Received: April 19, 2019

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Semantic Web Query Join Optimization Using Modified Grey Wolf Optimization Algorithm

Rubin Thottupurathu Jose[1]* **Sojan Lal Poulose[2]**

[1]*School of Computer Sciences, Mahatma Gandhi University, Kottayam, Kerala, India*
[2]*Mar-Baselious Institute of Technology and Science, Kothamangalam, Kerala, India*
* Corresponding author's Email: rubinthottupuram@amaljyothi.ac.in

Abstract: Presently, the distributed Resource Description Framework (RDF) partition the data across several computer nodes. In that, many existing RDF systems results in expensive query evaluation and high start-up cost. To address these issues, a new optimization algorithm: modified Grey Wolf Optimization (GWO) has been developed in this research paper. In conventional GWO algorithm, after finding the best values of G_α,G_β and G_δ, stopping criteria is accomplished. In modified GWO algorithm, after finding the best values of G_α,G_β and G_δ, the alpha α value once again encircles the possible solutions for obtaining an optimal solution. In RDF data, query optimization is a challenging task, which has been effectively handled by modified GWO algorithm. In the experimental phase, modified GWO showed good performance in terms of execution time and memory usage as compared to the existing methodologies: Partial Evaluation and Centralized Assembly (PECA), Partial Evaluation and Distributed Assembly (PEDA), RDF-3X, Graph-Based SPARQL Query Engine (gStore), and Legato on Lehigh University Benchmark (LUBM) 10000 and DOREMUS 2017 datasets. Compared to these existing systems, the proposed system reduced the execution time around 2-5 minutes, and improves the precision, recall, and f-measure around 2-7%.

Keywords: Grey wolf optimization, Lehigh university benchmark dataset, resource description framework, structural query language, web query optimization.

1. Introduction

In recent decades, the large quantity of available data sources makes the data representation and classification as a complex process, so it is essential to represent the data in a semantically structured way that mainly relied on the RDF data model [1]. Presently, the data representation in RDF data model constantly growing in size, so querying and storing the RDF graphs becomes a very challenging task [2, 3]. Several approaches developed for increasing the efficiency of query retrieval. Most of the present approaches use map SPARQL queries and database management systems to structural query language for query retrieval [4, 5]. In addition, a few more approaches like RDF-3x, Jena, sesame, etc. developed for single node machines in the distributed environment [6]. These existing methods increase the storage space and delivers parallel query execution capabilities for managing the huge datasets [7]. The distributed environmental system utilizes a few Hadoop techniques: S2RDF, sempala on top of impala, rya on top of apache accumulo, SPARQLGX on top of apache spark, etc. [8]. These existing Hadoop systems optimized for a specific query pattern that may marginally improve the query performance.

So, there is a need for distributed RDF store with better performance on an extensive range of query types without renouncing a rapid loading phase. Generally, the computational time of the query depends on the optimization algorithm and query path. The size of the query path increases with the size of queries, so it consumes less time to optimize the query path [9, 10]. Presently, numerous soft computing methodologies utilized to reduce the time consumption and query path optimization. In this research study, an effective methodology developed

for improving the performance of semantic web query join optimization. At first, the input data were collected from the datasets: LUBM 10000 and DOREMUS 2017. Then, the distributed query graph developed on the collected data and successively calculate the cost function. At last, the modified GWO algorithm was developed for solving the engineering optimization issues. The modified GWO algorithm attempts to solve the optimization issues, due to the multipurpose property. In modified GWO algorithm, after finding the best values of G_α,G_β and G_δ, the alpha α again encircles the possible solutions for obtaining an optimal solution. This process effectively diminishes the computational time of the developed system compared to GWO algorithm.

This research paper is arranged as follows. Several papers on semantic web query join optimization are reviewed in section 2. Detailed explanation about the proposed methodology is given in section 3. Section 4 illustrates about the quantitative and comparative analysis of proposed methodology. The conclusion is done in section 5.

2. Literature review

The researchers in semantic web query join optimization developed numerous methodologies. In this literature review section, a brief discussion of some important contributions to the existing literatures is presented.

