I.J. Modern Education and Computer Science, **2018, 3, 38-46**
Published Online March 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2018.03.05

Specific Queries Optimization Using Jaya Approach

Sahil Saharan
Department of Computer Applications, National Institute of Technology, Kurukshetra, India
Email: sahil.saharan_1376@nitkkr.ac.in

J.S. Lather
Department of Electrical Engineering, National Institute of Technology, Kurukshetra, India
Email: jslather@nitkkr.ac.in

R. Radhakrishnan
Department of Computer Science and Engineering, ABES Ghaziabad (Uttar Pradesh), India
Email: ramaswamiradhakrishnan@gmail.com

Received: 24 November 2017; Accepted: 15 December 2017; Published: 08 March 2018

***Abstract*—**The Fast query engine is a requirement as a supporting tool for the semantic web technology application such as Electronic Commerce environ. As the large data is represented using the effective data representation called RDF. The focus of this paper is to optimize the specific type of the query called Cyclic query and star query on main-memory RDF data model using ARQ query engine of Jena. For the considered problem, we ruminate a Jaya algorithm for rearrangement of the order of triple pattern and also compare the results with an already proposed approach in the literature. The evaluation result shows that Jaya performs better in terms of execution time in comparison to Ant Colony Optimization.

***Index Terms*—**Resource Description Framework (RDF), Query Optimization, Jaya, SPARQL, Reordering triple patterns, Semantic Web

I. Introduction

With the increasing popularity of Semantic Web [1], overwhelmed data stored over the many heterogeneous, yet interconnected resources which are generally represented using RDF representation. RDF(Resource Description Framework) [2] is a framework to describe and interchange meta-data, which empower machine-interpretable by providing contextual information of the data i.e. meta-data of data. So this interconnected data instead of pages, fulfill the complex information required in an efficient manner than that of the current web. The technologies of Semantic Web explore different RDF sources to meet very specific need of information. To execute queries W3C's provides a protocol called SPARQL [3, 4]. The fast query engine is vital requirement for SPARQL queries to fetch the results of the user query from the widely distributed RDF data in real-time environments. Execute the query with lower execution time requires optimization of triple patterns in SPARQL query. An efficient optimization algorithm for triple patterns can, therefore, contribute to efficient execution time. A list of optimization techniques has been proposed already as a solution to the current problem such as 2-Phase Optimization (2PO) algorithm [5], genetic algorithm (GA) [6], Ant Colony Optimization (ACO) algorithm [7] etc.

The proposal of the current study has been encouraged by the optimization of query paths in traditional databases. For dynamic semantic web, which support the change in the data time to time and needs a better optimization strategy; Jaya [8] is an alternative to the current approaches and have already been applied to the different field like: constrained mechanical design optimization problems [9], optimization of machining performance characteristics during the turning of CFRP composites [10], optimum power flow (OPF) problems [11], optimization of traditional machining process named surface grinding [12], dimensional optimization of a micro-channel heat sink [13], optimization of combined economic emission dispatch solution [14] etc.

The behavior of the proposed algorithm is such that it works in a continuous manner and allows accommodating to changes in the data in real time. Through this paper, we are introducing Jaya applicability to the RDF query optimization in specific query forms over a single data source.

The rest of paper is organized as follows. Section 2 introduced a literature review. Section 3 introduces the concepts for RDF and SPARQL queries. Additionally, Jena API and the ARQ engine are also introduced. The BGP construction from where clause predicates of SPARQL queries are illustrated. Section 4 introduced Jaya optimization algorithm. Section 5 describes the problem. Section 6 describes the observations using experimental study of the proposed algorithm and its comparison in terms of execution time, fitness function

value and solution quality. Finally, in later sections the discussion which is followed by a conclusion.

II. Related Work

Regarding RDF databases context, literature study of rearrangement of the order of triple patterns and other key factors for optimization of join ordering in RDF databases is discussed here.

Stuckenschmidt et al. [15] presented the first solution for the reordering triple pattern (by optimizing the order of path expression) using hybrid algorithm two-phase optimization (2PO) (a combination iterative improvement and simulated annealing) over RDF Cyclic query by extending the Sesame system [16]. They considered the issue of integrated access to distributed RDF repositories from a practical point and provide flexibility, freshness, and independence of data as advantages over the centralized approach. Shironoshita et al. [17] introduce an algorithm for cardinality estimation of queries over ontology models of data. The introduced algorithm is an important section for building an efficient query engine for distributed and heterogeneous data sources.

Maduko et al. [18] presented a framework using pattern-based summarization to estimate cardinality. The strategy proposed follows 1) pattern and their sub-pattern can have almost equal frequencies and 2) previous knowledge of pattern's importance. For results calculation, they use Dynamic programming with two greedy solutions over real world and synthetic datasets. Stocker et al. [19] presented a static optimization approach using reordering triple patterns of the BGP. For joined triple patterns, they presented a set of heuristics for the selectivity estimation and summary statistics of RDF data. Ruckhaus et al. [20] presented a hybrid cost model and dynamic programming based optimization approach for queries optimization and named it as deductive ontology base (DOB). Additionally, they used an adaptive sampling technique to estimate the cardinality of intermediate inferred facts.

