A Hybrid BATCS Algorithm to Generate Optimal Query Plan

Gomathi Ramalingam[1] and Sharmila Dhandapani[2]
[1]Department of Computer Science and Engineering, Bannari Amman Institute of Technology, India
[2]Department of Electronics and Instrumentation Engineering, Bannari Amman Institute of Technology, India

Abstract: *The enormous increase in the amount of web pages day by day leads to progress in semantic web data management. The issues in semantic web data management are increasing and there is a need for improvement in research to handle them. One of the most important issues is the process of query optimization. The semantic web data stored in the form of Resource Description Framework (RDF) data can be queried using the popular query language SPARQL Protocol And RDF Query Language (SPARQL). As the size of the data increases, complication arises in querying the RDF data. The problem of querying the RDF graphs involves multiple join operations and optimizing those joins becomes NP-hard. Nature inspired algorithms are becoming much popular in recent days to handle problems with high complexity. In this research, a hybrid BAT Algorithm with Cuckoo Search (BATCS) is proposed to handle the problem of query optimization. The algorithm applies the echolocation behaviour of bats and hybrids with cuckoo search if the best solution stagnates for a designated number of iterations. Experiments were conducted with benchmark data sets and the algorithm proves that it performs efficiently in terms of query execution time.*

Keywords: *Data management, query optimization, nature inspired algorithms, bat algorithm, cuckoo search algorithm*.

Received November 7, 2014; accepted August 3, 2015

1. Introduction

Over the last decade, there is an enormous increase in the amount of data in the web. These data comes from a variety of fields including education, engineering, finance, weather reports and many more. Naturally the numbers of web users have evolved and gradually there is a change from data consumers to data producers. Managing the data is becoming very important and challenging. Although many new challenges exist in semantic web data management, querying the semantic web data in an efficient manner is becoming a very particular challenge in this context.

Querying the semantic web data first and foremost needs data to be stored in a data model. One of the most common frameworks to store the semantic web data is the Resource Description Framework (RDF).

The RDF stores the semantic web data in the form of triples which consists of subject, predicate and object. The most popular of them is the SPARQL Protocol and RDF Query Language (SPARQL).The major challenge in processing a query is choosing the optimal query plan for execution. A query can be executed in different manners to produce the same result. Query optimization is the process of choosing the best query plan among all possible plans. Although a lot of traditional optimization methods exist for query optimization, nature inspired algorithms are becoming quite common as an alternative to choose the best plan. With increase in size and complexity of the data, the application of nature inspired algorithms are best suited to find the optimal solutions. This research work is divided into the following sections: section 2 surveys the existing algorithms for query optimization; section 3 throws a light on the existing nature inspired optimization algorithms (Bat and Cuckoo Search); section 4 describes the implementation of the proposed algorithm; section 5 elaborates the datasets used in this research; section 6 discusses the experimental results obtained by applying the proposed algorithm; section 7 deals with the conclusion and the future work.

2. Related Work

A complex task in query processing is the query optimization. With the increase in complexity of the queries, searching for the best plan also becomes complex. Genetic Algorithm (GA) is becoming a best optimization method for handling very difficult optimization problems. The application of genetic algorithm [7] to the database query optimization was studied in literature. The robustness and efficiency of the algorithm is the main motivation for this application. Query plans are represented using query trees. Experimental results show that GA turns out to be the best alternative to the existing algorithms.

An efficient query processing algorithm based on swarm intelligence [11] was proposed in literature to reduce energy consumption in wireless networks. The

algorithm is based on the behaviour of ants and clustering and routing behaviour of networks. The results of the proposed algorithm show that optimal query agents can be produced efficiently. It also reduces cost and delay in delivery of events to appropriate query agents.

The increase in the amount of web pages leads to the development of new algorithms to process the querying mechanism of web data. To query large RDF graphs, an efficient algorithm called Adaptive Cuckoo Search (ACS) [2] algorithm was proposed in research. The algorithm is tested by varying the number of predicates and the results prove that the proposed algorithm works better than the existing algorithms.