P. Peng, L. Zou, M.T. Özsu, L. Chen, and D. Zhao, [11] presented a new technique for processing the SPARQL queries over a huge RDF graph in a distributed environment by adopting the "partial evaluation and assembly" system. At first, the developed technique simultaneously evaluates the queries on each graph fragment for identifying the local partial matches. In the second step, these local partial matches assembled for computing the crossing matches. In this research work, two dissimilar assembly strategies (centralized assembly and distributed assembly) were developed in order to minimize the edges and vertices in the intermediate results. The developed methodology effectively preservers the inter-connected RDF repositories as a virtually integrated distributed database. A few RDF repositories provide SPARQL endpoints and others may not have query capability. In real-time applications, queries in the same time usually overlapped.

R. Harbi, I. Abdelaziz, P. Kalnis, N. Mamoulis, Y. Ebrahim, and M. Sahli, [12] developed an AdPart distributed RDF system for addressing the short-comings of existing works. Initially, AdPart distributed RDF system used lightweight portioning on the input data for distributing the triples by hashing on the subjects. Besides, locality-aware query optimizer was used on the AdPart distributed RDF system for minimizing the data communication. As a result, the communication cost of future queries reduced. The experimental result confirmed that the developed AdPart distributed RDF system works faster than all existing systems. In this research work, it was very hard to select and execute the queries by a single worker.

E.G. Kalayci, T.E. Kalayci, and D. Birant, [13] presented an effective system to optimize the SPARQL queries with dissimilar graph shapes. In this research paper, ant colony optimisation algorithm was used to re-order the triple patterns that effectively decreases the execution time of SPARQL queries. This system mainly focused on in-memory models of RDF data, and optimized the SPARQL queries in terms of max-min ant system, elitist ant system and other ant system algorithms. The experimental outcome confirmed that this methodology delivered better performance in terms of execution time. In large datasets like LUBM, the developed algorithm includes two major concerns; additional computational load and the problem-specific inapplicability.

P. Peng, L. Zou, and Z. Qin, [14] developed a new hybrid query: SPARQL-Keyword (SK) for improving the performance of key word search and SPARQL. In this research, the developed query was integrated with the structural and distance based index for answering the SK queries effectively. Usually, the structural index works based on frequent start patterns in RDF data. Likewise, the distance based index works based on shortest path trees of selected pivots in the RDF graph. Here, the experiments were conducted on three large real RDF graphs and the outcomes demonstrate the efficiency of SK query. The keyword mapping consumed more time, because the inverted index keywords were hard to retrieve from the storage. In such circumstances, it was difficult for SK method to reduce the execution item.

L. Zou, M.T. Özsu, L. Chen, X. Shen, R. Huang, and D. Zhao [15] developed a new method (gStore) to answer SPARQL queries efficiently. In developed system, the RDF data were stored in large graph for representing SPARQL query as query graph. In addition, an index with pruning rule was developed to achieve scalable and effective query processing. Also, an effective maintenance approach was used for handling online updates over RDF repositories. The experiment result shows that the efficiency of developed system was better related to the existing

systems. The developed system lag with the concern towards handling the missing attribute values in the dataset.

M. Achichi, Z. Bellahsene, M.B. Ellefi, and K. Todorov, [16] developed a new system for semantic web query join optimization. The developed system includes four major phases: pre-processing, instant profiling, vector representation and post-processing. Initially, data cleaning process was carried-out for removing the irrelevant data. Then, the instant profiling was used to represent each resource for the comparison task. To reduce the false positives rate, instant vector representation compares the resources. Finally, post-processing phase was carried-out by using key ranking methodology and hierarchical clustering for disambiguating highly not identical instances. The major disadvantage of the developed system was the creation of post-processing phase that decreases the system performance in light of recall, precision, and f-measure.

A new optimization algorithm (modified GWO) has been developed for improving the performance of semantic web distributed RDF and to overcome the above-mentioned problems.

