation 2014a) provides a technology called *Hive* and an SQL like language called *HiveQL* (Foundation 2014b). 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 proposes an extension to materialized views called *Discardable, In-Memory Materialized Query - DIMMQ for Hadoop* (Hyde 2014). DIMMQ proposes that the resultant dataset 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 with minimized total query processing cost of a set of queries defined as frequent warehouse queries for a specific data warehouse application.

1.2 Materialized view selection for big data management framework

In Big Data management framework a query is executed by accessing data spread over a cluster of hardware storage or data-nodes as distributed file system (DFS) 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. Thus the materialized view selection problem is 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 solution set of views amounts to different query processing costs and other associated costs for different combination of views, independent of total number of views selected. Thus the solution space increases exponentially with increased number of queries and underlying views considered. Thereby the problem becomes NP-hard.

1.3 Existing approaches

Initially, several heuristic greedy approaches have been proposed by defining different cost parameters to deal with the view selection for materializing problem in conventional RDBMS based data warehousing (Harinarayan et al. 1996; Gupta et al. 1997; Gupta and Mumick 1999; Nadeua and Teorey 2002; Serna-Encinas and Hoya-Montano 2007). In different representation of the view selection problem using conventional relational model, it has been observed that the materialized view selection problem is an NP-hard problem (Gupta and Mumick 1999; 2005; Aouiche and Darmont 2009). Therefore, various stochastic, evolutionary algorithms and data mining with clustering based approaches have been proposed with different types of data structures and representations (Derakhshan et al. 2006; Derakhshan et al. 2008; Yang et al. 1997; Zhang and Yang 1999; Zhang et al. 2001; Aouiche et al. 2006; Song and Gao 2010; Goswami et al. 2012; 2013; Goswami et al. 2016). Recent approaches use randomized algorithms such as Simulated Annealing (SA), Parallel Simulated Annealing (PSA), Multi Objective Simulated Annealing (MOSA), Evolutionary Algorithms like Genetic and Memetic Algorithm (MA), Particle Swarm Optimization (PSO) etc. using query execution plan derived directed acyclic graphs (DAG) representation from frequent query workload over a historical period (Lawrence 2006; Derakhshan et al. 2006; Derakhshan et al. 2008; Loureiro and Belo 2006; Zhang and Yang 1999; Zhang et al. 2001; Lee and Hammer 2001; Sun and Wang 2009; Qingzhou et al. 2009; Goswami et al. 2012;

2013). In some recent approaches the problem has been handled by defining the problem as multi-objective optimization problem (Lawrence 2006; Goswami et al. 2012, 2013).

1.4 Our approach on materialized view selection for big data warehousing

In this paper we present our attempt to design a system for finding solution set of views for materializing to optimize query processing cost, materialized view maintenance cost and HDFS storage from views generated while processing a set of queries on Big Data warehousing in HDFS framework. 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 (Deb et al. 2002). The Differential Evolution (DE) algorithm is a powerful stochastic real-parameter optimizer for non-linear and non-differentiable continuous space function (Storn and Price 1997). Gong and Tuson (2006) present the use of *forma analysis* to exploit usage of DE for discrete optimization problem. Here, we customized *forma analysis* based single objective DE presented in Gong and Tuson (2006) to multi-objective DE for binary encoded data (MODE-BE) for selecting views for materializing in HDFS data warehouse framework by promoting diversity in decision 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. We have also developed a prototype of recommendation system using NSGA-II 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 framework.

1.5 Organization of this paper

The problem of selecting views for materializing in DFS data warehousing is defined in Section 2. In Section 3, materialized view selection in HDFS is defined as a multi-objective optimization problem. Multi-objective Genetic Algorithms and Differential Evolution algorithms in solving multi-objective optimization problems are discussed in Section 4. The design of a 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 is presented in Section 5. In Section 6, the experimentation process has been presented with test data and frame work used, along with an analysis on obtained results by implementing the algorithms discussed in Section 5. Section 7 presents comparative analysis of state-of-the-art techniques with respect to quality of solutions for materialized view selection problem in Big Data warehousing. Finally in Section 8 concluding remarks and perspectives are presented.

2 The problem of selecting views for materializing in big data management 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 Hadoop's scale out and robust capabilities for data storage and processing on large number of commodity hardware. HiveQL enables to do ad-hoc querying, summarization and data

analysis on massive data easily (Foundation 2014b). 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 (Foundation 2014b) query processing on Hadoop version 1 often had to submit number of MapReduce jobs to complete a query processing. With Hadoop2 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 (Hagleitner 2014). But in general, for Big Data management in HDFS, processing standard is by MapReduce (Foundation 2014a). 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 function. 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 (Hagleitner 2014).

2.1 The cost model and problem definition of view selection for materializing in HDFS big data management

The cost model to be used for handling materialized view selection problem for HDFS 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 HDFS 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 unstructured or semi-structured data in batch fashion unlike the traditional RDBMS where data-set has been indexed to deliver low-latency seek and update time (Dean and Ghemawat 2004). 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, substantial overheads in job submission and scheduling. In Big data management DFS, block size and split size are fixed. Therefore, in HDFS/MapReduce framework, a small number of large files are better than large number of small files (White 2012). This means that in case of HDFS, smaller number of bigger views are to be preferred for materializing. This criterion is not applicable in case of traditional RDBMS based data warehouse 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.

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 HDFS, the query processing cost of the set of considered queries may be defined as Definition 1.

Definition 1 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, and $n \leq m$, if $V' \subseteq V$ is the set of views $V' = \{v_1, v_2, \cdots, v_p\}$ that are materialized, the total HDFS query processing cost can be defined by the following expression.

$$C^Q_{V'} = C^Q_{|V'|=0} - \sum_{i=1}^{p} M_{v_i} \tag{1}$$

where $C^Q_{|V'|=0} = \sum_{i=1}^{n} M_{q_i}$ is the total query processing cost of Q without materializing, and $\sum_{i=1}^{p} M_{v_i}$ is the MapReduce cost of processing V'.