One of the popular nature inspired algorithms is the Particle Swarm Optimization (PSO) algorithm. It has the capability to solve a large class of complex search problems. An algorithm using Bare Bones PSO [1] was modelled to handle the problem of distributed query optimization. The capabilities of PSO were evaluated against iterative programming and genetic algorithm.

The semantic web data represented by RDF needs fast query engines to process the data. Optimizing a special class of queries called RDF chain queries was focused in research. A genetic algorithm called RDF Chain Queries-Genetic Algorithm (RCQ-GA)[3] which determines the order in which the joins are to be processed was devised in literature. The proposed algorithm outperforms the benchmark quality.

The difficult and challenging issue in distributed database design is the query processing. The problem of query optimization was solved by using certain heuristics [13] and genetic algorithm. Computational experiments were conducted on the proposed algorithms and the experiments show that heuristics and genetic algorithms are feasible methods for solving query optimization problem in large scale distributed database systems.

A hybrid approach to answer SPARQL queries was proposed. The proposed approach makes use of both link traversal-based and distributed query processing-based approaches [4] in order to combine query answering over the Web of Linked Data and SPARQL endpoints respectively. Demonstrations are performed on a set of heuristics and optimization techniques for queries with time constraints.

The elementary concepts associated with efficient processing [6] of SPARQL queries was studied in literature. The study was performed on

1. The complexity analysis of all operators in SPARQL query language.
2. Equivalences of SPARQL algebra.
3. Algorithm for optimizing semantic SPARQL queries. The complexity analysis shows that all fragments of SPARQL fall into the category of NP.

A semantic technique on queries for retrieving more relevant results in cross language [12] information retrieval was presented in research. Experiments were evaluated in terms of precision and recall. The challenging issue in information retrieval is the way to express the queries. An interactive query expansion methodology [5] based on concept based directions finder was proposed. The proposed approach determines the directions in which to search the query.

3. Existing Nature Inspired Approaches

3.1. Cuckoo Search Algorithm

The optimization technique [9] based on the brood parasitism of cuckoo species by laying their eggs in the nests of other host birds is the Cuckoo Search (CS) algorithm. If a host bird find out the eggs which are not their own, it will either throw these unfamiliar eggs away or simply discard its nest and build a new nest elsewhere. This activity is used in the CS algorithm. A solution is represented by an egg in the nest and a cuckoo egg represents a new solution. The new solution (cuckoo), if better is replaced with the solution which is not so good in the nest. In most cases, each nest contains only one egg. A new solution was generated by Levy flight. The rules for CS are depicted as follows:

- Only one egg is laid by each cuckoo lays at a time, and it is dumped into a randomly chosen nest.
- The best nests with worthy eggs will be carried over to the next generation.
- The number of available host nests is fixed, and a host can discover a foreign egg with a probability $p_a \in [0, 1]$.

In this case, the host bird can either throw the egg away or discard the nest so as to build a completely new nest in a new location.
The algorithm for CS is given in Algorithm1:

Algorithm 1: Pseudo code for CS

Generate an initial population of n host nests;
while (t<MaxGeneration) or (stop criterion)
Get a cuckoo randomly (say, i) and replace its solution by performing Levy flights;
Evaluate its fitness F_i
Choose a nest among n (say, j) randomly;
if ($F_i < F_j$)
Replace j by the new solution;
end if
A fraction (p_a) *of the worse nests is abandoned and new ones are built;*
Keep the best solutions/nests;
Rank the solutions/nests and find the current best;
Pass the current best to the next generation;
end while

While generating new solution *x(t+1)* for a cuckoo *i*, a Levy flight is performed using the following Equation (1).

$$x_i(t+1) = x_i(t) + \alpha \oplus \text{Levy}(\lambda) \quad (1)$$

The symbol $\oplus$ is an entry-wise multiplication. Levy flights provide a random walk while their random steps are drawn from a Levy distribution for large steps as given in Equation (2)

$$\text{Levy} \sim u = t^{-\lambda} \tag{2}$$

which has an infinite variance with an infinite mean. Here the consecutive jumps of a cuckoo essentially form a random walk process which obeys a power-law step-length distribution with a heavy tail.