Next, Hogenboom et al. [6] introduced a GA based optimization approach and experimental results over datasets presented the effectiveness of proposed approach over 2PO in case of large Chain queries and for small queries having 10 predicates, 2PO presents better results. Neumann and Weikum [21] presented an RDF-3X engine and dynamic programming for optimal plan generation and selectivity histograms to estimate the cost of joins between triple patterns. To meet the scalability when querying a large number of triple patterns, Neumann and Weikum [22] introduced a very light-weight strategy for sideways information passing between separate joins and used aggregated statistics to accurately estimate the selectivity of join-ordering. Kaoudi et al. [23] studied the query optimization problem on top of distributed hash tables and using three greedy algorithms. These 3 algorithms are used to optimize the intermediate relations generated using the selectivity-based heuristics named: naive static algorithm, semi–naive static algorithm ,and dynamic algorithm. Neumann and Moerkotte [24] proposed new accurate cardinality estimation for RDF databases based on characteristic sets. They show experimentally that new method is superior to the estimation methods of commercial DBMSs and RDF-3X[21].

For multi-join ordering problem, Ouyang et al. [25] introduce a strategy using a genetic algorithm (GA) to optimize the execution plan and for generating the best plan, they use a bushy tree. Hogenboom et al. [7] introduce a new solution to the current problem of query optimization using ACO algorithm over Chain query and shows best results when compared with [21] and 2PO [15]. The heuristic used by this experiment is introduced by [14]. In these solutions for query optimization, Gomathi et al.[26] also contributes to an efficient algorithm named adaptive Cuckoo search (ACS) to an optimal query plan for large RDF data. Next, Kalayci et al. [27] proposed a new optimization strategy using Ant Colony Optimization (ACO) algorithm for reordering triple patterns and they used statistics for selectivity estimation proposed by [19] with some modification.

Meimaris and Papastefanatos [35] present a new approach of join reordering that converts a query into a multidimensional vector space and performs distance-based optimisation.

They compared the results with existing approaches and proved that the proposed approach is better than existing.

The prior work studies suggest that there is still need for better heuristic and so we are investigating a new algorithm called: Jaya. The efficiency of the proposed approach can be seen through the experimental results.

III. RDF and SPARQL

RDF is data model which has a flexibility of schema-free. RDF develops major momentum in different areas like knowledge-management communities by collecting facts about entities and their relations using RDF representation. RDF data can be shown by entity-relationship graph and each triple of RDF can be represented as a node-arc-node link [28].

SPARQL query depends on matching of graph patterns with triple patterns of the query and SPARQL queries generally made of triple patterns known as BGPs (Basic Graph Patterns) [4]. The structure of triple pattern is such that it contains <s, p, o>, that could be concrete or variable [19].

Jena[29] is a framework for Semantic Web applications that is capable to store and manipulate in-memory RDF data. ARQ [30] as query engine used in Jena for querying RDF data. We are using ARQ in the proposed approach.

The study traditional database joins ordering operation of SQL queries enforces to study the rearrangement of the order of triple pattern of SPARQL query and this type of approach also make a greater impact on the execution time of the query.

We are given an example of Cyclic query to make clarity and importance about arrangement of order of

BGP (i.e. where clause of the SPARQL query containing the different pattern of triples) in Table 1 and we have taken this Cyclic query with six triple patterns from the LUBM [31] dataset provided by the Lehigh University. We have assigned sequence number from 0 to 5 to each triple pattern.

In the query example, we provide a numbering from 0 to 5 to each triple pattern. For the given order [0, 1, 2, 3, 4, 5] in Table 1, Jaya algorithm execute this 6 cyclic query with an execution time of 231714ms, whereas the execution time for [3, 1, 2, 0, 4, 5] order of listing 2 is 71128ms and for the order [0, 5, 2, 3, 4, 1] of listing 3, the execution time was 65ms. Here, [0, 5, 2, 3, 4, 1] is the optimal order that have optimal execution time.

Table 1. BGP of an example Cyclic query[31]

0	?X rdf:type ub:GraduateStudent.
1	?Y rdf:type ub:University.
2	?Z rdf:type ub:Department.
3	?X ub:memberOf ?Z.
4	?Z ub:subOrganizationOf ?Y.
5	?X ub:undergraduateDegreeFrom ?Y.

Through the example given above, we observed that the simple query of SPARQL in ARQ provides longer execution, [34] suggested a solution: adding more selective part of the query first, makes an impact on the execution time of the query.

IV. Optimization Algorithm: Jaya

The first phase of the introduced strategy consider three major parts; first is to evaluate the number of concrete and variable element matching, second is to estimate the join values using selectivity based on estimated cardinality and third is to the construction of cost matrix using these estimated join values and estimated cardinality values. During the second phase of the proposed strategy, apply the Jaya algorithm over the constructed cost matrix for the rearrangement of the order of triple patterns.

