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

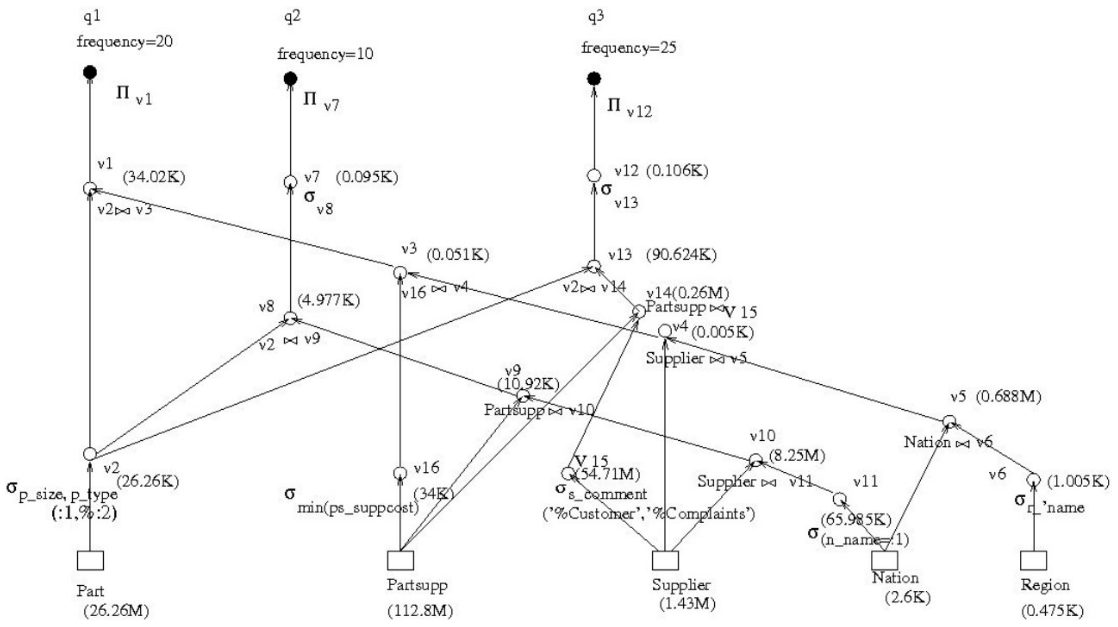

Figure 3-1: An example MVPP graph using TPC-H benchmark data warehouse

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.

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

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

If $S_0 \nprec S_1$ and $S_1 \nprec S_0$, where $\nprec$ 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 Ψ =$\{\psi_1, \psi_2, \cdots, \psi_D$ }, 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
 1: Generate $NP$ random vectors $x_1, x_2, \cdots, x_{NP}$ of dimension $D$ that satisfy
    the specified constraints
 2: $N \leftarrow NP$
 3: $g \leftarrow 1$
 4: **repeat**
 5:    **for** $i = 1$ to $N$ **do**
 6:       select $x_i$ and $x_{r_1}, x_{r_2}, x_{r_3}$, such that $x_i \neq x_{r_1} \neq x_{r_2} \neq x_{r_3}$
 7:       **for** $j = 1$ to $D$ **do**
 8:          $v_{j,i} \leftarrow D_{\Psi_j}(x_{r_1}, round(F.D_{\Psi_j}(x_{r_2}, x_{r_3})))$
 9: where $round(F.D_{\Psi_j}(x_{r_2,g}, x_{r_3,g})) =$
    $\begin{cases} 1, & \text{if random}[0,1] < F \wedge (D_{\Psi_j}(x_{r_2,g}, x_{r_3,g}) = 1) \\ 0, & \text{otherwise} \\ D_{\Psi_j}(x_{r_2}, x_{r_3}) = \begin{cases} 0, & \text{if } x_{j,r_2} = x_{j,r_3} \\ 1, & \text{otherwise} \end{cases} & \end{cases}$
10:          $j \leftarrow j + 1$
11:       **end for**
12:       $rand_i = random[1, D]$
13:       **for** $j = 1$ to $D$ **do**
14:          $u_{j,i} \leftarrow \begin{cases} v_{j,i}, & \text{if}(random[0,1] \leq CR) \text{ or } j = rand_i \\ x_{j,i}, & \text{otherwise} \end{cases}$
15:          $j \leftarrow j + 1$
16:       **end for**
17:       **if** $u_i$ satisfies the constraints **then**
18:          Evaluate query processing cost and materialized view maintenance
    cost of $G$ for $u_i$, $x_i$
19:          **if** $u_i \prec x_i$ **then**
20:             $x_i \leftarrow u_i$
21:          **else**
22:             **if** $x_i \nprec u_i$ **then**
23:                $NP \leftarrow NP + 1$
24:                Append $u_i$ to the population
25:             **end if**
26:          **end if**
27:       **end if**
28:       $i \leftarrow i + 1$
29:    **end for**
30:    **if** $NP > N$ **then**
31:       Keep the elite $N$ members from $NP$ population in the list and discard
    the rest
32:       $NP \leftarrow N$
33:    **end if**
34:    $g \leftarrow g + 1$
35: **until** $g < g_{max}$
36: **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 particular *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 $\{X_k \colon k \geq 0\}$ 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 :