3.2. Bat Algorithm

Microbats are insectivores. Bats use echolocation to locate and catch their prey [10]. Bat echolocation is a perceptual system where ultrasonic sounds are emitted specifically to produce echoes. When the outgoing pulse is compared with the returning echoes, the bat produces detailed images of the environment. From this bats can perceive, limit and even categorize their prey in complete darkness. When bats fly, they produce a steady stream of high-pitched sounds that can be heard only by them. When the sound waves produced by these bats hit an insect or other animal, the echoes bounce back to the bats, and guide them to the source .Their pulses vary in properties and can be correlated with their hunting strategies, depending on the species. The noise also varies from the loudest when searching for prey and to a quieter base when homing towards the prey.

The rules for Bat algorithm are:

1. Bats use echolocation to sense distance, and they also know the difference between food/prey and background barriers.
2. Microbats fly randomly with velocity v_i at position x_i with a fixed frequency f_{min}, varying wavelength λ and loudness A_0 to search for prey. They can automatically fine-tune the wavelength (or frequency) of their emitted pulses and alter the rate of pulse emission $r \in [0, 1]$, depending on the proximity of their target.
3. Even though the loudness can vary in many ways, it is assumed that the loudness varies from a large (positive) value A_0 to a minimum constant value A_{min}.

The pulse frequency, velocity and position of the bat are given by

$$f_i = f_{min} + (f_{max} - f_{min}) \times \beta \tag{3}$$

$$v_i(t) = v_i(t-1) + (X_i(t) - X^*) \times f_i \tag{4}$$

$$X_i(t) = X_i(t-1) + v_i(t) \tag{5}$$

Where $\beta \in [0,1]$ is a random number drawn from a uniform distribution. X^* is the current global best location among n bat solutions.

For the local search, once a solution is selected among the current best solutions, a new solution for each bat is generated locally using random walk.

$$X_{new} = X_{old} + \varepsilon A^t \tag{6}$$

Where ε is a random number $[-1, 1]$ and A^t is the average loudness of all bats at time step t.

The loudness A_i and the rate r_i of pulse emission have to be updated accordingly as the iterations proceed.

$$A_i(t+1) = \alpha A_i(t) \tag{7}$$

$$r_i(t+1) = r_i(0)[1-\exp(-\gamma t)] \tag{8}$$

Where α and γ are constants, given by $0 < \alpha < 1$ and $\gamma > 0$

The bat algorithm [10] is explained in algorithm 2:

Algorithm 2: Pseudo code for Bat Algorithm

Initialize the bat population and velocity
Define pulse frequency f_i at x_i
Initialize pulse rates r_i and the loudness A_i while (t <max number of iterations)
Generate new solutions by adjusting frequency, and updating velocities and locations (using Equations 3, 4 & 5)
if (rand > r_i)
Select a solution among the best solutions
Generate a local solution around the selected best solution (using Equation 6)
end if
if (rand < A_i & $f(x_i) < f(x^*)$)
Accept new solutions
Increase r_i and reduce A_i
end if
*Rank the bats and find the current best x**
end for
end while

4. The Proposed Hybrid BATCS Algorithm

4.1. Representation of Bats

The input SPARQL query can be executed in a number of diverse ways to produce the same result. Each query can be represented as a query tree with triples at the leaf nodes and the intermediate nodes used to join the triples. Different forms of query trees are available like bushy trees, left deep trees, right deep trees and so on. A left deep tree representation is used in this research. Each query plan (bat) is represented as a left deep query tree as in Figure 1.

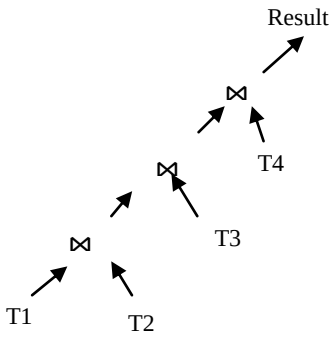

Figure 1. A sample left deep tree.