Jaya [8] algorithm procedure is such that: using the upper and lower bound of the process variables, initially generate the random solution. Then, update the variable of every solution by Equation (1). The best candidate indicates the best value of the objective function from the whole candidate solutions and worst candidate indicate the worst value of the objective function from the whole candidate solutions. If $S_{i,j,k}$ indicate the value of the j-th variable for the k-th candidate at i-th iteration. The equation is:

$$S_{i+1,j,k} = S_{i,j,k} + r_{i,j,1}(S_{i,j,best} - abs(S_{i,j,k})) - r_{i,j,2}(S_{i,j,worst} - abs(S_{i,j,k})) \quad (1)$$

$r_{i,j,1}$ and $r_{i,j,2}$ represents random number chosen from the range [0,1] for the jth variable at the ith iteration. If the modified value is better function value than the previous value then use the modified value otherwise avoid it. The equation (1) has an expression $r_{i,j,1}(S_{i,j,best} - abs(S_{i,j,k}))$ which helps to move the solution to the best solution and $r_{i,j,2}(S_{i,j,worst} - abs(S_{i,j,k}))$ helps in avoiding the worst solution. Random number and absolute of the variable ensures good exploration.

Algorithm: Jaya algorithm[8]

Initialize the population size, number of variable and stoping criteria
Until stoping criteria met
  Evaluate the fitness for each population
  Evaluate best and worst solution in the population
  Update the solution using:
  $Y'_{j,k,i} = Y_{j,k,i} + rand_{1,j,i}(Y_{j,best,i} - Y_{j,k,i}) - rand_{2,j,i}(Y_{j,worst,i} - Y_{j,k,i})$
  Update the solution
  No update in the solution
Repeat
Report optimum result

Discrete conversion of Jaya

The originally given for the continuous problem, but for our use we have to use a discrete version of the algorithm so that we can use Jaya algorithm in the query optimization problem of query optimization. So, to convert it in to a discrete problem, we convert the equation into such a way:

$$Y'_{j,k,i} = Y_{j,k,i} \circ (Y_{j,best,i} \circ Y_{j,k,i}) \circ (Y_{j,worst,i} \circ Y_{j,k,i}) \quad (2)$$

Operator is the crossover operator we used here the PMX crossover [33] operator and finally to eliminate the duplicate value of the final result, we use here the mutation operation which will generate the final result. The intermediate result will use mutation $Y = \bullet(Y_{j,best,i} \circ Y_{j,k,i})$ and the mutation operation is the swapping of the one randomly generated triple pattern of the BGP with the current generated triple pattern if the random generated value is less than the mutation probability.

Algorithm: DJaya algorithm

Initialize the population size, number of variable and stoping criteria
Until stoping criteria met
  Evaluate the fitness for each population
  Evaluate best and worst solution in the population
  Update the solution using:
  $Y'_{j,k,i} = Y_{j,k,i} \circ (Y_{j,best,i} \circ Y_{j,k,i}) \circ (Y_{j,worst,i} \circ Y_{j,k,i})$
  $Y = \bullet(Y_{j,best,i} \circ Y_{j,k,i})$ % mutation in above intermediate result
  Update the solution
  No update in the solution
Repeat
Report optimum result

V. Problem Description

We are considering the problem of optimization of the triple patterns of SPARQL queries using rearrangement of the order of these triple patterns. For rearrangement, we are considering Jaya algorithm.

1. Used Cost Model

From the different triple patterns of BGP, we constructed a complete digraph [27] of the given Ɓ where nodes of digraph which represent the triple patterns. And the arc between nodes is resulted as a join between two triple patterns whenever join occur between two triple patterns otherwise it is resulted as a Cartesian product between triple patterns and resulted as 1(which is the upper limit of selectivity(range from 0 to1)). The arc is computed using addition of selectivity of most selective triple pattern 1st pattern and join estimation of two patterns whenever some join occur between two triple patterns. The digraph construction is important due to the fact that selectivity of one triple pattern is different than that of other. Rearrangement of order of triple pattern doesn't affect join between two triple patterns.

For the calculation of values of arc values, firstly consider the two triple patterns. The different triple patterns have different number of bound and unbound elements. According to these triple patterns, calculate bound and unbound elements of each triple patterns and based on these bound and unbound elements, calculate the different joins between these triple patterns. Each bound and unbound component have its matching triple patterns and different bound and unbound elements join value depends on the cardinality and the estimated selectivity values. To evaluate these values the steps to be followed are:

1. Evaluate the number of concrete and variable element matching.
2. Evaluate the estimated selectivity.

Next, the explanation of the above two steps are as follow:

1. Evaluate the number of concrete and variable element matching.

To evaluate the number of concrete and variable element matching. To make it understandable we are using an example:

?GraduateStudent rdf:type ub:GraduateStudent

Above triple pattern has two concrete elements rdf:type ub:GraduateStudent and one variable and due to these two concrete elements the use the procedure [27] and evaluate cardinality for each concrete element using GSH[32]. In other words, we can say that find the number of triple pattern of each concrete element at the different position(s, p, and o) of the triple pattern using GSH and preserve the smallest number of triple pattern as the new cardinality. And finally, use the lowest value as the resulted cardinality. Note that, GSH only provide cardinality for one concrete element. For understanding, use the following example:

? GraduateStudent ub:memberOf ?Depatment

The example shows that at only predicate position one concrete element is present, so for this only cardinality can be provided by GSH.