2.1.2 Materialized view maintenance cost

In Big Data analysis on HDFS based warehouse, there are generally very few occurrence 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 (Hyde 2014). 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. Thus there will be another set of DFS overheads. The materialized view maintenance cost therefore may be defined as follows.

Definition 2 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

$$U(V') = \sum_{i=1}^{p} U_{v_i} \tag{2}$$

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

2.1.3 Number of views to be materialized and storage space requirements

In HDFS, smaller number of bigger tables are preferred (as discussed in Section 2.1). The storage space requirements for p number of materialized views V', where $\left|V'\right| = p$, can be defined as Definition 3 below.

Definition 3 If S_{v_i} is the storage space required by ith materialized view, then the total space required for materializing p number of views is

$$S(V') = \sum_{i=1}^{p} S_{v_i} \tag{3}$$

2.1.4 The materialized view selection problem

Considering the Definitions 1, 2 and 3, the view selection for materializing in HDFS based data warehousing can be stated as Definition 4 below.

Definition 4 The view selection for materializing in DFS based data warehousing for a given set of n frequent queries $Q = \{q_1, q_2, \cdots, q_n\}$ on an HDFS based data warehouse, where V is a set of m views generated while processing Q, such that $n \leq m$, 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. $C^Q_{V'}$ defined by (1),
2. $U(V')$ defined by (2) and
3. $|V'| = p$ with maximized $S(V')$ as defined by (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.

3 Materialized view selection in big data management by DFS: A multi-objective optimization problem

From the above Definitions 1, 2, 3 and problem statement in Section 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 -

$$\mathbf{Y} = \mathbf{f}(\mathbf{V}') \equiv (C^Q_{V'}, U(V'), |V'|) \tag{4}$$

such that the space requirement for materializing V' i.e., $S(V')$ is maximized.

3.1 Simple problem representation

Deb et al. (2001) suggest few important features that must be present in an multi-objective optimization problem for solving by randomized and evolutionary algorithm. According to Deb et al. (2001), 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 HDFS data warehouse can be derived or known. In our definition this known set of views are defined as V. Where $|V| = m$ and the cardinality of selected views for materializing V', $|V'| = p$. As $p \leq m$, the dimension of the solution is m. A solution vector may be defined as a string of bits of length m as each of the m dimensions 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 labelled 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 labelled as the first view, the second bit represents the view labelled 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, it's 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 number 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 $S_0 \not\prec S_1$ and $S_1 \not\prec S_0$, then S_0 and S_1 are two non-dominating solutions of the problem.

Definition 5 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 (4).

3.2 Scalability

In Deb et al. (2001), it has been presented 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.

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 (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 (1) and (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 Genetic algorithms and differential evolution algorithms in solving multi-objective optimization problem

Due to exponential expansion of solution search space with number of decision vectors, the problem is best suited for applying randomized algorithms (Zhang et al. 2001). 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 (Lawrence 2006) for it's ability to find multiple Pareto-optimal solutions in one single run (Deb et al. 2002). But with basic MOEAs like NSGA (Srinivas and Deb 1995), criticisms like high computational complexity of non dominating sorting, lack of elitism etc. were there. NSGA-II is an improved version of NSGA proposed by Deb et al. in Deb et al. (2002).

Differential Evolution (DE) algorithm is another Evolutionary Algorithm introduced by Storn and Price (1997), which outperforms Genetic Algorithms (GAs) on many numerical single objective optimization problem (Tusar and Filipic 2007). Though the original DE was designed for single objective optimization, recently many approaches have

been developed that use NSGA-II suggested non-dominated sorting of Pareto-ranks and crowding distance based elitism of solution population for adopting multi-objective optimization problem (Tusar and Filipic 2007; Madavan 2002; Xue et al. 2003; Iorio and Li 2006; Kukkonen and Lampinen 2005). In (Deb et al. 2001), a set of 9 test problems termed as DTLZ test problems for testing and comparing performances between different MOEAs are suggested. In Kukkonen and Lampinen (2005), a generalized DE algorithm for multi-objective optimization named GDE3 was presented where it was presented that with certain parameters, tri-objective test problems DTLZ1 and DTLZ4 performs better than NSGA-II. Another version of multi-objective DE named DEMO, that uses NSGA-II based methods for diversity preservation and elitism in solution, shows comparable performances with respect to NSGA-II in case of DTLZ1 to DTLZ7, when tested for 2,3 and 4 objectives (Tusar and Filipic 2007) (Table 1).

Next, we present a customized multi-objective DE and an NSGA-II using binary encoded solution vector population for materialized view selection problem.

5 Multi-objective differential evolution algorithm and NSGA-II for materialized view selection in HDFS based data warehouse

It has been observed from our discussion presented in Sections 4 and 5 that multi-objective Evolutionary Algorithms (MOEAs) are suitable for applying in materialized view selection problem. In our solution representation for handling this problem we have defined solution vectors as a string of bits. Therefore, we have customized single objective DE for binary encoded data as presented by Gong and Tuson (2006) for multi-objective optimization to handle materialized view selection problem. We have also implemented NSGA-II proposed by Deb et al. (2002) to handle this problem for analyzing their performances.

Table 1 Performances of multi-objective DE and NSGA-II with respect to DTLZ test problems

Test Problem	Algorithm	Performance compared to NSGA-II for 3 objectives
DTLZ1	DEMO^{NS-II}	Almost same
	GDE3	Almost same
DTLZ2	DEMO^{NS-II}	No difference
	GDE3	Not evaluated
DTLZ3	DEMO^{NS-II}	Almost same
	GDE3	Not evaluated
DTLZ4	DEMO^{NS-II}	No difference
	GDE3	Almost same
DTLZ5	DEMO^{NS-II}	No difference
	GDE3	Not evaluated
DTLZ6	DEMO^{NS-II}	Almost same
	GDE3	Not evaluated
DTLZ7	DEMO^{NS-II}	No difference
	GDE3	Not evaluated

5.1 Multi-objective DE with binary encoded solution representation (MODE-BE) for materialized view selection

The Differential Evolution (DE) algorithm is a stochastic parallel direct search method for *real vectors*. DE uses NP number of D-dimensional parameter vectors $x_{i,g}$, $i = 1, 2, \cdots, NP$, representing NP as the population size for a population generation g of an evolutionary system (Storn and Price 1997). 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. The DE introduced by Storn and Price in Storn and Price (1997) was originally designed for global optimization problem over *continuous spaces* using solution population of real vectors. 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 (5).

$$v_{i,g+1} = x_{r_1,g} + F.(x_{r_2,g} - x_{r_3,g}) \tag{5}$$

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 $F > 0$. 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.

