International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-2, July 2019

Teaching Learning Based Optimization in Semantic Web of Distributed RDF

Kura Shailaja, Puligadda Veereswara Kumar, Sarvadevabhatla Durga Bhavani

Abstract*: Semantic web data use as a unified data model in various areas, such as Bioinformatics, media data, Wikipedia, social networks, and government open data. Sharing information among people using semantic web helps to understand and manipulation of information. In the semantic web, the Resource Description Framework (RDF) denotes the linked data. The logical data is represented as RDF model to manage the unformatted data and it provides an ability to machine interpretability of data. The major problem on the web is to handle the large volume of the data that also has other challenges like query processing and optimization over widely distributed RDF data. In this research, the Teacher Learning based Optimization (TLBO) algorithm is proposed for the query optimization to reduce query cost, and optimize the computation time of the query. The TLBO technique select the suitable location and size of the population based on the data that effectively provide the solution for the distributed data i.e.., triple pattern of semantic web. The experimental result showed that the TLBO in query optimization performed well in the manner of query computation time compared to existing methods like MARVEL. Additionally, the results showed that the proposed TLBO model achieved nearly 4.93 seconds for executing the multiple queries in LUBM dataset.

***Index Terms*: Resource Description Framework, Structural Query Language, Semantic Web, Teaching Learning Based Optimization, Query Optimizer.**

I. INTRODUCTION

The semantic Web is like a huge single database made through structured metadata, vocabularies and linking between open data. In this process, three standards like RDF, SPARQL, Web Ontology Language (OWL) are used and these are established by the World Wide Web Consortium (W3C). The semantic web development processed layer-by-layer [1]. Foundation for representing metadata process established by RDF. The important parameter in RDF are subject, predicate and object and this is a graph-based model. In the recent years, use of RDF increased as a common data model for information modeling and conceptual description [2]. RDF is domain independent, but RDF schema helps to describe specific domains that is a primitive ontology language. RDF Schema provides key features like property, sub-property, class, subclass relations, range restrictions and domain. OWL developed based on the RDF schema that provides the features like the scope of properties, combination of classes, disjointness of classes, special properties (transitive/unique/inverse) and cardinality restrictions. SPARQL is the query language established by the W3C recommendation to query RDF data. SPARQL designed based on the triple pattern that is subject, object and predicate position. The solution of query binds the variable to respective URI or literal in the RDF model according to query structure [3]. Several SPARQL query-processing methods proposed in the recent years to store RDF data and process the queries [4]. Many techniques process on the single machine that does not provide the effective answer for complex queries. The Jena and Sesame method designed to execute the queries within the community of Semantic Web at earlier time. To maintain the large number of queries in the dataset, various methods are proposed such as RDF-3X [5], Hexastore [6], and SW-Store [7].

Zou [8] proposed gStore by considering the characteristic of RDF graph, a graph-based approach store RDF data and process SPARQL query by subgraph matching. With the rapid generation of RDF data, it is difficult for RDF data management systems to process data storage effectively [9-10]. Moreover, complex SPARQL queries is another problem that cannot be ignored, and most of the systems only considered BGP queries. Additionally, the requirements of users are growing when RDF data adopted by many application fields, like Wikipedia. Therefore, multiple requirements make a great challenge to handle [11]. The major issue faced by the RDF query engines is to process the query with the lower execution cost. The total query computation time depends on the query path and optimization techniques is applied to the order of the query execution thus contribute to efficient querying. The query paths increase exponentially with the query size, which makes it's less important to optimize the query path. Several soft computing methods used to overcome this problem [12]. The proposed TLBO method process the query in the constrained manner that reduces the computation cost of the method. Query optimization is an issue existing in the RDF data that is difficult to process. The experimental results shown that the TLBO outperforms the existing method in query optimization in the manner of query execution cost. The experiments involve in the use of different queries like star, chain to analyze the

Revised Manuscript Received on 30 July 2019.
***** Correspondence Author
Kura Shailaja*, Department of Computer science and Engineering, Methodist College of Engineering and Technology, Hyderabad, India.
Puligadda Veereswara Kumar, Department of Computer science and Engineering, Osmania University, Hyderabad, India.
Sarvadevabhatla Durga Bhavani, Department of Computer science and Engineering, Jawaharlal Nehru Technological University Hyderabad, Hyderabad, India.

computation time. This method is processed in main-memory of RDF data model. The TLBO technique can effectively handle the distributed data due to optimal selection of location and size of the population. The rest of the paper is organized as: Section II describes the related works of SPARQL queries in distributed RDF. The basic concept of SPARQL query graph is described in Section III. The challenges of query and their solution (i.e. proposed TLBO algorithm) are given in Section IV. The validated results of the proposed method are depicted in Section V and finally the conclusion is made in Section VI.