2. Evaluate the estimated selectivity and join[27].

To evaluate the join between the concrete and the variable elements of the given two triple patterns different steps are used:

Step1: Use the two intermediate triple patterns of the given query.

Step 2: Now find the different that match between the two intermediate triple patterns.

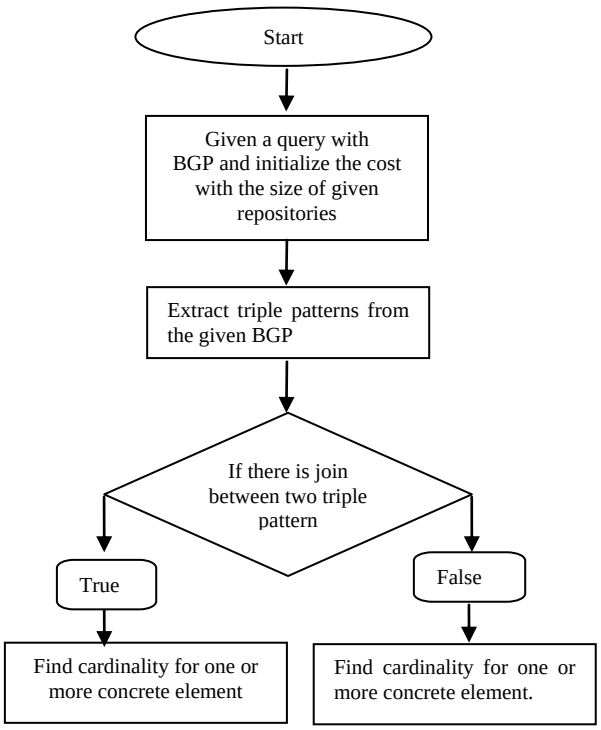
Step 3: Now find the ranks of each matching and add them. Different ranks are a modified version of [19].

Steps 4: Assign the cost as 32 [19] and then subtract the different added rank value. And factor using resulted cost value/original cost.

Step 5: Finally, calculate the join value between two intermediate triple patterns. (Fig 1)

3. Jaya based Query Optimization problem

We have SPARQL query with different triple patterns, we have to generate an execution plan that provides smaller execution time through rearrangement of the order of triple patterns. The solution is to construct the complete digraph according to BGP of given SPARQL query. Then use Jaya algorithm for rearrangement of the order of triple patterns.



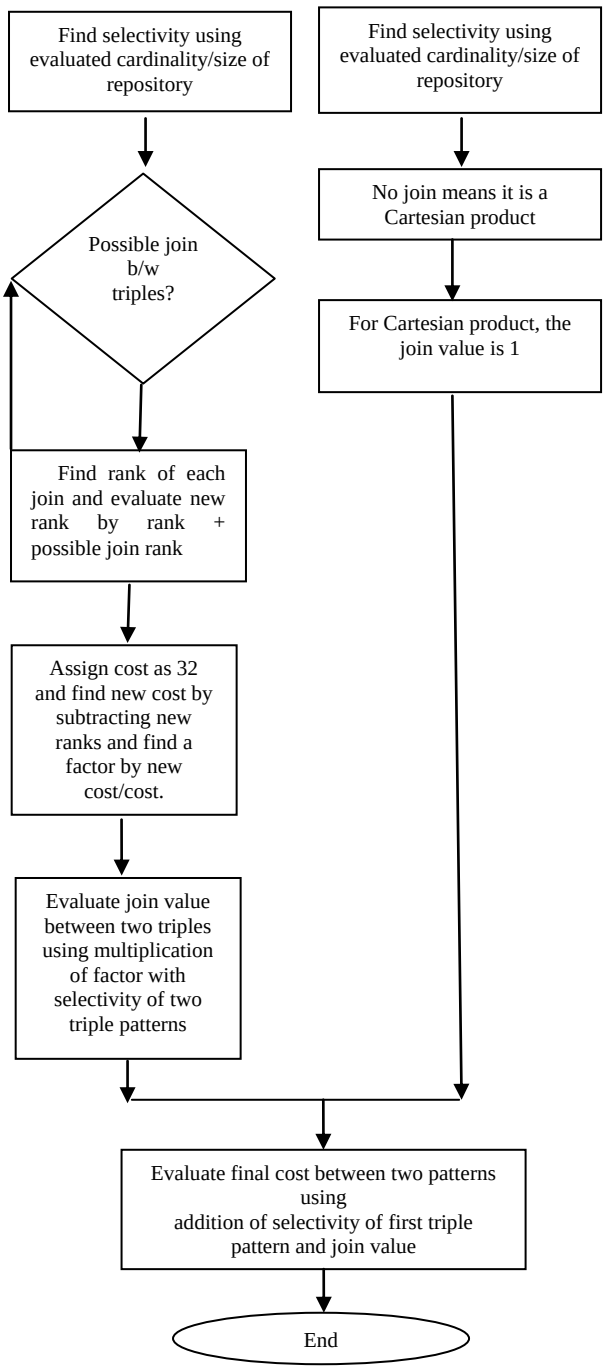


Fig.1. Flow Chart for the evaluation of join

VI. Experimental Results

1. Setup, Dataset detail and Used Queries

The experimental results were taken over the machine with 32-bit Oracle JDK virtual machine running on Intel(R) Core(TM) i7 -3770M CPU @ 3.40 GHz, 32-bit with 4 GB RAM and Window 10 OS.