In the left deep tree T1, T2, T3, and T4 represent the triples and the intermediate nodes join the triples. For example consider the sample query,

- Q4 of LUBM dataset

```
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, ?Y1, ?Y2, ?Y3
WHERE
{?X rdf:type ub:Professor .
 ?X ub:worksFor
<http://www.Department0.University0.edu> .
 ?X ub:name ?Y1 .
 ?X ub:emailAddress ?Y2 .
 ?X ub:telephone ?Y3}
```

The query can be represented as a left deep tree as in Figure 2.

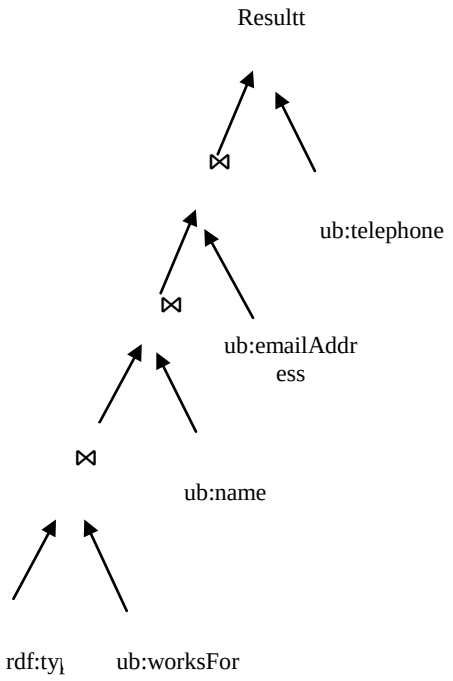

Figure 2. Left deep tree for sample query Q4.

4.2. Solution Space

The solution space of the proposed algorithm consists of a set of bats. The number of possible left deep trees depends upon the number of predicates in the query and the type of query tree used. Since a left deep tree is used in this research, there is a possibility of n! [8] different query plans for a tree with n predicates. The n! bats can be obtained by applying the transformation rules like join commutativity, join associativity, left join exchange and right join exchange.

4.3. Encoding of Bats

To apply any optimization algorithm to solve a problem, a suitable encoding format must be chosen for the bats in the solution space. Two types of encoding are available for left deep trees [8],

1. Ordered list.
2. Ordinal number encoding.

In this research, ordered list is chosen for encoding bats (query plans). Solutions are represented as an ordered list of leaves. For example, the query plan tree in Figure 1, (((T1∞T2) ∞T3) ∞T4) can be encoded as "1234".

The sample query query4 given in the previous section consists of five predicates. So there are 5! =120 different ways in which we can represent the query tree which gives the same result.

In Figure 2, consider
rdf:type as 1
ub:worksFor as 2
ub:name as 3
ub:emailAddress as 4
ub:telephone as 5
Then the possible encoding will be as follows:
12345
23451
34512
45123
51234
and so on up to 120 solutions are possible.

4.4. Fitness Function

To solve the problem of query optimization, let us choose the fitness function. The fitness function in the context of query optimization refers to the cost of the left deep tree. The cost of a left deep tree relies on the selectivity and cardinality estimation. Cardinality of a triple pattern is the number of triples that match a particular pattern. Selectivity of a join between two triples T1 and T2 is defined as the number of triples satisfying both T1 and T2. Let R_i be the cardinality and $f_{i,j}$ be the selectivity. If $p_{i,j}$ is the join predicate between R_i and R_j,we can define.

$$f_{i,j} = \left| \frac{R_i \bowtie_{pi,j} R_j}{R_i \times R_j} \right| \quad (9)$$

For a given join tree *T*, the resultant cardinality |T| can be recursively computed as

$$|T| = |R_i| \text{ if T is a leaf } R_i \quad (10)$$

$$|T| = (\prod_{R_i \in T1, R_j \in T2} f_{i,j}) |T1| |T2| \text{ if } T = T1 \bowtie T2. \quad (11)$$

For a given join tree *T*, the cost function C_{out} is defined as

$$C_{out}(T) = 0 \text{ if T is a leaf } R_i \quad (12)$$

$$C_{out}(T) = |T| + C_{out}(T1) + C_{out}(T2), \text{ if } T = T1 \bowtie T2 \quad (13)$$

4.5. Implementation of the Proposed Hybrid BATCS Algorithm

The proposed work uses a hybrid of Bat algorithm with CS called hybrid BATCS algorithm. In this proposed algorithm, initially Bat algorithm is applied to optimize the query and if bat algorithm stagnates for a designated number of iterations, then the CS algorithm is applied to find the optimal query plan. The all possible query plans are represented as a population of bats in the solution space.

