Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 54 (2015) 281 – 290

Procedia Computer Science

Eleventh International Multi-Conference on Information Processing-2015 (IMCIP-2015)

Generating Optimal Query Plans for Distributed Query Processing using Teacher-Learner Based Optimization

Vikash Mishra and Vikram Singh*

National Institute of Technology, Kurukshetra, Haryana 136 119, India

Abstract

Modern day's queries are posed on database spread across the globe, this may impose a challenge on processing queries efficiently, and a strategy is required to generate optimal query plans. In distributed relational database systems, due to partitioning or replication on relations at multiple sites, the relations required by a query to answer, may be stored at multiple sites. This leads to an exponential increase in the number of possible equivalent alternatives or query plans for a user query. Though it is not computationally reasonable to explore exhaustively all possible query plans in a large search space, the query plan with most cost-effective option for query processing is measured necessary and must be generated for a given query. In this paper, an attempt has been made to generate such optimal query plans using parameter less optimization technique Teaching-Learner based Optimization (TLBO). The TLBO algorithm was observed to go one better than the other optimization algorithms for the multi-objective unconstrained and constrained benchmark problems. Experimental comparisons of this algorithm with the multi-objective GA based distributed query plan generation algorithm shows that for higher number of relations, the TLBO based algorithm is able to generate comparatively better quality Top-K query plans.

Peer-review under responsibility of organizing committee of the Eleventh International Multi-Conference on Information Processing-2015 (IMCIP-2015)

Keywords: Aggregation based genetic algorithm; Distributed query processing; Top-K; Teacher learner based optimization; VEGA.

1. Introduction

Database systems of different type use various techniques to identify optimal query plans. In many application domains, end-users are more interested in the most important (top-k) query answers in potentially huge answer space[1]. A distributed database encompasses coherent data, spread across various sites of a computer network[22]. A Distributed Database Management System (DDBMS) deals with managing such distributed databases. DDBMS provides access to user via a simple and unified interface over disparate databases, as if they were not distributed[23]. The performance of a DDBMS is determined by its ability to process queries in an effective and efficient manner[18]. The query processing is more complicated, as there are various parameters affecting their performance[1]. The data relevant to query is usually available at different sites, the query processing, thus, would involve transmission of data between these sites. These data transmissions, along with local data processing, constitute a distributed query processing (DQP) strategy for a user query[6].

*Corresponding author. Tel.: +918816086589.
E-mail address: viks@nitkkr.ac.in

Peer-review under responsibility of organizing committee of the Eleventh International Multi-Conference on Information Processing-2015 (IMCIP-2015)
doi:10.1016/j.procs.2015.06.033

Query processing connects to many database research areas, including query optimization, indexing methods, and query languages[6,7]. As a consequence, the impact of efficient processing of query is becoming apparent in an increasing number of applications. One common way to identify the top-k objects is scoring all database objects on some scoring function. An object score acts as a valuation for that object according to its characteristics (e.g., price and size of house objects in a real estate database, or color and texture of images in a multimedia database). Query plans are usually evaluated by multiple scoring predicates or objective functions that contribute to the total object score. A scoring function is therefore usually defined as an aggregation over objectives or scores[26].

In DQP[10], the distributed query is parsed before arriving at an effective query processing strategy for it. This strategy comprises of effective and efficient query processing plans that would decompose the distributed queries into local sub-queries to be executed at their respective sites. Also, the order and the site at which the results of the sub-queries are integrated is also part of this plan. The finally integrated result is provided as the answer of the query. Thus the DQP strategy aims to generate query processing plans that reduce the amount of data transfer between sites and thereby reduces the distributed query response time[3,23]. This paper focuses on generating optimal query processing plans for distributed relational queries.

In DDB, data is dispersed over the multiple sites, distribution policy decides the manner of distribution of data, Full replication or Partition etc. due to which a given relation can be found in more than one sites[5], This distribution of data imposed a challenges to query processing, as for a query there are multiple query equivalent plans (QEP's) possible, these alternatives are equivalent in the result retrieve. Selecting best alternatives for processing from the generated pool is a critical task to query optimizers, whose primary objective is to choose optimal (best) solutions[22]. In optimization, an alternative is considered better or fitter than other based on the objective function values for the query result generation. In DDB environment, query processing is mainly influenced by different cost involved, e.g. communication cost, local processing cost, optimization cost, query localization cost etc.

In the proposed heuristic, optimality on query plans is based on the function of different cost models, cost models are assumed on attributes of query processing. Each of the query equivalent plans has set of pre-computed cost values, based on which optimization is performed. Computation of cost is according to the different primitives proposed, discussed in next section. In many real -world problems optimization on two or more objective functions simultaneously is desirable. These problems are known as multi-objective optimization problems (MOPs), and solution involves finding not one, but a set of solutions that represent the best possible trade-offs among the objective functions being optimized. Such trade-offs constitute the so-called Pareto optimal set, and their corresponding objective function values form the so-called Pareto front[25].