We are using 162923 triples RDF dataset which was taken from LUBM. LUBM provides OWL ontology for university domain. We are considering only Cyclic and star queries for our experiment with several triple patterns i.e. 4, 6, 8 and 10 which are constructed from LUBM. The selection of these triple patterns is due to the fact that if the number of the triple patterns are small then sometimes they will address no optimization at all, that's why we considered the large triple patterns.

According to our choice for this experiment, we use synthetically constructed queries [27] over LUBM having 4, 6, 8 and 10 triple patterns with Cyclic and star as shape. Queries are not bound to Cyclic and star shapes, and diverse types of the queries with different triple patterns and complex graph queries are planned.

Here, just one query of every different type has been taken into account to comparative evaluation of the proposed method. The selected queries i.e. Cyclic, Star as are sufficient enough to benchmark the results.

Jaya algorithm is independent of the algorithm-specific parameters and only uses some common parameters like population size and a number of generations. For the test purpose, we apply a series of 10 test over the single query of every different type and different algorithm.

2. Used Abbreviations and their Detail

1. EWO i.e. Executions Without optimization.
2. AS-MM i.e. Ant System using Modified Method.
3. EAS-MM i.e. Elitist Ant System using Modified Method.
4. MMAS-MM i.e. MAX–MIN Ant System algorithm using Modified Method.
5. Jaya-MM i.e. Jaya using Modified Method.

Here, Modified Method is name given to MVC[27] because it is a modified method. For Jaya algorithm, we are using different calibration like starting node as random with population size and iteration size as 100 and 100 respectively. For the result evaluation, consider the execution time of the query as a sum of the time taken by optimization, execution and population time (time for the iteration of the result set in millisecond) for all the algorithms except EWO.

For the comparison, we have taken different algorithms: (1) EWO (2) ACO different versions (AS, EAS, MMAS) [27] (3) Our proposed approach.

3. The results are compared on the basis of:

1. Triple patterns 4, 6, 8, 10 for Cyclic Type Query.
2. Triple patterns 4, 6, 8, 10 for Star Type Query.

1). Cyclic queries

Table 2 shows results of Cyclic queries consist of various triple patterns wherein query execution time is considered in milliseconds. Table results show that Jaya-

MM produces best results in case of 4 triple patterns. Additionally, EAS-MM, MMAS-MM, AS-MM represents nearby result to the Jaya-MM and EWO provides worst results. In case of six triple patterns, Jaya-MM shows best results and are better than ACO variations (MMAS-MM, AS-MM, EAS-MM) and EWO. In 8 triple patterns, proposed Jaya algorithm gives the best results over others. The result provided by EWO is again worst. Also, for 10 triple patterns, Jaya-MM providing the best execution time and in addition, MMAS-MM, AS-MM, EAS-MM is better than EWO. On average, we find that proposed algorithm provides better results than that of ACO versions.

In the comparison, we have compared the different versions of ACO. MMAS-MM is an algorithm that has proved its applicability to different applications and is shown that MMAS-MM is better than other two AS-MM and EAS-MM. But for the Cyclic query with 4, 6 and 8 and 10 triple patterns, MMAS-MM does not show a better result than AS-MM and EAS-MM.

Fig 2 represents the pictorial representation of the results listed in Table 2, where the *x*-axis represents different triple patterns of Cyclic queries and the *y*-axis represents query execution time.

Fitness value is the value when we apply an algorithm to the digraph which is result of the sub-section 1 of the Section V. Applied algorithms produce values according to the dimension of SPARQL query triple patterns.

Fig 3 provides the fitness value for all different triple patterns of Cyclic queries here *x*-axis represents different algorithm and y-axis represent fitness value.