Gong and Tuson (2006) used *forma analysis* (Radcliffe 1991a, b) to derive discrete DE operators for *discrete single objective otimization* 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. To compute mutant vector in DE, the difference between two random vectors of the population, $x_{r_2,g}$ and $x_{r_3,g}$ is to be amplified by a real amplication factor F and then it is added to another random vector $x_{r_1,g}$ of the population. By using *forma basis* (Radcliffe 1991a, b), Gong and Tuson (2006) expressed mutant vector defined by (5) for binary encoded solution vector as (6).

$$v_{j,i,g+1} = D_{\Psi_j}(x_{r_1,g}, F.D_{\Psi_j}(x_{r_2,g}, x_{r_3,g})) \tag{6}$$

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* (Gong and Tuson 2006) Ψ_j as (7).

$$D_{\Psi_j}(\mathbf{x}, \mathbf{y}) = \begin{cases} 0, & \text{if } x_j = y_j \\ 1, & \text{otherwise} \end{cases} \tag{7}$$

As in DE, F is a real number in the range [0, 1], $F.D_{\Psi_j}(x_{r_2,g}, x_{r_3,g})$ will be real value F or 0, and the vector will be transformed into a real vector. 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, (8) is used.

$$F.D_{\Psi_j}(x_{r_2,g}, x_{r_3,g}) = \begin{cases} 1, & \text{if random } [0, 1] < F \wedge (D_{\Psi_j}(x_{r_2,g}, x_{r_3,g}) = 1) \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

For selecting materialized views for conventional data warehousing, the mutant vectors in multi-objective DE with binary encoded data, the *forma basis* as discussed in Radcliffe (1991a, b) has been used in Goswami et al. (2013). We have used forma basis based multi-objective DE for binary encoded data (MODE-BE) to design Algorithm 1 for selecting views to materialize in HDFS data warehousing.

Algorithm 1 Multi-objective Differential Evolution using Binary Encoded Data for selecting materialized view in HDFS based data warehouse

Require: NP, g_{max}, F, CR, D, $C^{Q}_{|V'|=0}$, $M_{v_{i=1,\cdots,m}}$, $U_{v_{i=1,\cdots,m}}$, $S_{v_{i=1,\cdots,m}}$, 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**
    Compute and assign Pareto rank to each of the population vectors $x_{i=1,\cdots NP}$;
    Compute maximum distance of each solution vector to any other solution vectors of the population, Max_i ;
    Sort vectors $x_1, x_2, \cdots, x_{NP}$ in ascending order of Pareto-rank and descending order of Max_i ;
    $NP \leftarrow N$;
  end if
  $g \leftarrow g + 1$;
until $(g < g_{max})$;
Remove all dominated solutions from the population list $x_1, x_2, \cdots, x_{NP}$;
return Remaining non-dominated solutions from the final population list ;

In Algorithm 1, 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 in (6) (and (8)). Then trial vector $u_{i,g+1}$ is formed by crossover. To adapt the problem of view selection to materialize in HDFS Data Warehouse, the query processing cost, the materialized view maintenance cost and the number of views in solution sets of the considered HiveQL frequent queries Q, for each solution vector $x_{i,g}$ and trial vector $u_{i,g+1}$ are computed using (1),(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 NP, a technique is used to filter out NP elite solution population that maintains diversity in the solution population as discussed in Section 5.1.1. 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 removed from the archive to return the non-dominated solutions of the problem.

5.1.1 Promoting diversity and controlling the size of solution population

To control the population size by keeping the diversity in solution population in the intermediate generations of DE in our approach, diversity 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 Deb et al. (2002). To include all previous and current population members to ensure finding out the most elite NP solutions, Deb et al. (2002) suggests $\Gamma = 2$. 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 (Sokal and Michener 1958) is used . 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. We used density in solution space instead of *crowding distance* in objective function value space to promote diversity in solutions.

5.1.2 Complexity

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. The algorithm terminates when the maximum specified number of generations, i.e., g_{max} is reached. Therefore, the run time complexity is $O(NP.D.g_{max})$ (Das and Suganthan 2011). 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. 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 for M objectives is thus $O(MN^2)$ (Deb et al. 2002). For computing SMC based maximum neighborhood distances of each of the N solutions, complexity is $O(N^2)$. For two independent sorting of at most N solutions,

complexity is $O(2NlogN)$. Thus these three processings 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 (=3) are held constant. Thus the overall run time complexity is $O(g_{max}.N^2)$.

5.2 Implementing NSGA-II for materialized view selection in big data management by DFS

In Lawrence (2006), 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. (2002) 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 (Deb et al. 2002).

To adapt NSGA-II for selecting views to materialize in Big Data warehousing with distributed file system framework, particularly HDFS, we used the cost model and problem definition as discussed in Sections 2 and 3.

Each solution vector in our solution model is represented as $\Psi = \{\psi_1, \psi_2, \cdots, \psi_D\}$ of dimension D, where each dimension is a decision variable of one bit such that ψ_1 represents whether ith view is selected for materializing or not. First a random solution population $\Psi_1, \Psi_2, \cdots, \Psi_{NP}$ are created. 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 4. Randomly these p number of decision variables are set as 1 for the vector Ψ_i. As discussed in Section 2, in Big Data framework based data warehousing, 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 already not in the solution population.

In subsequent generations, the usual binary tournament selection, recombination and mutation operators are applied to the NP solutions to create offsprings. 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.

5.2.1 Complexity of NSGA-II for materialized view selection in HDFS

The basic operations of NSGA-II based application's worst case complexities as presented in Deb et al. (2002) 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 $O(N^2)$.