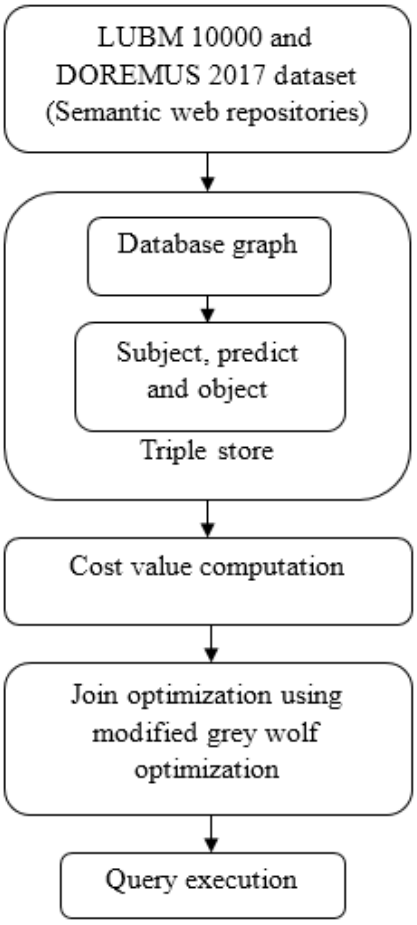

Figure.1 Work flow of proposed methodology

3. Proposed methodology

In semantic web query optimization, the modified GWO comprises of four major phases such as; data collection, triple store, cost value computation and query optimization using modified GWO. Fig. 1 represents the block diagram of the proposed methodology. The detailed explanation about the proposed methodology is given below.

3.1 Dataset description

At first, the input data is collected from LUBM 10000 and DOREMUS 2017 datasets. LUBM 10000 dataset is developed for facilitating the estimation of semantic web repositories in a systematic and standard way. The benchmark dataset evaluates the performance of repositories on the basis of extensional queries over a huge dataset, which promises a single realistic ontology. The LUBM 10000 dataset contains a repeatable synthetic data, customizable synthetic data, a university domain ontology, a set of test queries and numerous performance measures. The number of triples in LUBM 10000 dataset is 1,334,481,197, RDF N3 file size is 153,256,699, and the number of entities is 217,006,852. The other components of LUBM 10000 dataset are given below,

Ontology: The benchmark ontology is called as Univ-bench: Web Ontology Language (OWL) version.

Data generator: It generates synthetic data over Univ-bench ontology. The generated data are customizable and repeatable that allows users to specify the starting index of universities.

Test queries: Currently, the benchmark comprises of fourteen test queries. The file contains all queries in SPARQL 1.0 syntax format, which is separated by blank lines for identifying the comments. In LUBM 10000 dataset, the visualization of the queries in SVG and JPG format.

Test module: It comprises of both query test and data loading test with configurable test plans.

In addition, DOREMUS 2017 comprises of two important sections: Heterogeneities (HT) and False Positive Trap (FPT).

HT: It comprises of two datasets (BnF-1 and PP-1), containing 238 instances. It includes dissimilar types of HTs, which are collected from different degree of description, differences in spelling, multilingualism, and differences in catalogues.

FPT: It includes two datasets (BnF-2 and PP-2) that contains 75 instances each.

3.2 Detailed explanation about SPARQL and query graph

After the data collection, distributed query graph is built for semantic queries with edges, nodes and properties to represent and store data. Generally, RDF data utilize in several applications, where the RDF tuples data are very suitable for semantic web query join optimization. In a few circumstances, the data table increases rapidly if the number of tuples data are high. Whereas, the complex queries have more connection operations that degrade the query execution performance. The connections in RDF is explained below in the form of the query graph and SPARQL. In SPARQL query, the query engine starts with constructing a representation that is named as query graph. Each query triple pattern is turned into a node of the query graph. The two nodes connect only if the related triple patterns share a common variable or there is a FILTER condition related to the two triple patterns.

Theoretically, the query graph node analyses the datasets based on variable bindings, and the edges related to the join possibilities in the query. Additionally, the query graph node defines the query graph of SPARQL related to traditional query graph from relational query optimization, where the nodes are interconnected and joins predicates from the edges. In addition, SPARQL query is also represented as a graph structure, which is called as SPARQL graph. The query nodes state the query variables, where the triple patterns form edges between the nodes. The SPARQL uses W3C standard in order to extract and query the data from RDF graphs. This procedure denotes counter-part to select the project to join queries in the relational model that depends on the powerful graph-matching scheme by allowing the binding variables in the RDF graph. Operators correspond to relational projections, joins, selections, unions, and left outer joins are hybridized for developing the more expensive queries.

Each triple pattern comprises of subject, predicate, and object, which are either literal or variable. The query represents the known and unknown literal variables in multiple patterns for compounding the join operations. Additionally, a query processor requires in order to analyse the possible connections between the given patterns, which provides bindings to the application. Relational database management systems have continuously shown the scalability, efficiency and execution in hosting type data that previously not been predicted in the relational databases. Further, the powerful indexing tool use in the relational database management systems in order to manage the huge amount of data very effectively. Instinctively, this process describes the sub-graphs, which need to be extracted from the datasets. A few major challenges faced by the researchers in semantic web query join optimization is detailed in the following section.