Table 2. Execution time for cyclic query with different triple patterns

	Four Cyclic	**Six Cyclic**	**Eight Cyclic**	**Ten Cyclic**
EWO	125	236486	42444	149032
AS-MM	43	75	450	5341
EAS-MM	40	81	456	5066
MMAS-MM	46	84	459	5305
Jaya-MM	39	65	425	3865

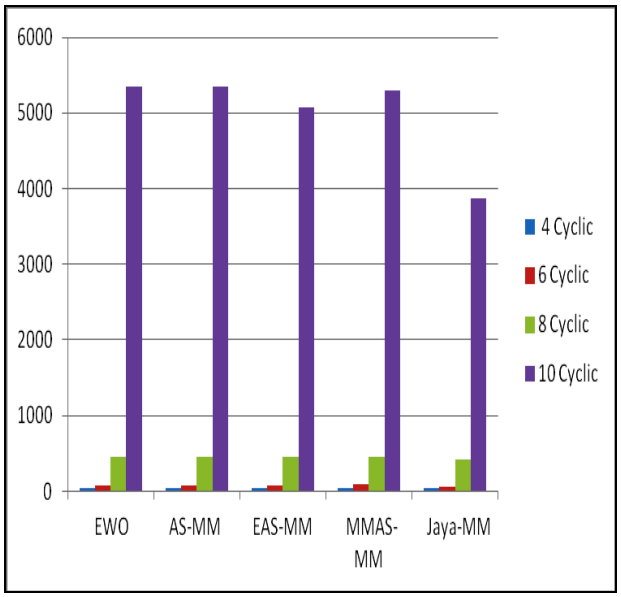


Fig.2. Execution times for the Cyclic queries different triple patterns

2). Star Queries

Table 3 presents the results of the star type queries with the different triple pattern where for the 4 triple patterns Jaya-MM shows the best results and is better when compare with ACO versions (AS-MM, EAS-MM,

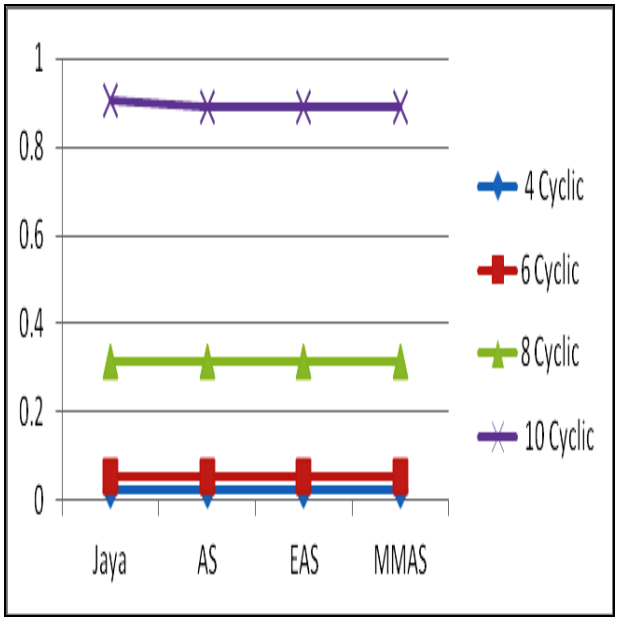


Fig.3. Fitness value Vs Algorithm on different triple patterns Cyclic Queries

MMAS-MM) and EWO. In case of 6 triple patterns, Jaya-MM provides the best result. Results of Jaya-MM is better when compare with ACO different versions and EWO. For 8 triple patterns, Jaya-MM gives good execution time over ACO versions and EWO. For 10 triple patterns, Jaya-MM presents the best result and Jaya-MM result is better than all three versions of ACO. For this case, EWO result is better than the result of MMAS-MM.

Here also, we have compared the different versions of ACO. For the star query with 4, 6, 8 and 10 triple patterns, MMAS-MM doesn't show the better result than AS-MM and EAS-MM. This reason for this is the extra computational load of MMAS-MM and inapplicability of MMAS-MM to a particular problem.

Fig 4 represents the graph of table 3 where the x-axis represents the different triple patterns of Star queries and the y-axis represents the execution time of the queries.

Fig 5 provides the fitness value for all different triple patterns of star queries, here *x*-axis represents different algorithm and y-axis represent fitness value

Table 3. Execution time for Star query with different triple patterns

	Four Star	Six Star	Eight Star	Ten Star
EWO	606	159	522	259
AS-MM	450	121	281	240
EAS-MM	410	125	284	247
MMAS-MM	456	131	300	284
Jaya-MM	408	103	250	168