Optimization computation in most of the evolutionary and swarm intelligence-based algorithms are probabilistic and requires controlling parameters, like the search space size, number of generations, elite size, etc. In addition to the common control parameters, algorithm-specific control-parameters are required, eg. In GA rate of mutation and crossover rate, similarly, inertia weight and social parameters in PSO. The proper regulation of algorithm explicit parameters is a very crucial factor, as it affects the overall performance of the above-mentioned algorithms. The improper regulation of algorithm-specific parameters may lead to increases the computational effort or yields a localized solution. Therefore, recently introduced the teaching–learning-based optimization (TLBO) algorithm, which requires only the common control parameters and does not require any algorithm-specific control parameters[15,20,21]. Other evolutionary algorithms require the control of common control parameters as well as the control of algorithm-specific parameters. The burden of tuning control parameters is comparatively less in the TLBO, thus the TLBO is simple, effective and involves less computational effort. Hence, in the present work, TLBO is used to multi-objective unconstrained test functions, and performance is compared with other nature-inspired optimization algorithms such as Vector Evaluated Genetic Algorithm (VEGA) and Aggregation based Genetic Algorithm.

2. Related Work and Contribution

The processing of distributed query plans leads to a problem of an exhaustive search and not computationally feasible[6]. Further, this being a combinatorial optimization problem[9], it can be addressed by techniques based on heuristics like greedy, evolutionary, and randomized[2,4–6,8,13]. However, efficiency of these techniques is affected by the unconventional behaviour, in specific instances, of the problem[13]. In[25], an approach that generates "close" query

plans with respect to the number of sites involved and the concentration of relations in the sites for a distributed relational query is given. As per[25,26], query processing over lesser number of sites would be more efficient and thus query plans involving fewer sites need to be generated. Such query plans, referred to as "close" query plans, are generated using the genetic algorithm (GA)[25,26], without considering the communication and local processing cost on optimization of QEP's. None of the existing approach considered the fundamental cost models for optimization.

The main contribution of the current work is to demonstrate the use of parameter-less optimization (other than nature-inspired optimization technique) for query optimization for distributed query processing and analysis of proposed cost functions for DQP. In section 3, proposed design objectives for 'Optimal Query' generation are discussed with an example. In section 4, fundamental of TLBO are discussed and algorithm of the same for query optimization is presented. Section 5, experimental results of performance of Aggregation based GA, VEGA and TLBO on 'Optimal Top-K Query' generation are shown via various graphs.

3. Distributed Query Processing

In DQP, the major costs incurred are CPU, I/O and the site-to-site communication cost. Among these, the intra-site communication cost is the dominant cost. This cost can be reduced if fewer sites are involved in processing a distributed query. In order to process a distributed query, the data required may have to be obtained from several sites distributed over a computer network. Furthermore, as the number of sites containing the relations accessed by the query increase, the number of possible valid query plans also increases. So it becomes imperative to arrive at a query processing plan that entails an optimal cost for query processing. However, the number of such possible query plans increases exponentially with increase in the number of relations in the query and also with increase in the number of sites containing them[10]. Thus, a large search space comprising all possible query plans needs to be explored in order to compute the optimal query plans. Query processing in such environment is difficult task for query processor, as for a user query there are multiple query equivalent alternatives possible, these alternatives are equivalent on the output terms. Selecting best alternatives for processing from the generated pool of alternatives is a critical task to query optimizers, whose primary objective is to choose optimal (best) solutions[22].

3.1 Heuristic's of design objective

(Objective-1) Query Affinity Cost (QAC): This criterion indicates the degree of heterogeneity on the number of sites involved in result generation for given query plan. The query plan that involves the least number of sites is consider better than the other alternatives. If there are more than one such plan then the plan having sites with higher concentration of relations is considered better. The QAC value of query plan is computed as follows,

$$QAC = \sum_{i=1}^{M} \frac{Ki}{N}\left(1 - \frac{Ki}{N}\right) \tag{1}$$

where M is the number of sites accessed by the query plan, K_i is the number of times the i^{th} site is used in the query plan; N is the number of relation accessed by the query.

(Objective-2) Query Localization Cost (QLC): The 'Query Localization' indicates two design objectives, first it quantify the degree of communication between two different sites in a query plans (QP) and second, it plays crucial role on deciding the control site for query answering of respective QP. Evaluation of the communication cost for QP, Relation-Site Matrix (RSM) and Relation-Size vector (number of tuples), An QP is mapped in a query graph on which sites as nodes and edges as join selectivity (QLC values) are mapped. Computing join selectivity for a QP on dynamic and is distributed environment is biggest challenge. QLC between two sites S_1 to S_2 represents the ratio of size of relation at that site and sum of sizes of total relations in FROM clause of query. For a given query plan the minimum QLC can is computed as follows,

$$QP_{\text{Cost}} = MIN_{i=1}^{Nr}\left[\sum_{j=1 \&\& j\neq i}^{Nr} \frac{\text{Size}(R_{S_j})}{\sum_{k=1}^{Nr}\text{Size}(R_k)}\right] \tag{2}$$

where, QP_{cost}-Communication/Localization cost of given Query Plan, MIN is a function to evaluate minimum value for $i = (1 \text{ to } N_r)$, N_r Number of relations in given RSM or number of relations in FROM clause in query, Size $(R_s j)$ is number of tuples in relation present at site S_j, Size (R_k) is number of tuples in relation R_k. QLC between sites S_i and S_i will be zero.