3.3 Challenges in semantic web query join optimization

Detecting the order of optimal join in SPARQL query is very critical, due to the nature of RDF data. Secondly, query optimization complexity is exponential in the number of joins. For instance, LUBM 10000 dataset comprises of SPARQL queries with more than twelve joins, so the undertaken optimizer cannot analyse the full search space, which potentially misses the best plan. The SPARQL query plans have F times more joins than correspondent SQL plans, where F is denoted as the average size of a start pattern. In real-time applications, SPARQL queries with joins involve a hundred index scans. Secondly, the lack of schema leaves the essential information, which is readily available to any relational optimizer such as, set of tables, foreign keys, etc. The relational optimizer keeps the statistics on the attributes and foreign keys, and uses it result for size estimation. All this information implicitly present in RDF data, where the attributes and foreign keys become structurally correlates in the RDF graph.

The simplest correlation corresponds to the attributes of similar entities, which are captured by the characteristic sets. However, the dynamic programming algorithm computes the characteristic sets based estimation for every non-empty sub-graph of the query. This process significantly increases the performance of the dynamic programming algorithm, and characteristic sets do not identify the relevance between the different sub graphs in RDF. Besides, the characteristics of RDF data creates the following challenges in the query optimizer.

- The approximate size of search space for more SPARQL queries does not allow the standard dynamic programming algorithm exploration, since it has to look at all the valid plans for identifying the cheapest one.
- Secondly, even in the mid-sized query graphs, the dynamic programming algorithm ignores the structure of query and it considers many a priori sub-optimal sub plans during the plan construction.
- The optimizer under the independence assumption fails to estimate the result sizes

of most partial plans. The independence assumption leads to a significant underestimation of the cost function, since the optimizer merely multiplies the frequencies of two predicates to obtain the join selectivity.

- The real cost of the partial plans is much worse than the optimizer's expectation. Hence, the cost estimation of the query should be evaluated for finding query optimization.

3.4 Cost value computation

In this sub-section, bind join $\forall_b$ and symmetric hash join $\forall_h$ are applied for query planning between the two cost estimations [17], which is mathematically represented in the Eq. (1), (2), and (3).

$$Cost\ (B1\ \forall_h\ B2) = \frac{(1+TC)}{TC} \times CSQ + C2_h + C3_h \quad (1)$$

Where,

$$C2_h = C(B2) \times CRT \quad (2)$$

$$C3_h = (C(B1) + C(B2)) \times CHT \quad (3)$$

Where, $C2_h$ is denoted as cost of receiving the largest tuple set, and $C3_h$ is represented as the cost of intersecting received sets. The query planning of bind join $\forall_b$ is mathematically denoted in the Eq. (4), (5), and (6).

$$Cost\ (B1\ \forall_b\ B2) = CSQ + C2_b + C3_b \quad (4)$$

Where,

$$C2_b = C(B1) \times CRT \quad (5)$$

$$C3_b = CSQ \times \frac{\left(\frac{C(B1)+BSZ-1}{BSZ}\right)+CTC-1}{CTC} \quad (6)$$

Where, $C2_b$ is denoted as cost of receiving $B1$ set, $C3_b$ is represented as cost of sending bound $B2$ requests, CSQ is stated as cost of sending a SPARQL query, CRT is denoted as cost of receiving a single result tuple, CHT is signified as cost of handling a single result tuple, BSZ is represented as binding block size, and TC is specified as number of threads utilized to query SPARQL end-points. The estimated cost function is given as the input for modified GWO in order to find the optimized cost value of the query. The explanation about modified GWO is detailed in the below section.

3.1. Web query join optimization using modified grey wolf optimization

Grey wolf optimization is a swarm intelligence optimization method that mimics the leadership hierarchy of wolves, which are known for group hunting. Generally, the grey wolfs belongs to the Canidae family, which mostly wish to live in group. The grey wolves have a strict social dominant hierarchy (leader may be a male or female) that is theoretically named as alpha (α). Mostly, the alpha is accountable for decision making and the orders of the dominant wolf follow the pack. Respectively, beta (β) represents the sub-ordinate wolves that help alpha in decision making. Beta(β) acts as an advisor to alpha and discipliner for the pack. The low ranking grey wolves are named as omega (ω) that submits all other dominant wolves. If a wolf is neither an omega or alpha nor beta, it is called delta (δ). Delta wolves dominate the omega wolves and report the status to alpha and beta wolves.