II. RELATED WORK

E.G. Kalayci, T.E. Kalayci, and D. Birant [13] implemented a method for SPARQL queries optimization with different shapes of graphs. This technique reordered the triple based on Ant Colony Optimization (ACO) technique. The execution cost of the SPARQL reduced by reordering the triple patterns. The method focuses on the main memory of RDF data and optimize the SPARQL queries using the different methods like MAX-MIN Ant System (MMAS), Ant System (AS) and Elitist Ant System (EAS). The experiments shown that the MMAS significantly reduced the computational time of the query. The results proved that this approach provided considerable performance than other techniques. MMAS algorithm failed to fulfill the expectations because MMAS does not provide better results for many queries. The additional computation and the problem of inapplicability of the MMAS are the main reasons for low performance.

P. Peng, L. Zou, M. T. Ozsu, L. Chen, and D. Zhao [14] developed a method for the query processing in the RDF graph in the distributed process and this technique based on the "partial evaluation and assembly" method. The method processed the Q query for each graph part in the parallel manner to identify the local partial matches. The second step was to assemble the matches in local partial to process the crossing matches. Two different techniques followed in this method named as distributed assembly and centralized assembly. The method treated the interconnected RDF (in linked open data) as a virtually connected distributed dataset. Some RDF repositories gave the endpoint of SPARQL and other methods are not having query capability. The data present in these sites need to be transferred and this affects the performance and cost function.

S. Saharan, J. S. Lather, and R. Radhakrishnan [15] proposed a new method for the optimization of the SPARQL query with various triple patterns. The Differential Evolution (DE) applied for the triple pattern rearrange in order. This method was tested on RDF data in main memory and ARQ queries of Jena. The experiments compared the result of the DE approach with the different versions of the ACO algorithm and some other existing methods. The results showed that DE techniques provided better computation time as compared to the other approaches. The efficiency of the method is low due to the ACO method convergence time is uncertain and can be improved by using new optimization algorithms like Artificial Bee Colony, Cuckoo Search for better cost computation strategy.

P. Peng, L. Zou, and Z. Qin [16] presented a hybrid query known as SPARQL-Keyword (SK) that integrated keyword search and SPARQL. An integrated query algorithm is introduced in this method depend on a structural index and a distance-based index to answer SK queries. The structural index depends on frequent star patterns in the RDF data, and the distance-based index depends on the trees shortest path of chosen pivots in the RDF graph. The SK method was evaluated on the three large RDF graphs and the obtained result showed the efficiency of the method. The keyword mapping taken a long time because of the inverted index for the keywords were stored on disk and taken up a large part of the total time. Hence, it is difficult for SK method to improve efficiency.

R. Harbi, I. Abdelaziz, P. Kalnis, N. Mamoulis, Y. Ebrahim, and M. Sahli [17] implemented an AdPart that is a distributed RDF system to address the shortcomings of high start-up cost and flexibility. The AdPart uses a lightweight partition on the initial data that is distributed triples using hashing on their subjects. Therefore, the communication cost of future queries significantly minimized or even eliminated. The experiments verified that the AdPart start faster than other existing method and process the thousands of queries even before the starting of other methods. The wall time was divided into three parts of Adpart execution, redistribution, distributed execution and parallel queries. Adpart components works well with some workers and communication overhead is high. Only a single worker that are not measured executes few data.

D. Ibragimov, [18] proposed a method called MAterialized Rdf Views with Entailment and incompLetness (MARVEL). This method consists of view selection algorithm that depends on the associated RDFspecific cost model. This is an algorithm for rewriting SPARQL queries based on materialized RDF views and a view definition syntax. The experimental analysis showed that the MARVEL increased the query response time and effectively handling RDF specifics. The method provided poor performance when the updates present in materialized views of RDF.

Yuanyuan Chen, [19] implemented the query process which was pre-processed in the relational database to increase the effectiveness of queries that didn't contain data set attributes in a query. The LUBM dataset used in this method for performance analysis and the designed query was executed on it.

M. Pham and P. Boncz [20] developed the triples in actual RDF store based on the tabular structure, which does not follow the metadata ontologies. This technique has the rich physical database and has the considerable efficiency. The optimization technique can be applied to improve the query processing with low cost.

Ferr´e, [21] developed two algorithms namely answer portioning and lazy join and this method are process to increase the efficiency. This technique scales much better with the size of the query. The complexity of the method is low and the efficiency is considerable. The performance of the method is low and cost of the query need to be reduced.

In the above discussed research works, the main issue is the query processing time, which leads to poor performance in the semantic web. Therefore, the proposed TLBO optimization algorithm concentrates more on query processing time and optimization to increase the performance of distributed RDF.