(Objective-3) Local Processing Cost (LPC): Local processing primarily concerns by relations stored on a local site and selectivity of operators on local relations, eg. Selection, Projection etc., quantify the value of LPC of a QP. Generally LPC is dependent on number of memory accesses or memory fetch for transferring set of tuples from secondary memory to main memory, while retrieving data for local relations. Evaluation of LPC is dependent on two sub-computations, first due to LPC at remote sites and second LPC on control sites (for final result preparation). LPC evaluation is based on following formulas: (assumptions) N_t is number of tuples, N_r is number of Relations, S_{qp} is total number of sites in the QP, R_s is total number of relations on local site, S_r is selectivity of relation R on local site, S_j is selectivity of join and N_j is number of joins for a query plan.

Relation Processing Cost-

$$RPC = N_t * S_r / \sum_{k=1}^{Nr} Nt(k) \tag{3}$$

(a) LPC for Remote Site used in Query Plan

$$RLPC = Max_{i=1 \text{ to } Rs}[RPC(i)] \tag{3a}$$

(b) LPC for Control Site used in Query Plan

$$CLPC = Max_{i,j=1 \text{ to } Nr}[N_t(JOIN(R)_i, R_j) * S_j((JOIN(R)_i, R_j))/ \sum_{k=1}^{Nr} Nt(k)] \tag{3b}$$

LPC of a query plan QP_i

$$(3.a + 3.b) = \sum_{k=1}^{sqp-1} (RLPC(k)) + (CLPC - Max_{i=1 \text{ to } Nr\&R(i)\varepsilon(CS\&JOIN)}[RPC(i)].$$

Example 1. *In, distributed database systems relations are spread across multiple sites, Table 1(a) depicts a typical scenario of DDBS. Relation-Site matrix gives details about sites and stored relation in it, eg. Relation R1 is stored in S1, S2, S3, S5, S6, S8, S10 and it is assumed that R1 is replicated in these sites for keeping reliability and availability degree high. Once user or application posed query on the DDBS, it is critical to identify the relevant sites for retrieval of results and based on the RSM a query optimizer identify the sites on which particular relation store. For a user query, as shown below, multiple alternatives are possible as a relation is stored in multiple sites and all the alternatives are initialized based on the RSM, Table 1(b) shows 'a subset of possible alternatives' for below mentioned user query.*

Q1: SELECT a, m
FROM R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8,
WHERE $R_1.a = R_4.t$**AND**$R_4.p = R_2.x$**AND**$R_1.a_= R_7.q$
AND $R_2.x = R_3.n$**AND**$R_4.x = R_5.s$**AND**$R_8.w = R_6.d$**AND**$R_7.j = R_6.k$

There are two important aspects namely the content of the query plans and the length of the query plans. The length of the query plan is computed as the number of relations in the FROM clause of the user query. The content is the sites of these relations. A query plans represent the set of data sites on which relevant data is stored and result will be aggregated, eg. In query plans [1,1,2,2,2,3,5,3], relation R_1 & R_2 from S_1, relation R_3, R_4 & R_5, from S_2, R_6 & R_8 from S_3 and finally relation R_7 from S_5. Each initialized QP is associated with cost values or design objective. Which are computed based on the heuristics proposed as in section 3.1.

Table 1. (a) Relation-site matrix (RSM); (b) Subset of query plans (QP's) of Q1.

Site / Relation	R1	R2	R3	R4	R5	R6	R7	R8
S1	1	1	1	1	1	0	0	0
S2	1	1	1	1	1	0	0	0
S3	1	0	0	0	0	1	0	1
S4	0	0	0	0	0	1	0	0
S5	1	1	1	0	0	1	1	0
S6	1	0	0	0	0	1	1	0
S7	0	1	1	1	1	1	0	0
S8	1	1	1	1	0	0	1	1
S9	0	0	0	0	0	0	0	1
S10	1	0	1	0	0	0	0	0
S11	0	1	1	1	0	0	1	0
S12	0	1	0	0	0	0	0	1
S13	0	0	1	0	0	0	0	0
S14	0	0	0	1	0	0	0	1
S15	1	0	1	1	1	0	0	0
S16	0	1	1	0	1	1	1	0

Population(Query Plan) $[R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8]$	
1	[1,1,2,2,2,3,5,3]
2	[3,5,7,15,4,6,8]
3	[6,7,8,11,16,6,8,9]
4	[10,11,11,15,16,11,16,14]
5	[8,8,10,14,16,11,11,14]
6	[15,12,13,11,15,15,11,12]
7	[1,2,5,7,2,4,6,8]
8	[3,5,7,8,15,3,5,3]
9	[2,2,2,2,3,5,3]
10	[2,1,113,2,15,3,5,3]
11	[8,8,16,14,16,15,16,14]
12	[3,7,7,8,16,7,16,3]
13	[2,2,2,2,15,16,14]
14	[1,1,8,8,2,7,8,8]
15	[8,8,8,2,7,8,9]
16	[5,16,15,15,16,6,8,9]
17	[1,1,1,1,15,15,16,14]
18	[10,11,11,8,7,3,5,3]
19	[15,16,15,15,15,16,14]
20	[1,1,8,8,1,7,8,8]