The proposed hybrid BATCS algorithm is given by algorithm 3:

Algorithm 3: Pseudo code for Hybrid Bat Algorithm with Cuckoo Search

Initialize the bat population and velocity
Define pulse frequency f_i at x_i
Initialize pulse rates r_i and the loudness A_i
while (t <max number of iterations)
Generate new solutions by adjusting frequency, and updating velocities and locations (using equations 3, 4 & 5)
if (rand > r_i)
Select a solution among the best solutions
Generate a local solution around the selected best solution (using equation 6)
end if
if (rand < A_i & $f(x_i) < f(x^*)$)
Accept new solutions
Increase r_i and reduce A_i
end if
*Rank the bats and find the current best x**
end for
if best solution stagnates for designated number of iterations
Apply Cuckoo search algorithm
end if
end while

Table 1 lists the parameters set for the proposed algorithm.

Table 1. Parameters and their values for CS and bat algorithm.

Parameter	Value
Cuckoo Search algorithm	
No.of iterations	100
p_a	0.3
α	1
λ	1.5
Bat algorithm	
No.of iterations	100
α	rand(0,1)
β	rand(0,1)
γ	0.5
A	0.25
r	0.5

5. Datasets

The dataset used to test the proposed algorithm is the Lehigh University Benchmark (LUBM) dataset which is the most popular benchmark for semantic web repositories. Using the data generator available with LUBM three datasets LUBM (1, 0), LUBM (3, 0) and LUBM (5, 0) of different sizes were generated. The benchmark consists of 14 test queries. LUBM (1,0) provides an ontology describing the structure of a single university. LUBM (3,0) and LUBM (5,0) describes an ontology with three and five universities respectively.

6. Experimental Results

The proposed algorithm is experimented in a Microsoft Windows 8 platform on a Intel Pentium 4 machine with 2GB RAM. Each of the three datasets consists of more than 1, 00,000 triples. The number of predicates varies according to the type of the query. The algorithm is iterated for 100 times and the fitness values obtained are recorded. Figure 3 and 4 shows the fitness values obtained for a sample set of queries using the LUBM (1, 0) dataset.

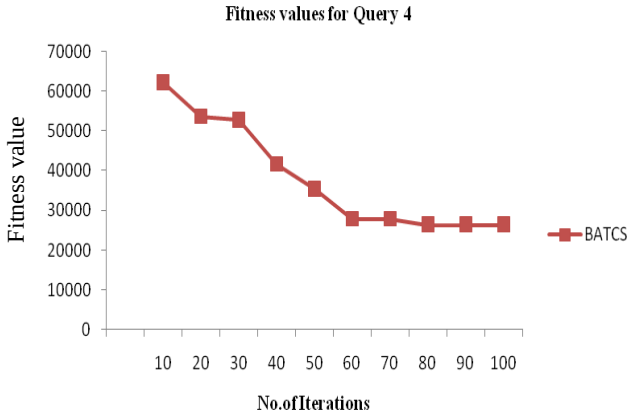

Figure 3. Fitness value for Query 4.

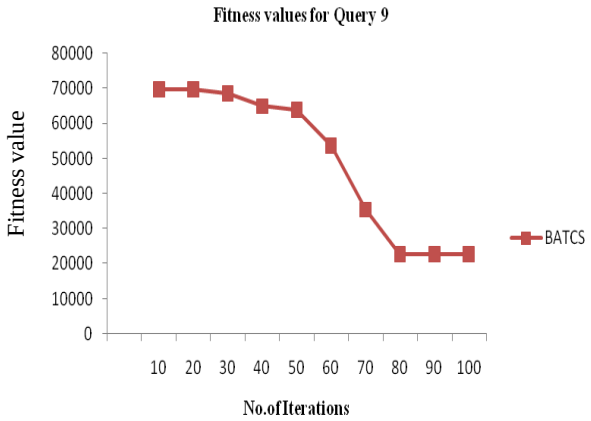

Figure 4. Fitness value for Query 9.