The hierarchy of wolves is theoretically modelled for developing GWO and accomplish optimization. The GWO approach is tested with the test functions that represent the exploitation and exploration characteristics compared to other swarm intelligence algorithms. Further, the GWO algorithm is successfully employed to solve several engineering optimization issues. Due to the multipurpose property, the modified GWO algorithm attempts to solve the optimization issues.

3.5.1. Overview of modified grey wolf optimization algorithm

The GWO approach mimics the social hierarchy and hunting behaviour of grey wolves. In addition to the hunting behaviour of grey wolves, group hunting is another appealing societal action of grey wolves. The GWO algorithm includes three main segments such as, encircling, hunting and attacking of prey. The step by step procedure of modified GWO is described below.

Step 1: At first, initialize the GWO parameters like design variable size Gd, search agents Gs, maximum number of iterations $iter_{max}$, and vectors a, A, C that is mathematically denoted in the Eq. (7) and (8). The value $\vec{a}$ linearly decreases from two to zero over the course of iterations.

$$\vec{A} = 2\vec{a}.rand_1 - \vec{a} \quad (7)$$

$$\vec{C} = 2.rand_2 \quad (8)$$

Where, $rand_1$ and $rand_2$ are denoted as random vectors, which ranges between [0, 1].

Step 2: Then, randomly generate the wolves based on pack that is represented in the Eq. (9).

$$Wolves = \begin{bmatrix} G_1^1 & G_2^1 & G_3^1 & \dots & G_{Gd-1}^1 & G_{Gd}^1 \\ G_1^2 & G_2^2 & G_3^2 & \dots & G_{Gd-1}^2 & G_{Gd}^2 \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ G_1^{Gs} & G_2^{Gs} & G_3^{Gs} & \dots & G_{Gd-1}^{Gs} & G_{Gd}^{Gs} \end{bmatrix} \quad (9)$$

Where, G_j^1 is denoted as initial value of the j^{th} pack of the i^{th} wolves.

Step 3: Calculate the fitness value of each hunt agent using the Eq. (10) and (11).

$$\vec{D} = \left|\vec{C}.\vec{G}_p(t) - \vec{G}(t)\right| \quad (10)$$

$$\vec{G}(t+1) = \vec{G}_p(t) - \vec{A}.\vec{D} \quad (11)$$

Where, t is denoted as number of iteration, $\vec{A}$ and $\vec{C}$ are represented as coefficient vectors, $\vec{G}_p$ is stated as position vector of the prey, and $\vec{G}$ is stated as position vector of a grey wolf.

Step 4: Determine the best hunt agent G_α, second and third best hunt agents G_β and G_δ by using the Eqs. (12)-(17).

$$\vec{D}_\alpha = \left|\vec{C}_1.\vec{G}_\alpha - \vec{G}\right| \quad (12)$$

$$\vec{D}_\beta = \left|\vec{C}_2.\vec{G}_\beta - \vec{G}\right| \quad (13)$$

$$\vec{D}_\delta = \left|\vec{C}_3.\vec{G}_\delta - \vec{G}\right| \quad (14)$$

$$\vec{G}_1 = \vec{G}_\alpha - \vec{A}_1.(\vec{D}_\alpha) \quad (15)$$

$$\vec{G}_2 = \vec{G}_\beta - \vec{A}_2.(\vec{D}_\beta) \quad (16)$$

$$\vec{G}_3 = \vec{G}_\delta - \vec{A}_3.(\vec{D}_\delta) \quad (17)$$

Step 5: Update the location of the present hunt agent by using the Eq. (18).

$$\vec{G}\ (t+1) = \frac{\vec{G}_1+\vec{G}_2+\vec{G}_3}{3} \quad (18)$$

Step 6: Evaluate the fitness value of all hunts.

Step 7: Update the value of G_α,G_β and G_δ.

Step 8: By using the best values of G_α,G_β and G_δ, alpha once again encircles the current solutions to obtain a possible solution.

Step 9: Check the stopping criteria, whether the $iter$ reaches $iter_{max}$ or not, if yes, print the current best value, or else again go to step 5.