4. Teacher Learner Based Optimization (TLBO)

Teacher-Learner-Based-Optimization algorithm is a teaching–learning process inspired algorithm recently proposed in[19–21] and based on the effect of influence of a teacher on the output of learners or students in a class. This algorithm consider a group of learners as population and different subjects offered to the learners are considered as different design objective and a learner's result is analogous to the 'fitness' value of the optimization problem. The best solution in the entire population is considered as the teacher. The design objectives are actually the parameters involved in the objective function of the given optimization problem and the best solution is the best value of the objective function. The working of TLBO algorithm is alienated into two phases, 'Teacher phase' and 'Learner phase', working of both phases is described in detail[19,20]. The multi-objective unconstrained and constrained test functions in this paper, and the results were compared with other optimization algorithms[21].

All evolutionary and optimization methods requires optimization computation, most of the evolutionary and swarm intelligence-based algorithms this is computation probabilistic and requires controlling parameters, like the search space size, number of generations, elite size, etc. In addition to the common control parameters, some algorithm requires tuning of algorithm-specific control-parameters required for better optimization, eg. In GA rate of mutation and crossover rate, similarly, inertia weight and social parameters in PSO. The improper regulation of algorithm-specific parameters may lead to increases in computational effort or yields a localized solution. Recently, an optimization algorithm is introduced named teaching–learning-based optimization (TLBO) algorithm, which requires only the common control parameters and does not require any algorithm-specific control parameters[15,20,21]. The burden of tuning control parameters is comparatively less in the TLBO algorithm.

Thus, the TLBO algorithm is simple, effective and involves comparatively less computational effort. Hence, TLBO is used. The TLBO algorithm has been already tested on several constrained and unconstrained benchmark functions and proved better than the other advanced optimization techniques[15]. It is also proving better in various field of engineering such as those reported[15,16] in the field of electrical engineering,[24] in the field of civil engineering. Similarly, used it for various fields related to manufacturing processes. Even though[14] raised some doubts about the algorithm-specific parameter less concept of TLBO algorithm and some other issues, however,[21] had already cleared all those issues and justified that the TLBO algorithm is an algorithm-specific parameter less algorithm.

In the literature, it is observed that, the TLBO algorithm is not yet used in the field of generation of 'Optimal Query' plans for distributed query processing. In the domain of database, hence the same is now used for the parameters optimization of query plans under consideration, as shown in Fig. 1.

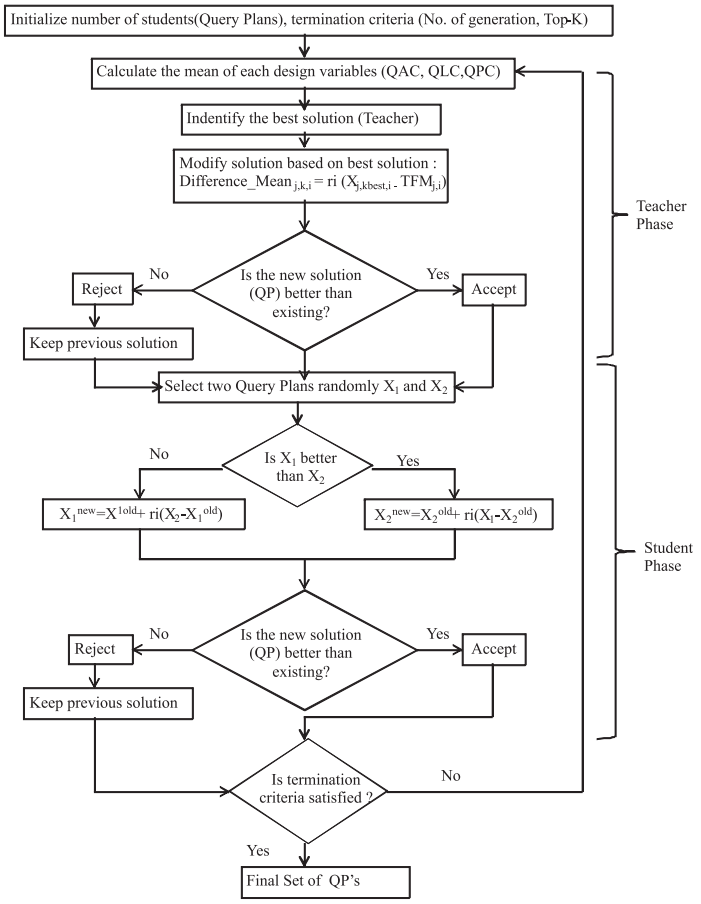

Fig. 1. Flowchart of TLBO for 'Optimal Query' plan generation.