III. SPARQL AND QUERY GRAPH

RDF data are used in many platforms, though the tuples data are very convenient however, this also has some disadvantages. The number of tuples becomes very large and the data table increases very fast. The complex queries have a large number of connection operation that affects the performance of query execution. A large number of connections in RDF can be explained below as the form of SPARQL and query graph.

When SPARQL queries are provided, the query engine represents the query in the query graph. Every triple patterns converted into the nodes and two nodes connected if the queries share the common variables. The query graph nodes analyze the dataset with the respective variable bindings and edges denotes the query join possibilities. Thus query can be defined in the graph from the relational query optimization, provides the connections and nodes of the edges. The SPARQL query can be represented in the graph structure called SPARQL graph. The nodes of the variable denote the query and the triple patterns formed between the edges.

The SPARQL follows the official W3C standard for extracting and querying the data from RDF graphs [22]. This denotes the counterpart to select project join query in the relational model. This depends on a powerful graph-matching scheme and this allows the binding variables in the RDF graph. Operators correspond to relational joins, projections, selections, unions, and left outer joins are hybridized to develop the more expensive queries.

Each triple pattern consists of subject, predicate, and object and these can be either literal or variable. The query specifies the known literal and correlates the unknown as the variables that can present in multiple patterns to compound join operations. The query processor is required to analyze all the possible connection that fulfill the given pattern and provide the bindings to the application. Relational Database Management Systems (RDBMSs) have continuously shown the efficiency, scalability and execution in hosting type data that previously not been anticipated to be stored in the relational databases. Furthermore, the powerful indexing mechanisms used in the RDBMS to handle the huge amount of data very effectively. Intuitively, this structure describes the sub graph that has to be extracted from the dataset. The block diagram of the proposed TLBO in query optimization is shown in Fig. 1.

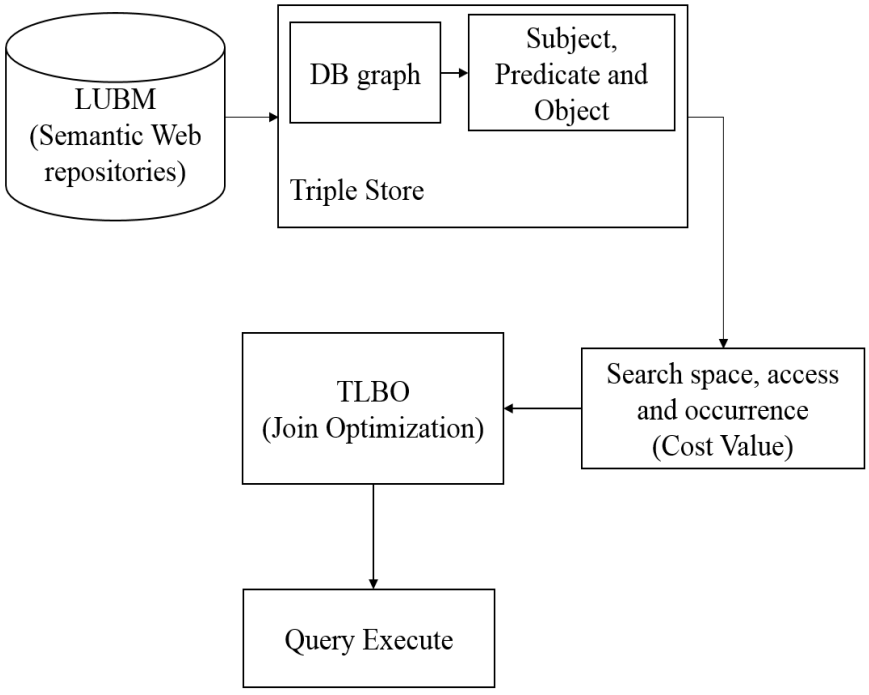

Fig. 1. The Block Diagram Of The TLBO Method In Query Optimization

IV. CHALLENGES IN QUERY OPTIMIZATION

The optimal join value of the SPARQL query is very difficult to find due to the nature of the RDF data. The query optimization complexity increases with the join's number. For instance, LUBM dataset are written in the SPARQL queries with the 12 joints, optimization technique cannot fully analyze the joins that tends to miss the potential plans to execute the query in low cost. The SPARQL queries have F times more joints than the SQL queries, where F denotes the average size of star pattern. This causes to increase the search space as 3F times. In the real time system, queries involve in a number of index scans. Secondly, the lack of schemes makes the optimizer with low information present for the relational optimizer like attributes, primary and foreign keys and the set of tables. The relational optimizer includes the statistics on these foreign keys and attributes, provide it for real time estimation. All these data present in the RDF data, where the attributes and foreign keys become structural correlation in the RDF graph.