6 Experimentation and discussion

For experimental analysis we implemented Multi-Objective DE for Binary Encoded solutions (MODE-BE) and NSGA-II as a recommending system which takes input from log-files generated on processing HiveQL instructions by HDFS. 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.

6.1 Experimental setup

For our experimental setup, we used Hortonworks Data Platform (HDP) version 2.0.6 with Hortonworks Sandbox version 2.0 VMware for 64 bit CentOS operating system workstation 6.5-7.x virtual machine (Inc. 2014). This is a single node implementation for experimenting HDFS. We used Hadoop version 2.2.0 and Hive version 0.12.0 of Apache (Foundation 2014b), which lets us manage data, perform ad-hoc queries and analysis of large data-set/data warehouse in Hadoop cluster. For executing HiveQL queries, we used an HDP interactive interface to Hive^{TM} named Beeswax provided by HDP. 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 our test-bed is presented in Fig. 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 our experimentation we used Lahman Baseball Database of American Major League Baseball statistics from 1871 through 2011 (Lahman 2014) as suggested by HDP 2.0.6 for experimenting with Hadoop version 2.2.0 .

We synthesized 20 HiveQL queries on Lahman baseball database for our experimentation. 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 2. The queries and constituent views are triggered to HDP to get query processing costs, view processing costs and maintenance costs in terms of MapReduce 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 Tables 3 and 4. 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 labelled 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 labelled as second view is not to be selected for materializing and so on. This representation is used in many materialized view selection techniques (Derakhshan et al. 2006; Derakhshan et al. 2008; Goswami et al. 2012, 2013). The extracted costs of our queries in one instance of execution as we present in Tables 3 and 4 are used as input to our multi-objective EA based view selection recommendation system.

6.2 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 Storn and Price (1997) it has been suggested that the value of NP should be around 5 to 10 times the dimensionality of the problem. Therefore, for 25 candidate views, we may st 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 Das and Suganthan (2011) and Tusar and Filipic (2007), for population size NP between 3 to 8 times of the dimensionality of the problem, the mutation scaling factor F=0.6 and

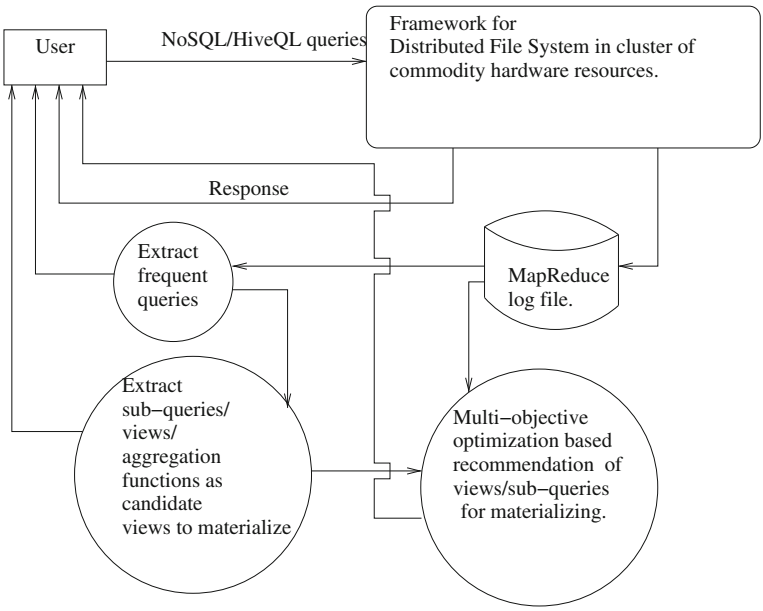

Fig. 1 Test-bed for selecting non-dominated solution sets of materialized views

Table 2 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}

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,
- 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

Table 3 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

may use unnecessary MapReduce split and overhead, Hadoop works better with smaller number of larger files (White 2012).

6.3 Results and discussion

In our experimentation with our experimental setup, parameters and data we observe 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 (Goswami et al. 2013).

Table 4 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

- From the query processing costs in terms of MapReduce time, plotted in Fig. 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 (Fig. 3).
- From Figs. 4, 5, 6 and 7 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.

7 Comparative analysis of state-of-the-art techniques with respect to quality of solutions

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. (2001). Derakhshan et al. (2006) introduce an approach for materialized view selection using Simulated Annealing (SA) with Multiple View Processing Plan (MVPP) (Yang et al. 1997) of frequent data warehouse queries as input. This approach was later modified by applying Parallel Simulated Annealing (PSA) (Derakhshan et al. 2008). In (Goswami et al. 2012), Multi-Objective Simulated Annealing (MOSA) (Smith et al. 2004; Smith et al. 2008) and Archived Multi-Objective Simulated Annealing (AMOSA) (Bandyopadhyay et al. 2008) are applied in materialized view selection problem using MVPP based representation of the problem (Goswami et al. 2016). A comparative analysis of Multi-Objective Simulated Annealing algorithm, MODE-BE and NSGA-II based techniques with respect to quality of solutions in view selection problem in Big Data paradigm has been presented in this section.

7.1 Comparison measures used and obtained values

The performance of different randomized multi-objective optimization algorithms may be analyzed in different ways. In materialized view selection, the objective is to find solutions as sets of views such that the solutions 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. The obtained solutions should be the nearest to the true solutions as well as they should be well represented from the actual complete set

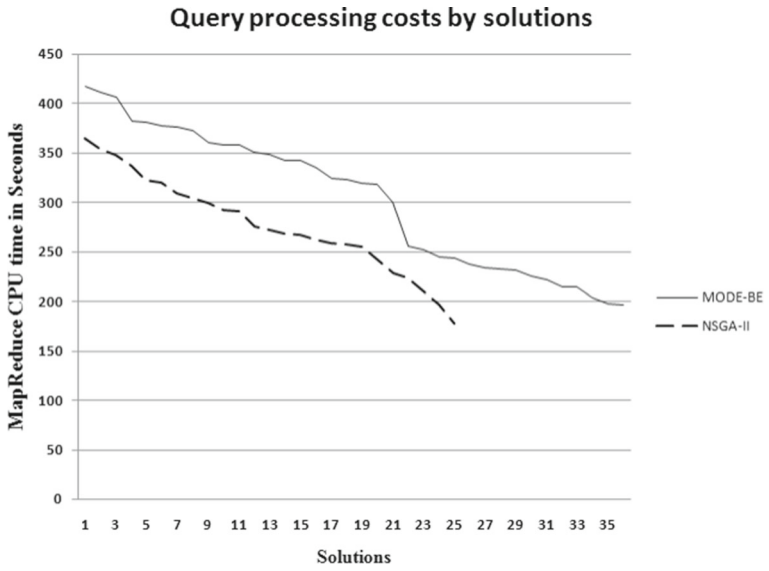

Fig. 2 Processing MapReduce cost (in Seconds) by NSGA-II and MODE-BE generated non-dominated solutions