TLBO, starts by initializing the entire set of query plans for given user or application query and using pre-determined Relation-Site Matrix (RMS), these query plans in solution space are equivalent to student or learners of TLBO. A Decision maker or query optimizer provides values of termination criteria's for TLBO during the initialization phase, eg. No. of Generation, values of K for Top-K query plans, Average Query Cost etc. These parameters are important as performance comparison among various optimization techniques. In the teacher phase of algorithm, students or learners learn via the teacher, a teacher attempts to increase the mean result of the class in the subject a teaches depending on his or her capability, different design objective are analogue is to the different subjects taught by teacher and the respective subject values represent the score of particular student or learner in the subject. The best overall result, best considering all the subjects together obtained in the entire population of learners can be considered the result of the best learner[20]. However, since the teacher is usually considered a highly learned person who trains learners so that they can have better results, the algorithm considers the best identified learner to be the teacher[21].

The identification of best learner's based on the fitness value of learner, fitness value is evaluated by benchmark function, learner with minimum fitness values considered best in set, shown in Table 2(d) query plan no. 15 is best among learners. Now difference means or teacher factor (T_F), which decides the degree of change on each students or learner, is the difference between the existing mean results of each subject and the corresponding result of the teacher for each subject, as shown in Table 2(c) and values of (T_F) either 1 or 2, as $T_F = \text{round}\,[1 + \text{rand}(0, 1), \{2 - 1\}]$. Based on the T_F, new population as shown in Table 2(a) and 2(b) are evolved in which the each of the subjects values (objective function) are updated based on T_F on each criteria, also new population is shown in Table 2(e).

Table 2. Teacher-phase related; (a) Initial population and costs; (b) Mean values of design objectives; (c) Teaching factor or difference factor (TF); (d) Best teacher query plan; (e) Updated population based on T_F.

Teacher Phase

Initial Population(Query Plan)		Costs			$Fitness_1$
QP No.	Query Plan	QAC	QLC	LPC	
1	[1,1,2,2,2,3,5,3]	0.7188	0.5354	0.2667	0.8744
2	[3,5,7,15,4,6,8]	0.8438	0.6691	0.3466	1.2797
3	[6,7,8,11,16,6,8,9]	0.8125	0.5374	0.3878	1.0994
4	[10,11,11,15,16,11,16,14]	0.7500	0.4315	0.3853	0.8971
5	[8,8,10,14,16,11,11,14]	0.7813	0.6691	0.3724	1.1967
6	[15,12,13,11,15,15,11,12]	0.7188	0.6691	0.2615	1.0327
7	[1,2,5,7,2,4,6,8]	0.8438	0.6691	0.4651	1.3760
8	[3,5,7,8,15,3,5,3]	0.7500	0.6691	0.3337	1.1215
9	[2,2,2,2,3,5,3]	0.5313	0.3130	0.1430	0.4007
10	[2,1,113,2,15,3,5,3]	0.8125	0.6691	0.3517	1.2316
11	[8,8,16,14,16,15,16,14]	0.7188	0.5215	0.2899	0.8726
12	[3,7,7,8,16,7,16,3]	0.7188	0.4315	0.3054	0.7960
13	[2,2,2,2,15,16,14]	0.5625	0.3130	0.1945	0.4522
14	[1,1,8,8,2,7,8,8]	0.6563	0.3091	0.2925	0.6117
15	[8,8,8,2,7,8,9]	0.4063	0.0867	0.2100	0.2167
16	[5,16,15,15,16,6,8,9]	0.8125	0.5513	0.4007	1.1247
17	[1,1,1,1,15,15,16,14]	0.6563	0.3289	0.2667	0.6100
18	[10,11,11,8,7,3,5,3]	0.8125	0.5023	0.4136	1.0835
19	[15,16,15,15,15,16,14]	0.5313	0.4090	0.1430	0.4700
20	[1,1,8,8,1,7,8,8]	0.5938	0.3091	0.1688	0.4765

Mean Value	
QAC	0.7015
QLC	0.4797
LPC	0.2999

Diff. Factor(T_F)	
QAC	-0.5906
QLC	-0.7860
LPC	-0.1799

Best Query Plan
15

QP No.	Updated Population_01		
	QAC	QLC	LPC
1	0.1281	-0.2506	0.0868
2	0.2531	-0.1169	0.1667
3	0.2219	-0.2486	0.2079
4	0.1594	-0.3545	0.2054
5	0.1906	-0.1169	0.1925
6	0.1281	-0.1169	0.0816
7	0.2531	-0.1169	0.2853
8	0.1594	-0.1169	0.1538
9	-0.0594	-0.4730	-0.0369
10	0.2219	-0.1169	0.1719
11	0.1281	-0.2645	0.1100
12	0.1281	-0.3545	0.1255
13	-0.0281	-0.4730	0.0146
14	0.0656	-0.4770	0.1126
15	-0.1844	-0.6993	0.0301
16	0.2219	-0.2348	0.2208
17	0.0656	-0.4571	0.0868
18	0.2219	-0.2837	0.2337
19	-0.0594	-0.3770	-0.0369
20	0.0031	-0.4770	-0.0111

Table 3. Student-phase related; (a) Population from table 2(e) with cost values; (b) Updated population_02 based on the mutual learning; (c) Top-20 query plans after 1st generation in TLBO.