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

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 $\hat{i} = (i^e; i) = (i^e_1, i^e_2, \cdots, i^e_r; i_1, i_2, \cdots, i_n)$, where $i^e_1, i^e_2, \cdots, i^e_r$ 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^e_k and therefore if $X_k = i$, we may express $X^e_k = i^e$.

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 $i \rightarrow j$, 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 -

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

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 $i^e_{s1}, \cdots, i^e_{sk}$ 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 $j^e_{s1}, \cdots, j^e_{sk}$ 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 $\hat{i} \rightarrow \hat{j}$. 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, $\mathbb{P}(\{X^e_k\} \subset \mathscr{P}) = \mathbb{P}(X_k \in E) = 1 - \mathbb{P}(X_k \in I) \rightarrow 1 - 0 = 1$ as $k \rightarrow \infty$. 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,

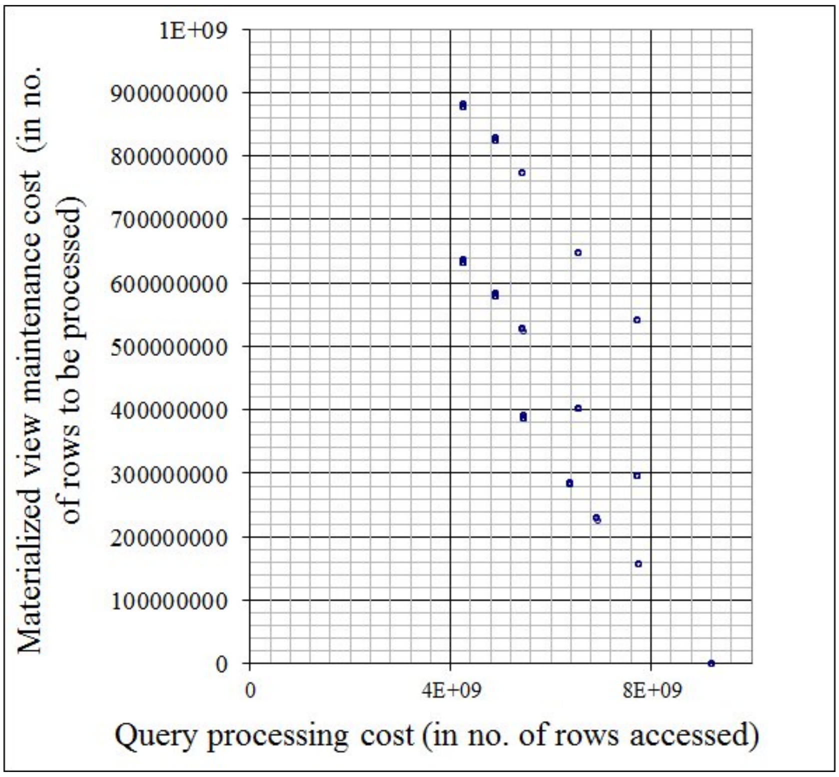

Figure 3-2: Randomly generated 242 solutions

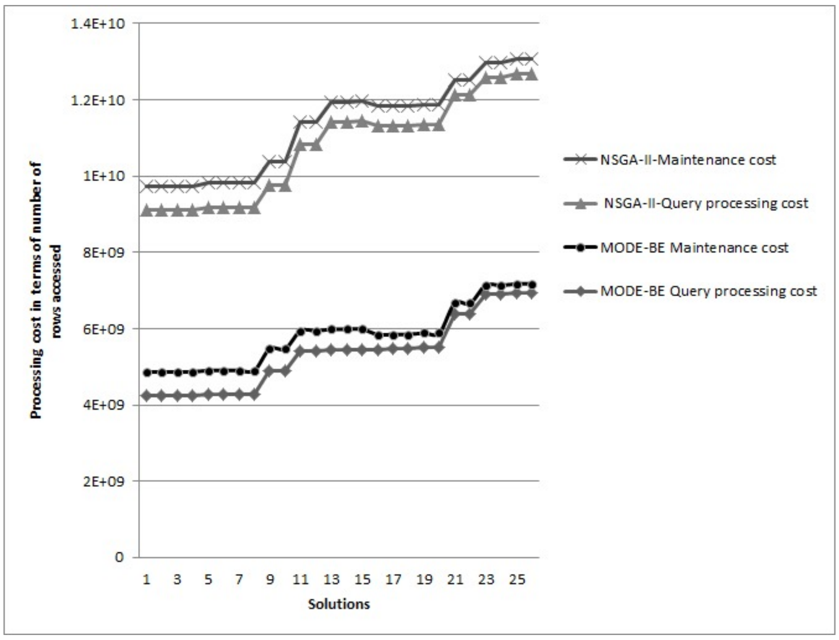

Figure 3-3: Distribution of costs by obtained non-dominated solutions after 40 iterations

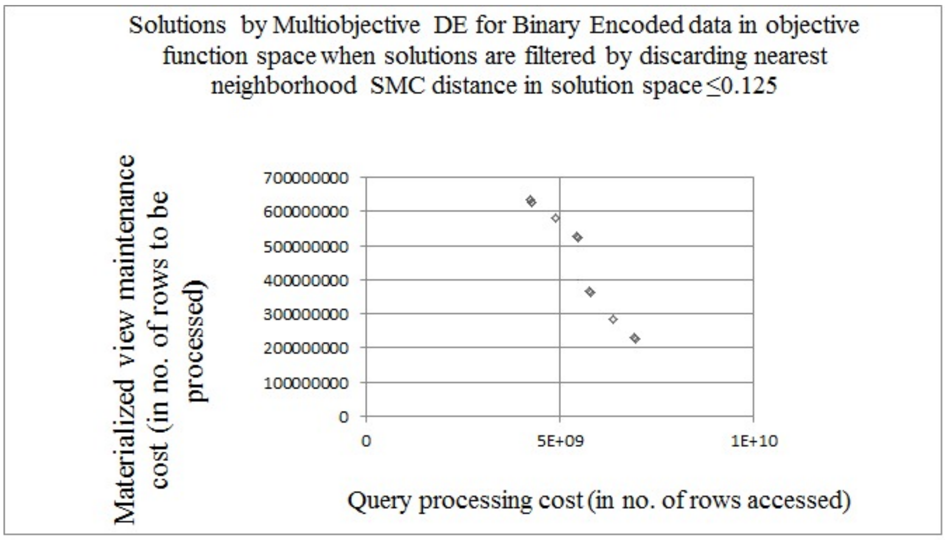

Figure 3-4: Distribution of significant representative solutions in objective space

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

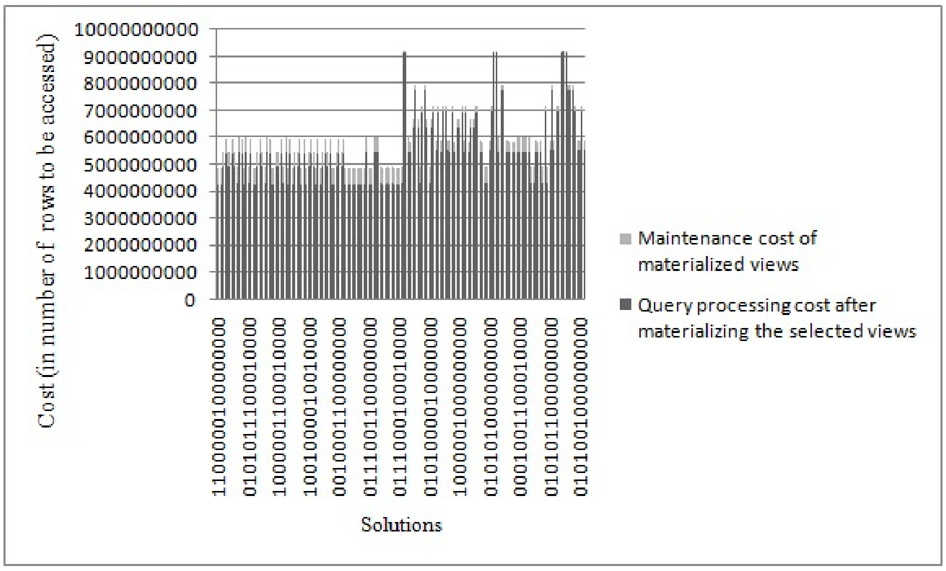

Figure 3-5: Objective function values of non-dominating solutions

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	