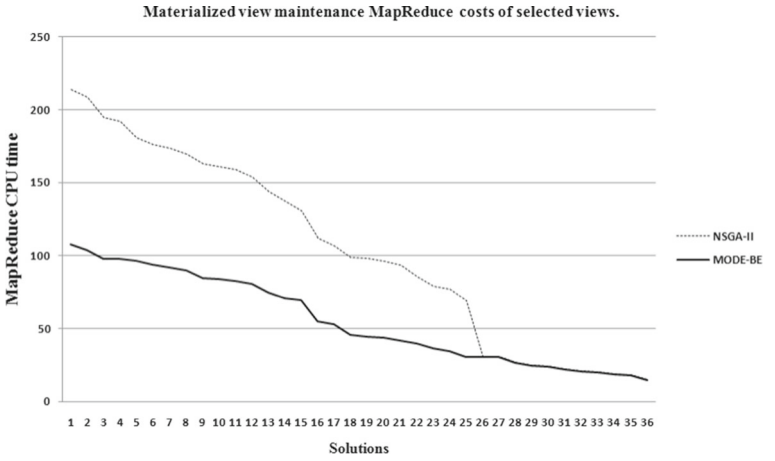

Fig. 3 Materialized view maintenance MapReduce cost (in Seconds) by NSGA-II and MODE-BE generated non-dominated solutions

of non-dominated solutions. By a single measuring parameter the performance of a multi-objective optimization algorithm in terms of quality of solutions can not be measured. In our work three measures have been used to evaluate the quality of obtained solutions. They are- (1) the extent of Convergence (Deb et al. 2002) of the solution set to an already known set of Pareto optimal solutions, (2) the fraction of solutions that remain non-dominated with respect to all solutions by other algorithms termed as the Purity (Bandyopadhyay et al. 2004) of solutions and (3) uniformity of the Spacing (Schott 1995; Deb et al. 2002; Bandyopadhyay

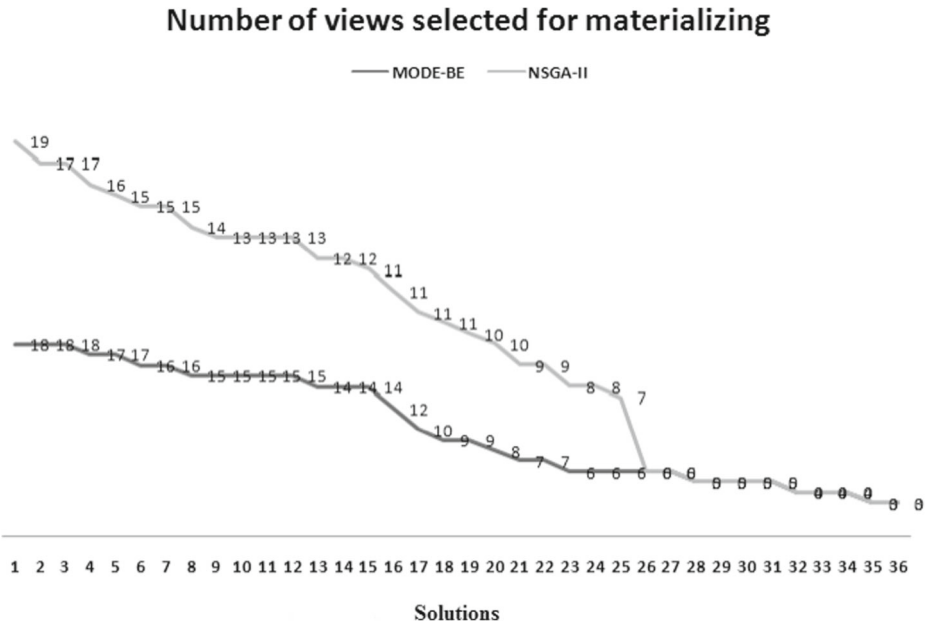

Fig. 4 Number of views in solution sets for materializing

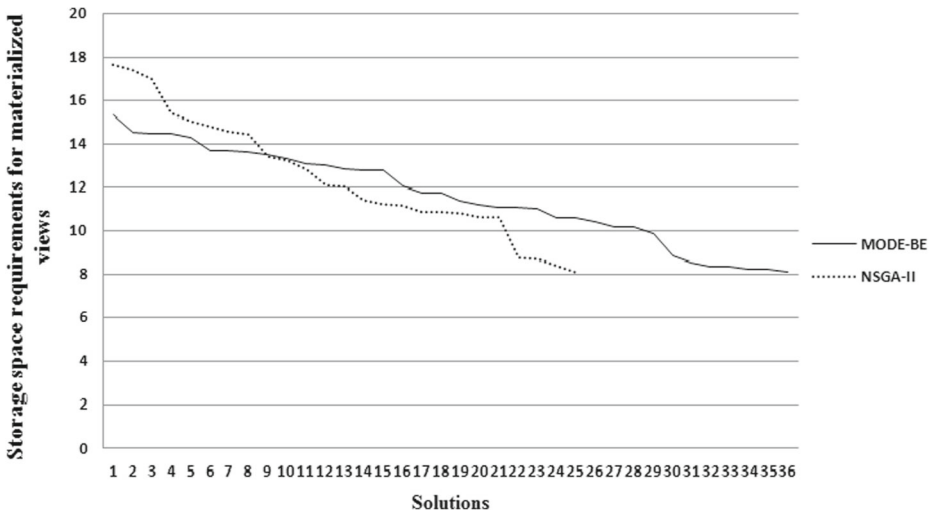

Fig. 5 Space requirements by solution sets of views for materializing

et al. 2004) between the solutions over the non dominated front. Experimental data sets have been generated using the setup presented in Section 6.1. Other than the 20 queries and 25 associated views presented in Section 6.1, another set of data is generated for 109 queries and 51 candidate views for experimenting with higher dimensional data. Using this 109 queries, two other data sets have been generated for 50 and 60 queries sharing the 51 candidate views.