The Characteristics Sets (CS) finds the simplest correlations to this attributes in the same entity [23]. The Dynamic Programming (DP) is required to measure the CS based metrics for the nodes in the graph. This will increase the execution time of the method and CS does not provide the relevance between the different sub graphs. The challenges present in the RDF data for the query optimizer. The DP methods require analyzing all nodes to find the cheapest one; this method is not efficient in the SPARQL queries, as it does not provide the approximate size of the search space. The DP method does not consider the structure of the query in the mid-size graph. This method analyzes many a priori sub-optimal sub plans in the construction of the plans. The optimization technique with the independent assumption does not measure the result size of most partial plans.

So, the independence assumption of the does not provide the accurate cost function, the optimization technique multiplies the frequency of the two predicates to process the selective join. The real cost values is highly differ from the optimization measure. Hence, the cost estimation of the query should be evaluated for finding query optimization.

A. Computing the cost by several factors

To find the query optimization value created in the bushy tree, two metrics named as "minimal access cost" and "search of minimal combinations" were used. In this section computation of various factors like search space, access cost and occurrence are explained.

a. Search Space computing:

The metrics are the basics for the identification of optimized query pattern. The search space is used to identify the proportion of the traversed varied combinations due to the low query elements connected under a join node pattern [24]. Hence, the metric search of minimal combinations supports to assess the search space utilization. The search of minimal combinations defined as the cumulative outcome of combinations amidst the subjects and objects count in a chosen predicate. The metric is very much reliant on the metric "search by minimal combinations". Minimal access cost is cumulative of cost for accessing the Distributed RDFs over varied subjects to attain the objects within the expected predicate.

The maximum possible combination (MPC) is carried out in Eq. (1).

$$MPC(p_k) = \sum_{i=1}^{|S|}\sum_{j=1}^{|O|}\left\{1 \exists p\left(S_i \rightarrow O_j\right) \equiv P\right\} \quad (1)$$

Where, p_k denotes the maximum combination and the counting of the possible combinations for a subject S_i and object O_j with an anticipated predicate, $p\left(S_i, O_j\right)$ is the concrete predicate of the subject and object. "$mpc\left(J_k\right)$" denotes the maximum possible combinations for a query pattern p_k The sum of the possible combination of the subject and predicate are used to provide the total possible combination in the graph.

b. Access Cost computing

Another important factor is the access time that is specific for all query elements in a query pattern, which targets the single RDF store [24]. The evaluation metric process is explored as Minimal access cost for accessing the Distributed RDFs over varied subjects to have the objects within expected predicate that is assessed in the Eq. (2).

$$ac(p_k) = \sum_{i=1}^{|S|}\sum_{j=1}^{|DRDF|}\left\{ac\left(S_i \Rightarrow rdf_j\right) \otimes v_c\right\} \quad (2)$$

Where, ac is the access cost, |S| is the cumulative number of subjects over a given query pattern $|DRDF|$ is the number of RDFs measured under distributed architecture $ac\left(s_i \Rightarrow rdf_j\right)$ can access cost to rdf_j under the selected subject S_i and v_c is indicating the number of visits. The notation $ac(p_k)$ denotes the minimal access cost of the query pattern .

c. Occurrence Cost

The occurrence cost of the query is calculated and used in the method to find the minimum query execution [24]. The query repeats for a number of times is considered as the occurrence cost and this is measured for each query in the dataset.

The cost like search space, access and occurrence are applied to the TLBO for identifying the optimized cost value of the query. The detailed explanation about the TLBO is provided in the next section.

B. Proposed Method: Teaching Learning Based optimization

TLBO method requires the common aspects like population size and number of generations. The input such as access cost, occurrence cost and search space cost are given into TLBO for cost optimization using fitness values of cost.

Considered two different teachers such as T_1 and T_2 teaching the same concepts with the same value level learners in the two different classes. The teachers measure the data distribution among the learners in the two different classes.

The normal distribution $f(X)$ is assumed for the query data; the query can have skewness. Normal distribution is expressed in the Eq. (3).

$$f(X) = \frac{1}{\left(\sigma\sqrt{2\pi}\right)} e^{\frac{\left(-(x-\mu)^2\right)}{2\sigma^2}} \quad (3)$$

Where σ^2 is the variance, μ is the mean and x is any value of required normal distribution function.

The difference between the learners is the result of their mean, i.e. a good teacher provides the better mean of the learners. The learners can also learn from the interaction between themselves, which helps to increase their mean.

The Teacher having more knowledge in the class and the learners try to mimic as the teacher. The teachers try to spread the knowledge among the students that increase the knowledge of the students and helps the students to obtain better performance. Therefore, the teacher increases the mean performance of the class based on the teacher knowledge.