3.5.2. Pseudo-code of modified GWO optimization algorithm

- Generate initial search agents $G_i(i = 1,2, \dots, n)$
- Initialize the vectors a, A and C
- Evaluate the fitness value of each hunt agent

  $G_\alpha \rightarrow$ Best hunt agent
  $G_\beta \rightarrow$ Second best hunt agent
  $G_\delta \rightarrow$ Third best hunt agent
  $Iter = 1$
- **Repeat**
- **For** $i = 1: Gs$ (grey wolf pack size)
- Update the location of current hunt agent
- **End for**
- Evaluate the fitness value of all hunt agents
- Update the value of G_α,G_β and G_δ.
- $G_\alpha = \min(\ G_\alpha, G_\beta$ and $G_\delta)$
- Update the vectors a, A and C
- $Iter = Iter + 1$
- Unit $Iter \geq =$ maximum number of iterations (stopping criteria)
- Output G_α
- **End**

4. Experimental result and discussion

In this research study, eclipse java 1.8 apache Jena was used for experimental simulation with 3.2 GHz and i5 processor. In order to analyse the effectiveness of proposed methodology (modified GWO), the performance of proposed methodology was compared with the existing methodologies: PECA [11], PEDA [11], RDF-3X [11], gStore [15], and Legato [16] on LUBM 10000 and DOREMUS datasets. The performance of the proposed methodology was evaluated in terms of execution time and memory usage. The performance metric determined as the regular measurement of outcome that creates the reliable information about the effectiveness of the proposed methodology. Here,

the relationship between the input and output variables of the proposed modified-GWO methodology is understood by using the performance measures like execution time, precision, recall, and f-measure.

4.1 Performance measure

Performance measure is determined as the measurement of experimental outcome that develops reliable information about the effectiveness of proposed system. The relationship between the input values and output values of the proposed system was understood by utilizing the performance measures like execution time, precision, recall, and f-measure. The formula to evaluate precision, recall, and f-measure are given in the Eqs. (19), (20), and (21).

$$Precision = \frac{TP}{TP+FP} \times 100 \quad (19)$$

$$Recall = \frac{TP}{TP+FN} \times 100 \quad (20)$$

$$F - measure = \frac{2TP}{2TP+FP+FN} 100 \quad (21)$$

Where, FP is specified as false positive, TN is indicated as true negative, TP is stated as true positive, and FN is represented as false negative.

4.2 Quantitative analysis on LUBM 10000 dataset by means of execution time

In this section, LUBM 10000 dataset is used for evaluating the performance of proposed and existing methodologies. The LUBM 10000 dataset consists of repeatable synthetic data, customizable data, university domain ontology and a set of test quires (fourteen). In Table 1, the proposed methodology performance evaluated in light of execution time and compared with the existing methodologies such as PECA [11], PEDA [11], and RDF-3X [11]. Table 1 shows that the proposed methodology works faster than the existing methodologies even when the query graph is complex such as, query 1, query 2, query 3, and query 7. Since, these queries do not contain any selective triple patterns, query graph structure is complex and the search space of these queries is very large.

The modified GWO has the advantage of parallel processing and reduce query response time effectively related to a centralized system. If the queries (4, 5 and 6) contains selective triple patterns, the search space becomes small. The centralized system of RDF-3X, PECA, and PEDA is faster than the proposed methodology in a few queries, since the proposed methodology spends more communication cost between the dissimilar machines. These queries only spend less than three seconds in both the existing and proposed method. However, for some challenging queries (1, 2, 3 and 7), the modified GWO outperformed the existing methodologies significantly. The graphical comparison of proposed and existing methodology is denoted in Fig. 2.

Table 1. Comparative analysis of proposed and existing system by means of execution time

Execution time (milliseconds)				
Number of queries	RDF-3X [11]	PECA [11]	PEDA [11]	**Proposed method (modified-GWO)**
1	10,840,47	3,26,167	3,09,361	**45**
2	81,373	23,685	23,685	**60**
3	72, 257	10,239	10,368	**38**
4	7	753	753	**33**
5	6	125	125	**43**
6	355	3388	1914	**79**
7	1,46,325	1,43,779	46123	**82**