Convergence measure γ The convergence measure γ presented in Deb et al. (2002) 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. 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 mentioned above, for AMOSA, 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. The convergence measures presented in Table 5 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.

For computing Convergence (γ), first the set of non dominating solutions obtained by already used algorithms for the application (materialized view selection), 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 lower the value γ, better is the convergence of the solution set obtained to the true Pareto optimal front.

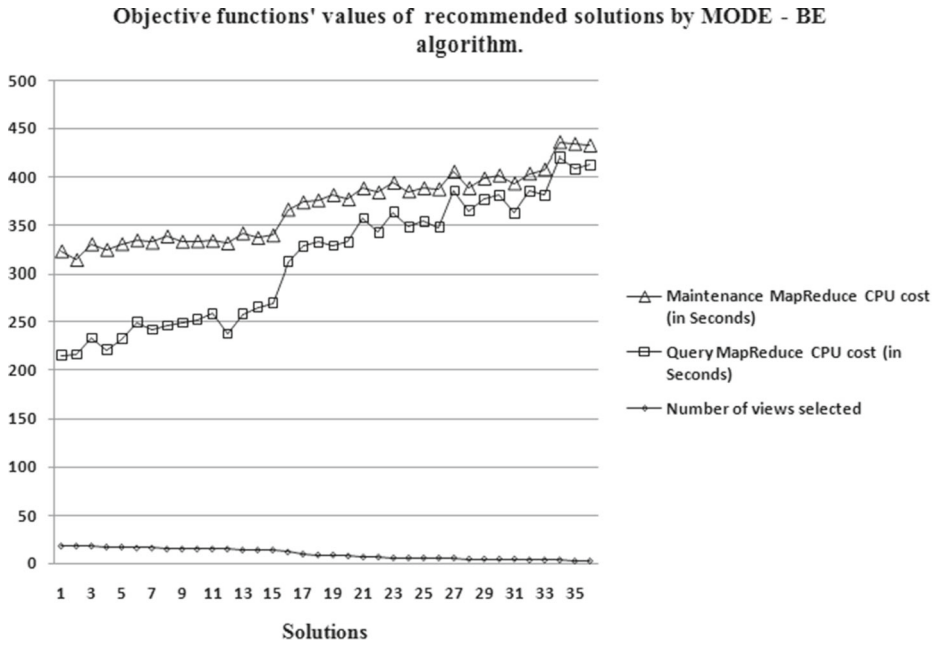

Fig. 6 Objective functions' values by MODE-BE generated non-dominating solutions

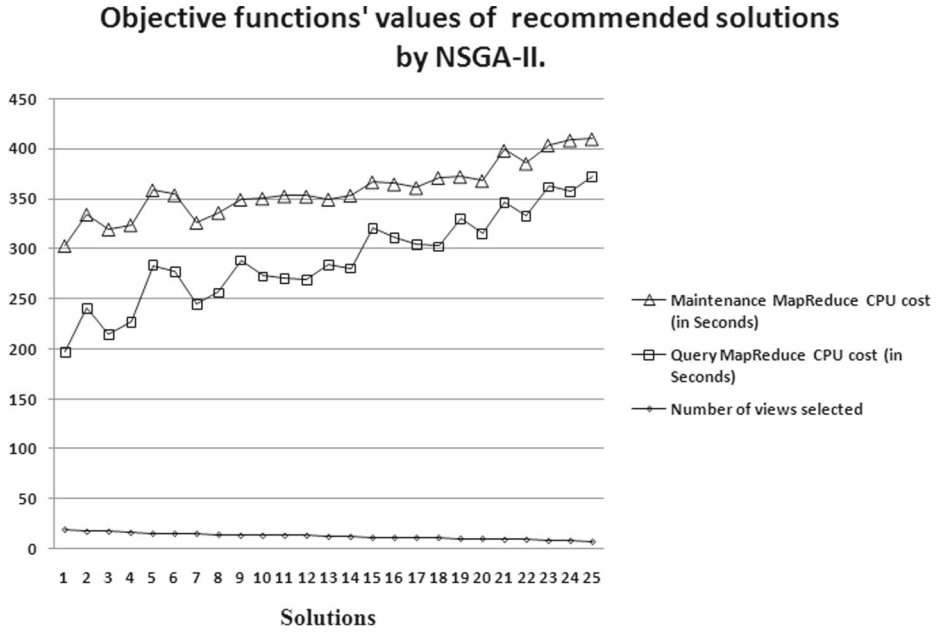

Fig. 7 Objective functions' values by NSGA-II generated non-dominating solutions

Table 5 Convergence (γ)

Number of queries	Number of candidate views	AMOSA 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

Purity The Purity measure (Bandyopadhyay et al. 2004) is the fraction of solutions from one particular technique that remains non-dominated by all (non-dominated) solutions obtained by the other techniques that are considered for comparing their solution quality. The Purity measure obtained by AMOSA, MODE-BE and NSGA-II algorithms in materialized view selection for data warehousing in our experimentation using the data sets mentioned above are presented in Table 6.

The Purity value 1 by an algorithm means all the solutions it has produced are not dominated by solutions produced by any other algorithm so far. Therefore, the Purity value near 1 indicates better performance and the value near 0 means poorer performance.