Teacher T_A try to increase the mean M_A towards their own level depending on their capabilities, this increase the learner' level to the new mean M_B . Teacher put maximum effort to provide the knowledge to the learners and the learners will obtain the knowledge based on their quality. The student quality is measured based on the average value of the population. Teacher T_A puts the efforts to push the students from M_A to M_B, at this stage the students' needs the new teacher, as higher quality than themselves, and the new teacher is denoted as T_B . Similar to the other natural inspired algorithm, the TLBO is a population based optimization method that uses a population of solutions to compute the global solution. In TLBO, the population is the class of learners or group of learners. The population has a different design variable in the optimization algorithm. Still, the teacher is the most knowledgeable person and as the best solution. The different design variables are analogous to the different subject and the learners is analogous in the manner of 'fitness' value. The process of TLBO consists of two phases. The first part consists of 'Teacher Phase' and the second part consists of 'Learner Phase'. The 'Teacher Phase' means learning from the teacher and the 'Learner Phase' means learning through the interaction between learners.

a. Teacher Phase

A good teacher attempts to increase the learner's knowledge to their level. This is not possible for the teacher to move the average of a class to some level depend on the capacity of the class. This technique is based on the different factors.

The learner will attempt to increase the knowledge towards the best learner depend on the present mean value of the learner that is related to mean quality of the class. The learner will update using the Eq. (4). The fitness value of the teacher is high and this feature value is given to the learners for increasing the fitness value.

$$newX_i = X_i + r \times \left(Teach - T_{factor} \times Mean\right) \quad (4)$$

Where, X_i and X_j are randomly selected learners r, T_{factor} denotes the teaching factor that measure the average value, and r denotes the random number present in the range of 0 to 1. Value of T_{factor} is present either 1 or 2, that is measured randomly using the Eq. (5).

$$T_{factor} = round\left[1 + rand\left(0,1\right)\{2-1\}\right] \quad (5)$$

Modification of the Learner can be denoted as:
for each given X_i

$T_{factor} = round\left[1 + rand\left(0,1\right)\{2-1\}\right]$

$newX_i = X_i + r\left(Teach - T_{factor} \times Mean\right)$ Replace X_i with new X_i if that provides better value of function.

end for

b. Learner Phase:

Learners improves the knowledge based on the two values: (1) gain the knowledge from the teacher and (2) gain knowledge by interacting between themselves. The learners discuss randomly with other learners with the help of group discussion, formal communication, presentation, etc. A learner learns something new when interacting with the other learners, have more knowledge of their own.

The learner increases the knowledge with the interaction of the other learners. So, their knowledge is updated as the following Eq. (6 & 7).

For i=1: P_n

Another learner X_j randomly selected, such $i \neq j$

$$newX_i = X_i + r \times \left(X_i - X_j\right) \text{ i } f\left(X_i\right) < f\left(X_j\right) \quad (6)$$

Else,

$$newX_i = X_i + r \times \left(X_j - X_i\right) \quad (7)$$

Here, X_i and X_j are the randomly selected learners, r is a random number in range [0,1]. The r differs from rand, which denotes the random number used in the Teacher phase.

Modification of the Learners can be derived as in the Eq. (8 & 9)

for $i = 1 : n$

Select X_i and X_j at random suchthat $i \neq j$

$$newX_i = X_i + r \times \left(X_i - X_j \right) \text{ if } f\left(X_i\right) < f\left(X_j\right) \quad (8)$$

else,

$$newX_i = X_i + r \times \left(X_j - X_i \right) \quad (9)$$

Accept new X_i if it provides better value of functions

End for

The optimized value of the TLBO is used to identify the minimum execution cost of the query. The query contains the minimum cost value, which is identified by using TLBO method and execute for low resource consumption. The pseudo code of the TLBO is shown below.

The pseudo code is divided into two phase teacher and learner phase. In the teacher phase, the learners try to improve the fitness function based on the teacher value. In the student phase, the fitness function increase based on the discussion between the learners. The suffix i denotes the current learner and r denotes the random number.

First, the optimization problem and initial optimization parameters are assigned such as population size P, number of generation g. The random population is generated based on the population size defines the number of learners and subject offered. In the teacher phase, mean population is calculated in column wise that give the mean of particular subject as X_{mean}. The best solution is act as the teacher that is defined in the pseudo code as $Xteacher \leftarrow Best\_Solution(P)$ .The teacher attempt to shift the mean for the iteration. The T_F is the teaching factor and the value is selected between 0 to 1. The difference between the mean is measured and added it to the current solution to update the value. Accept the X_{mean} if it has the better value. In the learner phase, the random population is generated and the learner try to improve the knowledge based on the interaction. The process is terminated maximum number of generation.