QP No.	Updated_Population_01(Cost-Matrix)			$Fitness_2$
	QAC	QLC	LPC	
1	0.1281	-0.2506	0.0868	0.0868
2	0.2531	-0.1169	0.1667	0.1055
3	0.2219	-0.2486	0.2079	0.1543
4	0.1594	-0.3545	0.2054	0.1933
5	0.1906	-0.1169	0.1925	0.0871
6	0.1281	-0.1169	0.0816	0.0368
7	0.2531	-0.1169	0.2853	0.1591
8	0.1594	-0.1169	0.1538	0.0627
9	-0.0594	-0.4730	-0.0369	0.2286
10	0.2219	-0.1169	0.1719	0.0924
11	0.1281	-0.2645	0.1100	0.0985
12	0.1281	-0.3545	0.1255	0.1579
13	-0.0281	-0.4730	0.0146	0.2247
14	0.0656	-0.4770	0.1126	0.2445
15	-0.1844	-0.6993	0.0301	0.5240
16	0.2219	-0.2348	0.2208	0.1531
17	0.0656	-0.4571	0.0868	0.2208
18	0.2219	-0.2837	0.2337	0.1843
19	-0.0594	-0.3770	-0.0369	0.1470
20	0.0031	-0.4770	-0.0111	0.2276

QP No.	Updated_Population_02()		
	QAC	QLC	LPC
1	0.1281	-0.2266	0.1075
2	0.2463	-0.2098	0.1667
3	0.2219	-0.2486	0.0970
4	0.1594	-0.3736	0.2054
5	0.1730	-0.1169	0.1925
6	0.1281	-0.1169	0.0816
7	0.2531	-0.3327	0.0922
8	0.1594	-0.1169	0.1538
9	-0.0505	-0.4552	0.0207
10	0.2219	-0.1169	0.1719
11	0.1536	-0.2645	0.1100
12	0.2036	-0.2883	0.0827
13	0.1677	-0.3446	0.0146
14	0.0739	-0.4319	0.1261
15	-0.0894	-0.2835	0.1184
16	0.2219	-0.2768	0.2208
17	0.0735	-0.4571	0.1367
18	0.2080	-0.2837	0.1562
19	-0.0594	-0.3178	-0.0369
20	0.0699	-0.4770	-0.0111

Top-20 Query Plans		
QP No.	Rank	QC Cost
15	1	0.0918
19	2	0.1325
9	3	0.1474
13	4	0.1721
20	5	0.2909
17	6	0.3375
11	7	0.3434
14	8	0.3733
6	9	0.3736
12	10	0.3794
1	11	0.3981
4	12	0.4447
8	13	0.5661
5	14	0.5788
16	15	0.6193
3	16	0.6278
18	17	0.6676
7	18	0.6959
10	19	0.6964
2	20	0.7355

Next phase is student phase, in which students or learners raise their knowledge level by interaction among themselves. A learner or students interacts randomly with other learners to enhance his or her knowledge. Core philosophy behind this phase is that a learner learns new things if the other learner has more knowledge than him or her. The updated population (Table 2(e)) is the input to this phase, algorithm randomly selects pair of the students and

compare based on the fitness value (fitness$_2$), winner of the comparison (QP_1 better than QP_2, or vice versa) will updating the weaker student in pair. The students with better subject's values will be improving the subjects values of weaker students, similarly each student go through this step at least once in algorithm run.

On the completion of student phase algorithm evolved with a new set of population (Update_population_02), as shown in Table 3(b) and this completes the algorithm run of first generation. After a number of sequential teaching–learning cycles in which, the teacher disseminates knowledge to the learners and their knowledge level increases toward the teacher's level, the distribution of the randomness within the search space becomes more and more smaller around a point that is considered the teacher. Therefore, the knowledge level of the entire class is smooth and the algorithm converges to a solution. The algorithm terminates for the specific values of termination criteria, eg. Top-K query plans, Query Cost, No. of Generation etc. The final ranks on QEP, represent the fitness quotient of specific QEP for result generation, as the cost values for the QEP optimal. The selected QEP's are used for result generation and thus supplied to query executor. The metadata related to the sites involved are fetched and supplied to query compiler and executor. Final result is send back to the origin site from which user send the query and query plans is kept in directory for future reference.