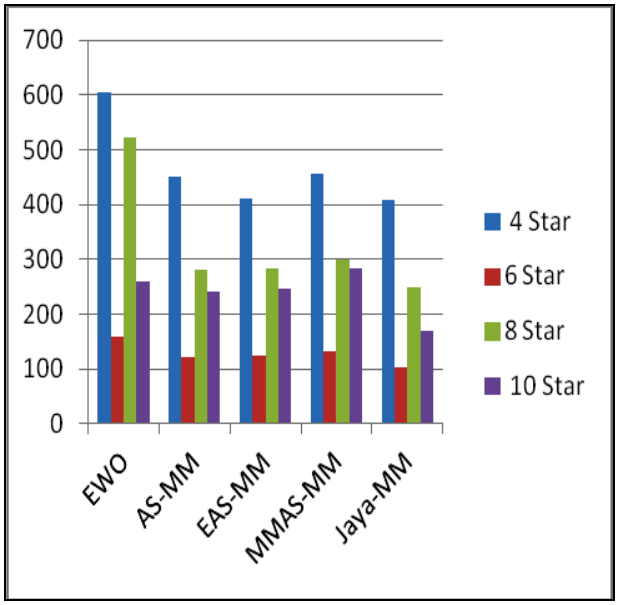


Fig.4. Execution times for the Star queries different triple patterns

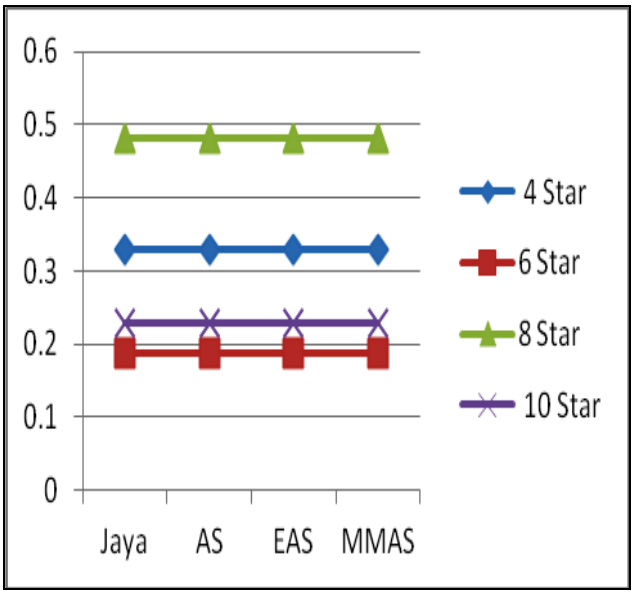


Fig.5. Fitness value Vs Algorithm on different triple patterns Star Queries

VII. Discussion and Conclusion

The results show execution time and Jaya-MM provides best execution time in comparison to ACO all versions. Also, Jaya performs better for all the queries when compare to EWO results.

In this paper, the on-disk graph implementation of BGP has not been considered since our work focuses on static optimization of BGP using main memory using Jena ARQ engine.

This paper proves the success of the proposed approach by means of execution time in comparison to the ACO different versions in semantic web SPARQL query optimization over different queries forms with several triple patterns using ARQ query engine.

Through this paper, we are discussed the query optimization using rearrangement of the order of triple pattern over the main memory RDF data model. The scope of the paper can be extended to other frameworks and query engines over different queries forms with different triple patterns. Additionally, other heuristic approaches for query optimization can be applied to the presented approach in future. Exploring better estimation techniques for the proposed approach is also another important future work.

References

[1] T. Berners-Lee, J. Hendler, O. Lassila, "The semantic web", Sci. Am. vol 284, no.5, pp: 34–43, 2001.
[2] Frank Manola and Eric Miller, "RDF primer", 2004.
[3] S. Harris, A. Seaborne, "SPARQL1.1querylanguage" – W3C working draft 05 January 2012.
[4] G.H.L. Fletcher, "An algebra for basic graph patterns", in: Proceedings of the Workshop on Logic in Databases, 2008.