Pseudo code

AlgorithmTLBO

begin $g \leftarrow 0$;

$Intiatlize\ Population(P, pop\_size)$

$Evaluate(P)$

$\{Calculate\_Fitness(P)\}$

repeat

$\{Teacher\ Phase\}$

$r = random(0 to 1)$

$T_F = round(1+r)\{1 or 2\}$

$X_{mean} \leftarrow Calculate\_Mean\_Vector(P)$

$X_{teacher} \leftarrow Best\_Solution(P)$

$Difference\_Vector = r \cdot (X_{teacher} - (TF \cdot X_{mean}))$

$X_{new}, i = X_{old}, I + Difference\_Vector$

Evalute X_{new}

$\{Calculate\_Fitness(P)\}$

$If X_{new}, i\ is\ better\ than\ X_{old}, i$ *then*

$\{Compare\ Fitness(P)\}$

$X_{old}, i \leftarrow X_{new}, i$

end if $\{End\ of\ Teacher\ phase\}$

$\{Learner\ Phase\}$

$ii \leftarrow random(pop\_size)\{ii \neq i\}$

if X_i *better than* X_{ii} *then*

$X_{new}, i = X_{old}, i + r \cdot (X_i - X_{ii})$

else

$X_{new}, i = X_{old}, i + r \cdot (X_{ii} - X_i)$

end if

$Evaluate(X_{new}, i)$

$\{Calculate\_Fitness(\ )\}$

if X_{new}, i *is better than* X_{old}, i *then*

$\{Compare\_Fitness(\ )\}$

$X_{old}, i \leftarrow X_{new}, i$

end if $\{End\ of\ Learner\ Phase\}$

end for

$g \leftarrow g + 1$

$until(g \neq num\_gen)$

$\{Termination\_Condition\}$

$Print\_Best\_Result(P)$

End

V. EXPERIMENTAL RESULT

The experiments were carried out and the efficiency of the proposed TLBO method in the optimization of the query with existing methods is evaluated. The proposed model was implemented using Java and executed on Intel 5^th generation platform. In the distributed environment of multiple triple stores and multiple clients was simulated using Java RMI, such that clients can submit SPARQL queries in parallel and the sample SPARQL query is shown below.

```
# Query1
# This query bears large input and high selectivity. It queries about just one class and
# one property and does not assume any hierarchy information or inference.
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT ?X
WHERE
{?X rdf:type ub:GraduateStudent .
 ?X ub:takesCourse
http://www.Department0.University0.edu/GraduateCourse0}
```

A. Dataset Description

The Lehigh University Benchmark (LUBM) dataset used in this work, to investigate the performance of the proposed method. The LUBM developed to analyze the performance of those repositories related to the queries in the large RDF data in a single realistic ontology.

B. Performance Evaluation

In this section, the performance of the proposed TLBO in query optimization is validated by using two metrics such as execution time for the query and memory usage for each query join. Table 1 depicts the performance of the proposed method by using a number of queries joins with its execution time. The computation time of the proposed TLBO in query optimization is compared with existing methods like different approaches of MonetDB RDF store: the original triple-based store (MonetDB-triple) and the emergent schema-based store (MonetDB-emer). The graphical representation of theexecution time is presented in Fig. 2. TLBO can effectively handle the distributed data due to the optimal selection of size and location of Distributed generation.

Table I. Computation Time Of The Proposed Method

Number of Query Joins	Execution Time (milli-seconds)			
	MonetDB-triple [20]	MonetDB-emer [20]	MARVEL [18]	Proposed Method
1	480	50	112	60
2	500	480	116	50
3	470	48	100	80
4	520	40	92	70
5	50	48	136	60
6	500	480	121	60
7	420	300	119	70
8	500	480	156	80

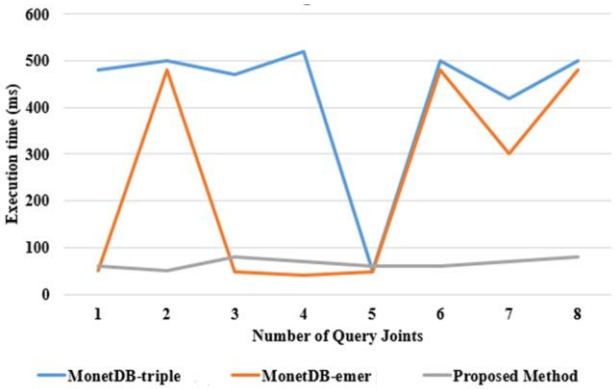

Fig. 2. Performance of Execution Time

From the above figure, the results show that the TLBO method achieved better execution for a number of queries. The execution time for a small number of queries decrease, but after some certain optimal point, the execution time increases when the number of queries increase.

C. Comparative Analysis

In this section, the computation time and memory usage of the proposed TLBO method compared with the existing method such as MARVEL approach in [18]. D. Ibragimov, [18] proposed a method to support implicit (derived) data. This method consists of a view selection algorithm based on an associated RDF specific cost model. This proposed method provides poor performance when the updates are presented as materialized views of RDF. Table 2 represents the performance of the proposed method with MARVEL in the manner of average computation time in the simple query and memory usage.