Spacing and Minimal Spacing The multi-objective optimization techniques are aimed to obtain a set of solutions spanning the entire Pareto-optimal region of the solution space. Schott in Schott (1995) proposed the measure of *Spacing*, S, to reflect the uniformity of the solutions over a non-dominated front. The Spacing, S is computed as expressed below.

$$S = \sqrt{\frac{1}{|Q|}\sum_{i=1}^{|Q|}(d_i - d)^2} \tag{9}$$

where $d_i = min_{k \in Q and k \neq i} \sum_{m=1}^{M} |f_m^i - f_m^k|$ 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 it is not possible to indicate the actual spaces between the solutions in cases where the complete true Pareto front is not known or only a segment of the front is considered for computing the S. Therefore in Bandyopadhyay et al. (2004), a modified measure for this *Minimal Spacing*, denoted as S_m is proposed, where $|Q|$ is replaced by $|Q| - 1$ as actually $|Q| - 1$ number of distances are considered here for measuring. Again

Table 6 Purity

Number of queries, shared views	AMOSA	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 7 Minimal Spacing

Number of queries, shared views	AMOSA	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

there may be diverse objective function values. Therefore, the term $|f_m^i - f_m^k|$ is divided by $|F_m^{max} - F_m^{min}|$ for normalizing the objective function values. Here F_m^{max} and F_m^{min} are the maximum and minimum objective function values respectively of mth objective. Larger value of S_m reflects that the solutions are not uniformly distributed over the known Pareto-optimal front. That is, smaller value of S or S_m indicates better performance. In this work, other than the Convergence and Purity measures, the algorithms are also analyzed for how the solutions are distributed over the known true Pareto front using the Minimal spacing measure S_m. Minimal Spacing values obtained by AMOSA, MODE-BE and NSGA-II algorithms in materialized view selection for data warehousing using our data sets are presented in Table 7.

7.2 Observations

The Convergence, Purity and Minimal Spacing, obtained by AMOSA, MODE-BE and NSGA-II algorithms in materialized view selection, presented in Tables 5, 6 and 7 respectively, show acceptable values by all these three techniques. However the average Convergence (γ) of MODE-BE is found to be the best. Though in one case AMOSA produces excellent Purity value, in case of other three data sets this measure is found to be poorer than the other two methods. As smaller Minimal Spacing, S_m, reflects that the solutions are uniformly distributed over the known Pareto-optimal front, it has been observed that the MODE-BE yields the most uniformly distributed solutions compared to the other two algorithms (Table 7). 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, 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. Randomized function based generation of solutions by perturbing values of dimensions of solution vectors from randomly selected solution vector (like in case of AMOSA) may produce distant solutions in objective function space. Overall it has been observed that the MODE-BE algorithm converges very well empirically in this application and consistently well performing among these three techniques in our experimental setup.

8 Conclusion and perspectives

We have presented here our study on view selection techniques for materializing in distributed commodity hardware file system data warehousing. Our main contribution here is

establishing the view selection for materializing problem in distributed commodity hardware file system as a multi-objective optimization problem and providing solution for solving this problem. As it is an NP-hard problem, we used multi-objective evolutionary algorithm based approach for designing a recommendation system for selecting materialized views.

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

Though, so far in our 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 generation of database from log-files produced by HiveQL query processing in HDFS is yet to be integrated to the system. Therefore, these are the future extensions of our work.

The present popular version of Hive^{TM} does not support materialized views. Our present work leads towards selecting views for materializing in cluster of distributed commodity hardware file system based data warehousing by using multi-objective DE and NSGA-II. 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) (Hyde 2014) etc. We believe that community like Hadoop may find many more such technologies and bench-mark framework for unbiased evaluation of them.

Compliance with Ethical Standards

Disclosure of potential conflicts of interest The authors declare that they have no conflict of interest.

Informed Consent Not applicable.

Research involving Human Participants and/or Animals The authors declare that, in this research, neither human participants nor animals are involved in experimentation.

References

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

Aouiche, K., Jouve, P., & Darmont, J. (2006). Clustering-based materialized view selection in data warehouses. Y. Manolopoulos, J. Pokorn & T. Sellis (Eds.), *Proceeding of 10th east-European conference advances in database and information systems, ADBIS 2006, LNCS*, (Vol. 4152, pp. 81–95). Springer-Verlag, Berlin, Heidelberg, Thessaloniki: Hellas.

Bandyopadhyay, S., Pal, S., & Aruna, B. (2004). A simulated annealing-based multiobjective optimization algorithm: Amosa. *IEEE Transactions on Systems Man and Cybernetics Part B*, *34*(5), 2088–2099.

Bandyopadhyay, S., Saha, S., Maulik, U., & Deb, K. (2008). A simulated annealing-based multiobjective optimization algorithm: Amosa. *IEEE Transactions on Evolutionary Computation*, *12*(3), 269–283.

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

Dean, J., & Ghemawat, S. (2004). Mapreduce: Simplified data processing on large clusters. In *OSDI'04: 6th symposium on operating system design and implementation, San Francisco, CA, December*. https://www.usenix.org/legacy/publications/library/proceedings/osdi04/tech/fullpapers/dean/dean_html/index.html.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2001). *Scalable test problems for evolutionary multi-objective optimization. Tech. rep., Institute fur Technische Informatik und Kommunikationsetze*. Switzerland: Zurich.

Derakhshan, R., Dehne, F., Korn, O., & Stantic, B. (2006). Simulated annealing for materialized view selection in data warehousing environment. In *Proceedings of 24th IASTED international conference on Database and applications* (pp. 89–94). Austria: Innsbruck.

Derakhshan, R., Stantic, B., Korn, O., & Dehne, F. (2008). Parallel simulated annealing for materialized view selection in data warehousing environments. In *Proceedings of ICA3PP 2008 international conference on algorithms and architecture 2008, LNCS* (Vol. 5022, pp. 121–132). Berlin: Springer.

Foundation, T.A.S. (2014). Apache hadoop. URL http://hadoop.apache.org.

Foundation, T.A.S. (2014). Apache hive tm. URL http://hive.apache.org.

Gong, T., & Tuson, A. (2006). Differential evolution for binary encoding. In *11th online world conference on soft computing in industrial applications (WSC11)*.

Goswami, R., Bhattacharyya, D., Dutta, M., & Kalita, J. (2016). Approaches and issues in view selection for materializing in data warehouse. *International Journal of Business Information Systems*, *21*(1), 17–47.