5. Result Analysis

5.1 Top-K query plans generation for a given query cost of 'optimal query' plan generation

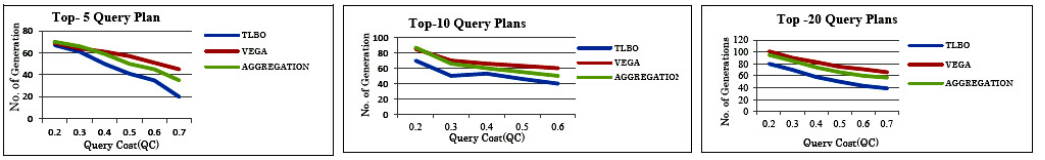

Fig. 2. Generation of Top-K ($K = 5, 10, 20$) query plans for query cost ($QC = 0.2, 0.3, 0.4, 0.5, 0.6.0.7$) with no. of evolution or iterations on algorithms (TLBO, multi-objective VEGA and aggregation based GA with, crossover probability (P_c) $= 0.8$, Mutataion probability (P_m) $= 0.2$, Weight$_{QAC} = 0.2$, Weight$_{QLC} = 0.5$, Weight$_{LPC} = 0.3$).

5.2 Generation of 'optimal top-K' in different number of sites (N_s) and relations (N_r)

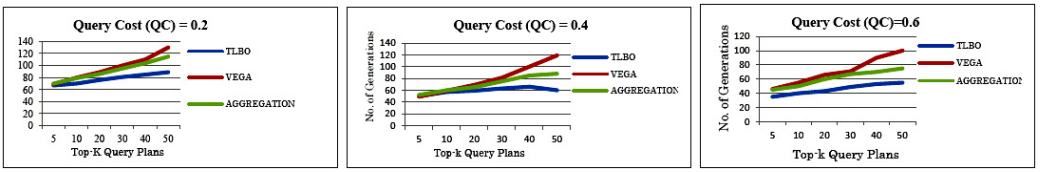

Fig. 3. For given query cost ($QC = 0.2, 0.4, 0.6$) generation of Top-K ($K = 5, 10, 20, 30, 40, 50$) query plans for in no. of iterations or generations of Algorithm (TLBO, Multi-Objective VEGA and Aggregation based GA, crossover probability (P_c) $= 0.8$, Mutataion Probability (P_m) $= 0.2$, Weight$_{QAC} = 0.2$, Weight$_{QLC} = 0.5$, Weight$_{LPC} = 0.3$).

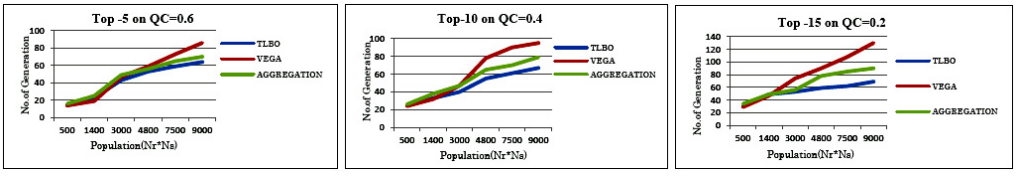

Fig. 4. Generation of 'optimal top-K' query plans with query cost ($K = 5$ & $QC = 0.6$, $K = 10$ & $QC = 0.4$, $K = 15$ & 0.2) for different Number of Sites (N_s) and relations sizes (N_r) in number of iterations or generations of algorithm (TLBO, multi-objective VEGA and aggregation based GA with, crossover probability (P_c) $= 0.8$, Mutataion probability (P_m) $= 0.2$, Weight$_{QAC} = 0.2$, Weight$_{QLC} = 0.5$, Weight$_{LPC} = 0.3$).

6. Conclusion

In distributed database system data is dispersed over the multiple sites, this distribution of data is based on partition or replication based due to which a given relation can be found in more than one sites. Query processing in such environment is difficult task for query processor, as multiple equivalent alternatives. Query processing in such environment, major objective design are CPU, I/O and the site-to-site communication cost, among these, the site-to-site communication cost is the dominant cost. This requires an optimization mechanism to generate optimal set of query plans to retrieve results for user query. Multi-objective optimization is a very important research area in engineering studies, because real-world design problems require the optimization of a group of objectives. Multiple, often conflicting, objectives arise naturally in most real-world optimization scenarios. Adding more than one objective to an optimization problem adds complexity. In this paper, TLBO algorithm is employed to generate optimal top-k query plans for proposed design objectives and in unconstrained function and its performance is compared with nature-inspired optimization technique, genetic algorithm (GA). The experimental results in Fig. 2, Fig. 3 and Fig. 4 show that the TLBO performs competitively better on the generation top-k query plans with other optimization methods. Therefore, the TLBO algorithm is effective and robust and has a great potential for solving similar multi-objective problems. The optimality on generation of query plans by other swarm based optimization techniques is part of our future work based on the similar design objectives.

References

[1] B. M. Alom, F. Henskens and M. Hannaford, Query Processing and Optimization in Distributed Database Systems, *IJCSNS International Journal of Computer Science and Network Security*, vol. 9, no. 9, pp. 143–152, September (2009).

[2] Arun Swami and Anoop Gupta, Optimization of Large Join Queries, *Proceedings of the ACM SIGMOD International Conference on Management of data*, Chicago, Illinois, USA, pp. 8–17, June 01–03, (1988).

[3] Chengwen Liu, Hao Chen and Warren Krueger, A Distributed Query Processing Strategy using Placement Dependency, *Proceedings of the Twelfth International Conference on Data Engineering*, pp. 477–484, February 26–March 01 (1996).