Table II. Performance of proposed method

Author	Methodology	Execution Time (seconds)	Memory usage (bytes)
D. Ibragimov [18]	MARVEL	11.8	365
Proposed Method	TLBO	4.93	30.43

The table 2 shows that, the proposed TLBO method achieved the best results in both memory usage and execution time. The existing method achieved 11.8s for multiple query joins, whereas the proposed TLBO achieved the only average of 4.93s for multiple query joins in SPARQL. In addition, the proposed methodology is analyzed with the query execution time in milliseconds for pre-processing the query. According to the characteristic of the data that the number of subjects and objects was large and the number of attributes was small. Yuanyuan Chen, [19], proposed the preprocessing storage of the data based on the relational database. The query analyzes the characteristics, and then process that helps to increase the efficiency of the query execution. Finally, the data were generated and used the LUBM data set, then, the designed query was executed on it. Table 3 explains the comparison of execution time for pre-processing the given query.

Table III. Execution Time for Pre-processing the Query

Author	Methodology	Execution Time (ms)
Yuanyuan Chen, [19]	Pre-process the query by relational database	68.33
Proposed Method	TLBO	35.35

From the above tables, it is clear that the proposed TLBO method achieved low execution time than pre-processing all queries in LUBM dataset. The execution time of the existing method achieved nearly 69 ms for pre-processing of nearly nine query joins, but the proposed TLBO achieved 35.35ms for processing the same number query joins.

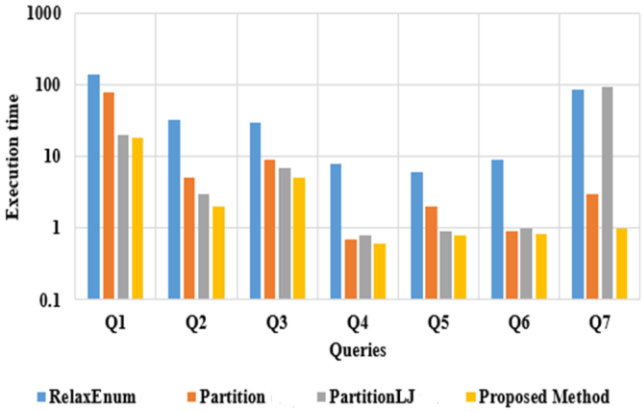

Fig. 3. The Computation Time Of The Various Methods

The different methods such as RelaxEnum, Partition, PartitionLJ and TLBO measured in terms of execution time for various queries which is shown in Fig. 3. The answers partitioning and lazy join are the two algorithms proposed in the research [21]. These methods are executed with the seven different queries and the execution time is calculated. From Fig. 3, it shows that the proposed TLBO method has less computation time compared to the other methods. The proposed TLBO method and other existing method are evaluated in the LUBM dataset. From, the overall results evaluation, the proposed TLBO achieved better execution time for query processing when compared with the existing methods.

VI. CONCLUSION

This paper provided the better optimization method for the query join order technique for SPARQL queries that has the higher performance than state-of-art methods in the manner of produce plans and computation time. This experimental analysis showed that the execution time of the technique is low for the query processing. This paper showed the efficiency of the proposed TLBO method in terms of execution time and compared it with the existing methods in semantic web query optimization. This paper discussed the query optimization based on

reordering the triple pattern in the main memory of the RDF data model. The execution time of the TLBO method is 4.98 s, while the execution time of the existing method is 11.9 s. The TLBO technique reduces the execution time based on the optimal selection of the size and position of the population. The future work of the method can be extended to other framework of query engines over the different queries on the triple patterns.