Goswami, R., Bhattacharyya, D.K., & Dutta, M. (2012). Selection of views for materializing in data warehouse using mosa and amosa. In D. C. Wyld, J. Zizka & D. Nagamalai (Eds.) *Advances in computer science, engineering and applications, Proceedings of the 2nd international conference on computer science, engineering and applications (ICCSEA 2012), Volume 1, advances in intelligent and soft computing* (Vol. 166, pp. 619–628). Berlin: Springer.

Goswami, R., Bhattacharyya, D.K., & Dutta, M. (2013). Multiobjective differential evolution algorithm using binary encoded data in selecting views for materializing in data warehouse In B. K. Panigrahi, P. N. Suganthan, S. Das & S. S. Dash (Eds.). *Swarm, evolutionary, and memetic computing, lecture notes in computer science* (Vol. 8298, pp. 95–106). Springer.

Gupta, H., Harinarayan, V., Rajaraman, A., & Ullman, J. (1997). Index selection for olap. In *Proceedings of the 13th international conference on data engineering, ICDE'97* (pp. 208–219). Washington: IEEE Computer Society.

Gupta, H., & Mumick, I. (1999). Selection of views to materialize under a maintenance cost constraint In C. Beeri & P. Bruneman (Eds.) *Proceedings of international conference on database theory, ICDT 1999, LNCS* (Vol. 1540, pp. 453–470). Heidelberg: Springer.

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

Hagleitner, G. (2014). Cost-based optimization in hive. URL https://cwiki.apache.org/confluence/display/Hive/Cost-based+optimization+in+Hive.

Harinarayan, V., Rajaraman, A., & Ullman, J. (1996). Implementing data cubes efficiently. In *Proceedings of ACM SIGMOD international conference on management of data* (pp. 205–216). Montreal: ACM SIGMOD.

Hyde, J. (2014). Discardable in-memory, materialized query for hadoop. URL http://hadoopsummit.org/san-jose/schedule.

Inc., H. (2014). Hortonworks data platform. URL http://hortonworks.com/hdp/.

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

Kukkonen, S., & Lampinen, J. (2005). Gde3: The third evolution step of generalized differential evolution. In *The 2005 IEEE congress on evolutionary computation, 2005* (Vol. 1, pp. 443–450). IEEE.

Lahman, S. (2014). Baseball archive. URL http://seanlahman.com/files/database/lahman591-csv.zip.

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

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

Loureiro, J., & Belo, O. (2006). An evolutionary approach to the selection and allocation of distributed cubes. In *Proceedings of database engineering and applications symposium IDEAS-06* (pp. 243–248). Delhi: IEEE.

Madavan, N.K. (2002). Multiobjective optimization using a pareto differential evolution approach. In *Proceedings of the 2002 congress on evolutionary computation, 2002. CEC'02* (Vol. 2, pp. 1145–1150). IEEE.

Nadeua, T., & Teorey, T. (2002). Achieving scalability in olap materialized view selection. In *Proceedings of ACM 5th international workshop on data warehousing and OLAP, DOLAP-02* (pp. 28–34). McLean: ACM.

Qingzhou, Z., Xia, S., & Ziqiang, W. (2009). An efficient ma-based materialized views selection algorithm. In *Proceedings of the 2009 IITA international conference on control, automation and systems engineering* (pp. 315–318). China: Zhangjiajie.

Radcliffe, N.J. (1991). *Equivalenc class analysis of genetic algorithms*. Complex Systems pp. 183–205.

Radcliffe, N.J. (1991). Forma analysis and random respectful recombination. In *Proceedings of the international conference on genetic algorithms - ICGA 1991* (pp. 222–229). San Marco: Morgan Kaufmann.

Roy, T., & Fielding Laguna Beach, C. (1999). Certificate of incorporation of the apache software foundation. URL http://www.apache.org/foundation/records/certificate.html.

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

Serna-Encinas, M.T., & Hoya-Montano, J.A. (2007). Algorithm for selection of materialized views: based on a costs model. In *Proceedings of 8th international conference on current trends in computer science* (pp. 18–24). Morella, Maxico. doi:10.1109/ENC.2007.38.

Smith, J.R., Li, C.S., & Jhingran, A. (2004). A wavelet framework for adapting data cube views for olap. *IEEE Transactions on Knowledge and Data Engineering*, *16*(5), 552–565.

Smith, K.I., Everson, R.M., Fieldsend, J.E., Murphy, C., & Misra, R. (2008). Dominance-based multiobjective simulated annealing. *IEEE Transactions on Evolutionary Computation*, *12*(3), 323–283.

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

Song, X., & Gao, L. (2010). An ant colony based algorithm for optimal selection of materialized view. In *2010 international conference on intelligent computing and integrated systems (ICISS)* (pp. 534–536). IEEE.

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

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

Sun, X., & Wang, Z. (2009). An efficient materialized views selection algorithm based on pso. In *Proceedings of international workshop on intelligent systems and applications 2009* (pp. 1–4). China: Wuhan.

Tusar, T., & Filipic, B. (2007). Differential evolution versus genetic algorithms in multiobjective optimization. In *Proceedings of the 4th international conference on evolutionary multi-criterion optimization, LNCS*, (Vol. 4403, pp. 257–271). Springer-Verlag.

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

Xue, F., Sanderson, A.C., & Graves, R.J. (2003). Pareto-based multi-objective differential evolution. In *The 2003 congress on evolutionary computation, 2003. CEC'03* (Vol. 2, pp. 862–869). IEEE.

Yang, J., Karlapalem, K., & Li, Q. (1997). Algorithm for materialized view design in data warehousing environment. In *Proceedings of VLDB 1997* (pp. 136–145). Greece: Athens.

Zhang, C., & Yang, J. (1999). Genetic algorithm for materialized view selection in data warehouse environments. In M. Mohania & A. M. Tjoa (Eds.) *Proceedings of data warehousing and knowledge discovery, 1st international conference, DaWak 1999, LNCS* (Vol. 1676, pp. 116–125).

Zhang, C., Yao, X., & Yang, J. (2001). 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.