[5] H. Stuckenschmidt, R. Vdovjak, J. Broekstra, G. Houben, "Towards distributed processing of RDF path queries", Int. J. Web Eng. Technol. vol.2, no.2/3, pp.207–230, 2005.
[6] A. Hogenboom, V. Milea, F. Frasincar, U. Kaymak, "RCQ-GA: RDF Chain query optimization using genetic algorithms", in: Proceedings of the 10th International Conference on EC-Web, 2, pp:181–192, 2009.
[7] A. Hogenboom, F. Frasincar, U. Kaymak, "Ant colony optimization for RDF Chain queries for decision support", Expert Syst. Appl. Vol.40, no.5, 2013.
[8] R. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems". International Journal of Industrial Engineering Computations, vol. 7,no.1,pp:19-34, 2016.
[9] R.V. Rao, G.G. Waghmare. "A new optimization algorithm for solving complex constrained design optimization problems". Engineering Optimization.vol.49, no.1,pp:60-83, 2017.
[10] Abhishek K, Kumar VR, Datta S, Mahapatra SS. "Application of JAYA algorithm for the optimization of machining performance characteristics during the turning of CFRP (epoxy) composites: comparison with TLBO, GA, and ICA". Engineering with Computers. pp.1-9, 2016.
[11] Warid W, Hizam H, Mariun N, Abdul-Wahab NI. "Optimal Power Flow Using the Jaya Algorithm". Energies.vol.9, no.9,pp:678, 2016.
[12] Rao RV, Rai DP, Balic J. "Surface Grinding Process Optimization Using Jaya Algorithm". InComputational Intelligence in Data Mining—vol.2 pp:487-495, 2016.
[13] Rao RV, More KC, Taler J, Oclon P. Dimensional optimization of a micro-channel heat sink using Jaya algorithm". Applied Thermal Engineering.vol.103, pp:572-82, 2016.
[14] Phulambrikar S. "Solving Combined Economic Emission Dispatch Solution Using Jaya Optimization Algorithm Approach".2016
[15] H. Stuckenschmidt, R. Vdovjak, J. Broekstra, G. Houben, "Towards distributed processing of RDF path queries", Int. J. Web Eng. Technol.vol.2 no.2/3 pp.207–230, 2005.
[16] J. Broekstra, A. Kampman, F. Van Harmelen, "Sesame: A generic architecture for storing and querying rdf and rdf schema", in: International semantic web conference, Springer Berlin Heidelberg, pp. 54-68,2002.
[17] E.P. Shironoshita, M.T. Ryan, M.R. Kabuka, "Cardinality estimation for the optimization of queries on ontologies, SIGMOD Rec.vol.36,no.2, pp:13–18, 2007.
[18] A. Maduko, K. Anyanwu, A. Sheth, P. Schliekelman, "Estimating the cardinality of RDF graph patterns", in: Proceedings of the 16th International Conference on World Wide Web, ACM, Banff, AB, Canada, pp.1233–1234, 2007.
[19] M. Stocker, A. Seaborne, A. Bernstein, C. Kiefer, D. Reynolds, "SPARQL basic graph pattern optimization using selectivity estimation", in: Proceedings of the 17th International Conference on WWW, ACM, Beijing, China, pp:595–604, 2008.
[20] E. Ruckhaus, E. Ruiz, M. Vidal, "Query evaluation and optimization in the semantic web", Theory Pract. Log. Program. vol.8, no.3, pp:393–409, 2008.
[21] T. Neumann, G. Weikum, "RDF-3X: a RISC- style engine for RDF", Proc. VLDB Endow, Vol.1,No.1, pp.647–659, 2008.
[22] T. Neumann, G. Weikum, "Scalable join processing on very large RDF graphs", in: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD'09, ACM, New York, NY, USA, pp:627–640, 2009.
[23] Z. Kaoudi, K. Kyzirakos, M. Koubarakis, "SPARQL query optimization on top of DHTs", in: Proceedings of the 9th International Conference on the Semantic Web – vol. partI, ISWC'10, Springer-Verlag, Berlin, Heidelberg, pp:418–435, 2010.
[24] T. Neumann and G. Moerkotte, "Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins", in: ICDE, Hannover, Germany, pp:984-994, 2011.
[25] D. Ouyang, X. Wang, Y. Ye, and X. Cui, "A GA-based SPARQL BGP reordering optimization method", Advances in Information Sciences and Service Sciences, vol.4, no.9, pp:139–147, 2012.
[26] R. Gomathi, D. Sharmila, "A novel adaptive cuckoo search for optimal query plan generation", The Scientific World Journal, 2014.
[27] E. Guzel Kalayci, T.E. Kalaycı, D. Birant, "An ant colony optimization approach for optimising SPARQL queries by reordering triple patterns", Information Systems, vol.50 pp:51–68, 2015.
[28] J.J. Carroll, G. Klyne, "Resource description framework (RDF): Concepts and abstract syntax"– W3C recommendation, 2004.
[29] http://jena.apache.org.
[30] http://jena.apache.org/documentation/query.
[31] http://swat.cse.lehigh.edu/projects/lubm/
[32] com.hp.hpl.jena.graph.GraphStatisticsHandler
[33] S. N. Sivanandam, S.N. Deepa , "Introduction to Genetic Algorithm", ISBN 978-3-540-73189-4 Springer Berlin Heidelberg New York, Springer ñ Verlag Berlin Heidelberg 2008.
[34] O. Hartig, R. Heese, "The SPARQL query graph model for query optimization", in: Proceedings of the forthth European Conference on the Semantic Web: Research and Applications, ESWC'07, pp:564–578, 2007.
[35] Meimaris M, Papastefanatos G. Distance-Based Triple Reordering for SPARQL Query Optimization. in: Proceedings of the 33rd International Conference on Data Engineering (ICDE), IEEE, 2017, pp. 1559-1562.

Authors' Profiles

Sahil Saharan has done her MCA degree from NIT Kuruksherta and is pursuing Ph.D from the Department of Computer Applications, NIT, Kurukshetra. Her research interest is focused on Semantic Web, Query Optimization, Database and Data Analytics, Soft- Computing.

Dr. J.S. Lather has received B.E, M. Tech and Ph.D. from REC Kurukshetra. He has more than 23 year experience and presently working as Professor in Electrical Engineering Department, NIT Kurukshetra. His area of interest Wireless Communication, Robust Control of Time Delay Systems, Networked Control Systems, Consensus in WSN, Coop Control in Multi Agent Sys, Control of FACTs incorporating renewable energy. He has published more than 50 papers in various International National conferences and Journals.

Dr. R. Radhakrishnana has received B.E. and M.E. from NIT Trichy and Ph.D. from Jamia Milia Islamia in Handover Management in MIPv6. He has more than 17 years Industry and more than 10 years academic experience. His area of interest is Mobile and Wireless Communication. He has published more than 22 papers in various International National conferences and Journal.

How to cite this paper: Sahil Saharan, J.S. Lather, R. Radhakrishnan, " Specific Queries Optimization Using Jaya Approach", International Journal of Modern Education and Computer Science(IJMECS), Vol.10, No.3, pp. 38-46, 2018.DOI: 10.5815/ijmecs.2018.03.05