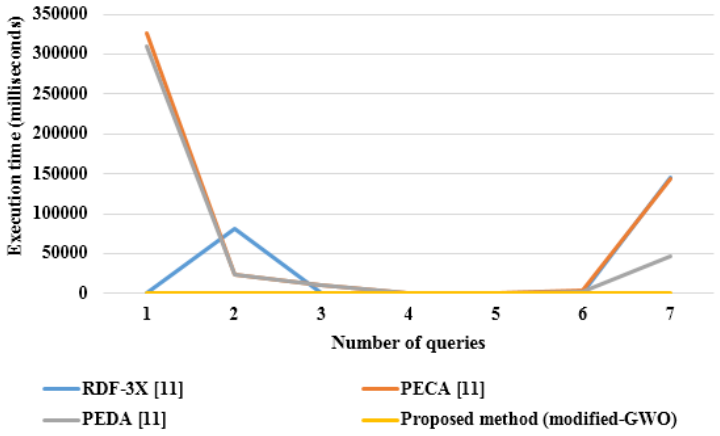

Figure.2 Graphical comparison of proposed and existing methodology

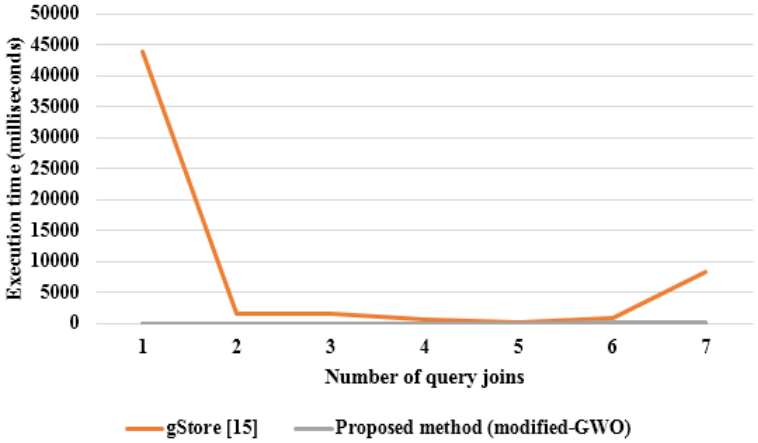

Figure.3 Graphical comparison of proposed and existing methodology

Table 2. Comparative analysis of proposed and existing system by means of execution time

Execution time (milliseconds)		
Number of query joins	gStore [15]	**Proposed method (modified-GWO)**
1	43832	**45**
2	1563	**60**
3	1491	**38**
4	680	**33**
5	131	**43**
6	828	**79**
7	8301	**82**

Table 3. Performance comparison of proposed and existing method in terms of precision, recall, and f-measure

Methods	**Data**	**Precision (%)**	**Recall (%)**	**F-measure (%)**
Legato [16]	HT	98	87	90
	FPT	90	78	83
Proposed method (modified-GWO)	**HT**	**100**	**95**	**90**
	FPT	**90**	**85**	**90**

Similarly, Table 2 depicts about the performance of proposed methodology (modified GWO) and existing methodology (gStore [15]) in terms of execution time by using the number of query joins. The execution time of proposed approach (modified GWO) is very low compared to the existing methodologies because the proposed method quickly evaluates the join order and also optimizes the join cost effectively that reduces the complexity of the system. The graphical representation of the execution time is denoted in Fig. 3. The experimental result demonstrates that the modified GWO achieved better execution time for a number of queries. The execution time decreases for a small number of queries, but after some certain optimal point, the execution time increases when the number of queries increases.

4.3 Quantitative analysis on DOREMUS 2017 dataset by means of precision, recall and f-measure

In this sub-section, performance of the proposed methodology (modified GWO) is compared with an existing methodology (Legato [16]) in terms of precision, recall, and f-measure. Table 3 states that the proposed method achieved a better result by means of precision, recall, and f-measure related to an existing system. The DOREMUS 2017 dataset contains two important sections: HT and FPT, where the result is averagely calculated. In HT phase, the proposed system achieved 100% precision, 95% recall, and 90% f-measure, which is superior related to the existing system. Similarly, in FPT phase, the proposed system attained 90% of precision, 85% of recall, and 90% of f-measure. Compared to existing system (Legato [16]), the proposed system achieved good performance. From Tables 1, 2, and 3, it is clear that the proposed method achieved low execution time, precision, recall, and f-measure than the other existing methodologies in LUBM 10000 and DOREMUS 2017 datasets.

5. Conclusion

In this research study, a new query optimization algorithm (modified GWO) was developed to improve the performance of query evaluation and start-up cost. In addition, this research paper also detailed about the query optimization based on reordering the triple pattern in the main memory of RDF data. Compared to the existing methodologies, the proposed method delivered an effective performance by means of quantitative and comparative analysis. From the experimental investigation, the proposed methodology improved

the execution time (2-5 minutes) related to the existing methodologies on LUBM 10000 dataset. Similarly, the proposed methodology improved precision, recall, and f-measure around 2-7% related to the existing system. In future work, a hybrid optimization algorithm will be developed in order to extend the current framework on dissimilar query engines and triple patterns.