[4] M. Gregory, Genetic Algorithm Optimization of Distributed Database Queries, *Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence*, pp. 271–276, (1998).

[5] Hongbin Dong and Yiwen Liang, Genetic Algorithms for Large Join Query Optimization, *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pp. 07–11, London, July (2007).

[6] Y. E. Ioannidis and Younkyung Kang, Randomized Algorithms for Optimizing Large Join Queries, *ACM SIGMOD Record*, vol. 19, no. 2, pp. 312–321, June (1990).

[7] Y. E. Ioannidis and Younkyung Kang, Left-Deep vs. Bushy Trees: An Analysis of Strategy Spaces and its Implications for Query Optimization, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Denver, Colorado, pp. 168–177, USA, May (1991).

[8] Y. E. Ioannidis and Eugene Wong, Query Optimization by Simulated Annealing, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, San Francisco, California, pp. 9–22, USA, May (1987).

[9] S. Lin and B. W. Kernighan, An Effective Heuristic Algorithm for the Travelling-Salesman Problem, Operations Research, vol. 212, pp. 498–516, (1973).

[10] C. Liu and C. Yu, Performance Issues in Distributed Query Processing, *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 8, pp. 889–905, August (1993).

[11] Matthias Jarke and Jurgen Koch, Query Optimization in Database Systems, *ACM Computing Surveys (CSUR)*, vol. 16, no. 2, pp. 111–152, June (1984).

[12] R. Mendes, J. Kennedy and J. Neves, Watch thy Neighbour or How the Swarm can Learn from its Environment, *Proceedings of the IEEE Swarm Intelligence Symposium*, Indianapolis, pp. 88–94, (2003).

[13] Michael Stillger and Myra Spiliopoulou, Genetic Programming in Database Query Optimization, *Proceedings of the First Annual Conference on Genetic Programming*, Stanford, California, pp. 28–31, July (1996).

[14] M. C repinsek, S-H. Liu and L. Mernik, A Note on Teaching–Learning-Based Optimization Algorithm, *Inf. Sci.*, vol. 212, pp. 79–93, (2012).

[15] T. Niknam, A. K. Fard and A. Baziar, Multi-Objective Stochastic Distribution Feeder Reconfiguration Problem Considering Hydrogen and Thermal Energy Production by Fuel Cell Power Plants, Energy, pp. 563–573, (2012).

[16] T. Niknam, F. Golestaneh and M. S. Sadeghi, H-Multiobjective Teaching–Learning-Based Optimization for Dynamic Economic Emission Dispatch, *IEEE Syst. J.*, vol. 6, no. 2, pp. 341–352, (2012).

[17] Peter Bodorik and J. Spruce Riordon, Distributed Query Processing Optimization Objectives, *Proceedings of the Fourth International Conference on Data Engineering*, pp. 320–329, February (1988).

[18] S. Rho and S. T. March, Optimizing Distributed Join Queries: A Genetic Algorithmic Approach, *Annals of Operations Research*, vol. 71, pp. 199–228, (1997).

[19] R. V. Rao, V. J. Savsani and D. P. Vakharia, Teaching-Learning-Based Optimization: A Novel Method for Constrained Design Optimization Problems, *Computer-Aided Design*, vol. 43, no. 3, pp. 303–315, (2011).

[20] R. V. Rao, V. J. Savsani and D. P. Vakharia, Teaching–Learning-Based Optimization: an Optimization Method for Continuous Non-Linear Large Scale Problems, *Inf. Sci.*, vol. 183, pp. 1–15, (2012).
[21] R. V. Rao and V. Patel, An Elitist Teaching–Learning-Based Optimization Algorithm for Solving Complex Constrained Optimization Problems, *Int. J. Ind. Eng. Comput.*, vol. 3(4), pp. 535–560, (2012).
[22] Stefano Ceri and Giuseppe Pelagatti, Distributed Databases Principles and Systems, McGraw-Hill, *Inc.*, NY, (1984).
[23] M. Tamer Ozsu and Patrick Valduriez, Principles of Distributed Database Systems (2nd ed.), Prentice-Hall, *Inc.*, Upper Saddle River, NJ, (1999).
[24] V. Togan, Design of Planar Steel Frames using Teaching-Learning Based Optimization, *Engineering Structures*, vol. 34, pp. 225–232, (2012), doi:10.1016/j.engstruct.2011.08.035?
[25] T. V. Vijay Kumar, V. Singh and A. K. Verma, Distributed Query Processing Plans Generation using Genetic Algorithm, *International Journal of Computer Theory and Engineering*, vol. (31), pp. 38–45, (2011).
[26] Vikram Singh and Vikash Mishra, Distributed Query Plan Generation using Aggregation Based Multi-Objective Genetic Algorithm, In *Proceedings of 2014 International Conference on Information and Communication Technology for Competitive Strategy (ICTCS-14)*, Udaipur, Rajasthan, India, vol. (26), pp. 1–8, November (2014).