REFERENCES

1. G. Antoniou, and F.V. Harmelen (2004), A semantic web primer, MIT press.
2. W.E. Zhang, Q.Z. Sheng, Y. Qin, K. Taylor, and L. Yao, (2018). Learning-based SPARQL query performance modeling and prediction. World Wide Web, 21, pp. 1015-1035.
3. D.R. Dadhaniya, and A. Makwana. (2016). Survey paper for different SPARQL query optimization techniques. MJSRE J, 2, pp. 83-85.
4. M. Schmidt, M. Meier, and G. Lausen. (2010). Foundations of SPARQL query optimization. In: Proc. of 13th International Conference on Database Theory, ACM.
5. T. Neumann, and G. Weikum. (2008). RDF-3X: a RISC-style engine for RDF. Proceedings of the VLDB Endowment, 1, pp. 647-659.
6. C. Weiss, P. Karras, and A. Bernstein. (2008). Hexastore: sextuple indexing for semantic web data management. Proceedings of the VLDB Endowment, 1, 1008-1019.
7. D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach. (2009). SW-Store: a vertically partitioned DBMS for Semantic Web data management. The VLDB Journal, 18, pp. 385-406.
8. L. Zou, J. Mo, L. Chen, M.T. Özsu, and D. Zhao. (2011). gStore: answering SPARQL queries via subgraph matching. Proceedings of the VLDB Endowment, 4, pp. 482-493.
9. T. Tran, and G. Ladwig. (2010). Structure index for RDF data. Workshop on Semantic Data Management (SemData@ VLDB). 2, 2010.
10. O. Udrea, A. Pugliese, and V.S. Subrahmanian. (2007). GRIN: A graph based RDF index", AAAI, 1, 2007.
11. J. Feng, C. Meng, J. Song, X. Zhang, Z. Feng, and L. Zou. (2017). SPARQL query parallel Processing: a survey. IEEE International Congress on In Big Data (BigData Congress), pp. 444-451.
12. A. Hogenboom, F. Frasincar, and U. Kaymak. (2013). Ant colony optimization for RDF chain queries for decision support. Expert Systems with Applications, 40, pp. 1555-1563.
13. E.G. Kalayci, T.E. Kalayci, and D. Birant. (2015). An ant colony optimization approach for optimising SPARQL queries by reordering triple patterns. Information Systems, 50, pp. 51-68.
14. P. Peng, L. Zou, M. T. Ozsu, L. Chen, and D. Zhao. (2016). Processing SPARQL queries over distributed RDF graphs. The VLDB Journal—The International Journal on Very Large Data Bases, 25, pp.243-268.
15. S. Saharan, J. S. Lather, and R. Radhakrishnan. (2018). Optimization of Different Queries using Optimization Algorithm (DE). International Journal of Computer Network and Information Security, 10, pp. 52, 2018.
16. P. Peng, L. Zou, and Z. Qin. (2017). Answering top-K query combined keywords and structural queries on RDF graphs., Information Systems, 67, pp.19-35, 2017.
17. R. Harbi, I. Abdelaziz, P. Kalnis, N. Mamoulis, Y. Ebrahim, and M. Sahli. (2016). Accelerating SPARQL queries by exploiting hash-based locality and adaptive partitioning. The VLDB Journal—The International Journal on Very Large Data Bases, 25, pp.355-380,
18. D. Ibragimov, K. Hose, T.B. Pedersen, and E. Zimányi. (2016). Optimizing aggregate SPARQL queries using materialized RDF views. In: Proc. of International Semantic Web Conference, Springer, pp. 341-359, 2016.
19. C. Yuanyuan. (2017). A preprocessing strategy of RDF data query based on relational database.
20. M. Pham, and P. Boncz, (2016). Exploiting emergent schemas to make RDF systems more efficient. International Semantic Web Conference. Springer.
21. S. Ferré, (2018). Answers Partitioning and Lazy Joins for Efficient Query Relaxation and Application to Similarity Search. In: Proc. of European Semantic Web Conference, Springer, pp. 209-224, 2018.
22. E. Prud'hommeaux, (2008). SPARQL query language for RDF. W3C recommendation.
23. T. Neumann, and G. Moerkotte. (2011). Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins. In: Proc. of IEEE 27th International Conference on Data Engineering (ICDE), 2011.
24. K. Shailaja, P. Kumar, and D. Bhavani. (2017). Dr-Qpo: Discrete Rank Based Query Pattern Optimization Towards Parallel Query Planning and Execution for Distribute Triple Stores. Journal of Theoretical & Applied Information Technology, 95, 22.

AUTHORS PROFILE

Mrs. K. Shailaja is a Post Graduate in M.Sc from Osmania University in1993.and M.Tech (CSE) from Osmania university in 2008 and she is pursuing her PhD from JNTUH in the Area of "Semantic Web and RDF Data Stores". She is working as Asst. Professor in Methodist College of Engineering and Technology in the Department of Information Technology. She is a Life Member of ISTE, CSI.

Dr. P.V. Kumar has completed his M.Tech from Osmania University in 1985 and Ph.D. from Osmania University in 2008.He has put up three years of industry experience, three years of research experience and twenty five years teaching experience. He has published many research papers both in national/international conferences / Journals. Two of his research scholars awarded Ph.D. from JNTUK. Ten research scholars are still doing research under his guidance. His interested domains include 'database management system', 'data structures', 'algorithm's'. He held various positions like, Chairman, BOS, BCA Convenor.

Dr. S. Durga Bhavani, professor and current in charge of Examination Cell in SIT, JNTU Hyderabad. She has done her M.Tech(Computer Science) from JNTU Hyderabad. She has done her Ph.D. from University of Hyderabad. She has a vast experience of 24 years. Under her guidance 3 students are awarded Ph.D. Areas of Interest include Evidential Reasoning and Data mining techniques.