References

[1] M. Meier, M. Schmidt, F. Wei, and G. Lausen, "Semantic query optimization in the presence of types", *Journal of Computer and System Sciences*, Vol.79, No.6, pp.937-957, 2013.

[2] A. Letelier, J. Pérez, R. Pichler, and S. Skritek, "Static analysis and optimization of semantic web queries", *ACM Transactions on Database Systems (TODS)*, Vol.38, No.4, pp.25, 2013.

[3] R. Taelman, J. Van Herwegen, M. Vander Sande, and R. Verborgh, "Comunica: a modular SPARQL query engine for the web", *In International Semantic Web Conference*, pp.239-255, 2018.

[4] O. Hogenboom, F. Frasincar, and U. Kaymak, "Ant colony optimization for RDF chain queries for decision support", *Expert Systems with Applications*, Vol.40, No.5, pp.1555-1563, 2013.

[5] J.J. Jung, "Semantic Optimization of Query Transformation in a large-scale peer-to-peer network", *Neurocomputing*, Vol.88, pp.36-41, 2012.

[6] A. Schätzle, M. Przyjaciel-Zablocki, S. Skilevic, and G. Lausen, "S2RDF: RDF querying with SPARQL on spark", In: *Proc. of the VLDB Endowment*, Vol.9, No.10, pp.804-815, 2016.

[7] Z. Chouiref, A. Belkhir, K. Benouaret, and A. Hadjali, "A fuzzy framework for efficient user-centric Web service selection", *Applied Soft Computing*, Vol.41, pp.51-65, 2016.

[8] W. Zheng, L. Zou, W. Peng, X. Yan, S. Song, and D. Zhao, "Semantic SPARQL similarity search over RDF knowledge graphs", In: *Proc. of the VLDB Endowment*, Vol.9, No.11, pp.840-851, 2016.

[9] T. Zhao, C. Zhang, L. Anselin, W. Li, and K. Chen, "A parallel approach for improving Geo-SPARQL query performance", *International Journal of Digital Earth*, Vol.8, No.5, pp.383-402, 2015.

[10] A. Boukorca, L. Bellatreche, S.A.B. Senouci, and Z. Faget, "Coupling materialized view selection to multi query optimization: hyper graph approach", *International Journal of Data Warehousing and Mining*, Vol.11, No.2, pp.62-84, 2015.

[11] P. Peng, L. Zou, M.T. Özsu, L. Chen, and D. Zhao, "Processing SPARQL queries over distributed RDF graphs", *The VLDB Journal-The International Journal on Very Large Data Bases*, Vol.25, No.2, pp.243-268, 2016.

[12] R. Harbi, I. Abdelaziz, P. Kalnis, N. Mamoulis, Y. Ebrahim, and M. Sahli, "Accelerating SPARQL queries by exploiting hash-based locality and adaptive partitioning", *The VLDB Journal-The International Journal on Very Large Data Bases*, Vol.25, No.3, pp.355-380, 2016.

[13] E.G. Kalayci, T.E. Kalayci, and D. Birant, "An ant colony optimisation approach for optimising SPARQL queries by reordering triple patterns", *Information Systems*, Vol.50, pp.51-68, 2015.

[14] P. Peng, L. Zou, and Z. Qin, "Answering top-K query combined keywords and structural queries on RDF graphs", *Information Systems*, Vol.67, pp.19-35, 2017.

[15] L. Zou, M.T. Özsu, L. Chen, X. Shen, R. Huang, and D. Zhao, "gStore: a graph-based SPARQL query engine", *The VLDB Journal—The International Journal on Very Large Data Bases*, Vol.23, No.4, pp.565-590, 2014.

[16] M. Achichi, Z. Bellahsene, M.B. Ellefi, and K. Todorov, "Linking and disambiguating entities across heterogeneous RDF graphs", *Journal of Web Semantics*, Vol.55, pp.108-121, 2019.

[17] M. Saleem, A. Potocki, T. Soru, O. Hartig, and A.C.N. Ngomo, "Costfed: Cost-based query optimization for sparql endpoint federation", *Procedia Computer Science*, Vol.137, pp.163-174, 2018.