The Figures 5 and 6 shows the fitness values obtained for a sample set of queries using the LUBM (3, 0) dataset.

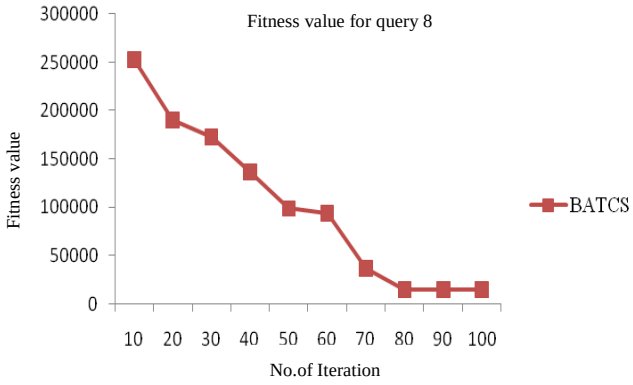

Figure 5. Fitness value for query 8.

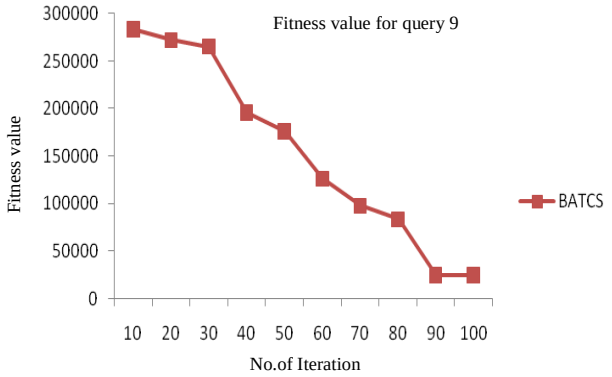

Figure 6. Fitness value for query 9.

The Figures 7 and 8 shows the fitness values obtained for a sample set of queries using the LUBM (5, 0) dataset.

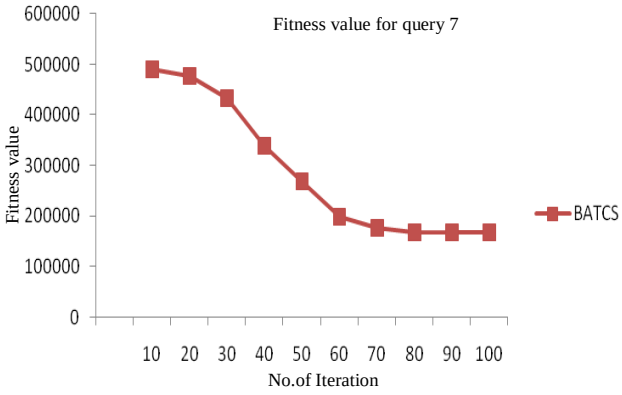

Figure 7. Fitness value for Query 7.

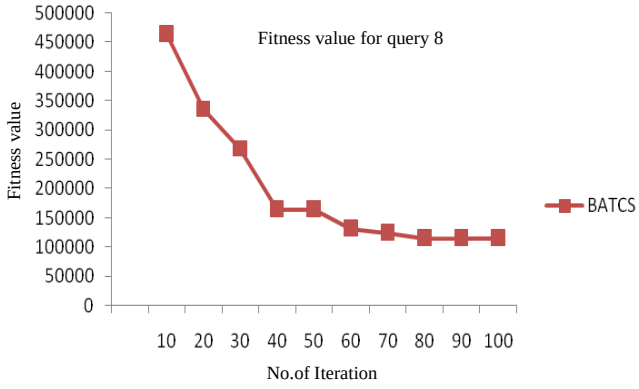

Figure 8. Fitness value for Query 8.

The average execution times obtained for three datasets for varying number of predicates is recorded. The proposed hybrid BATCS algorithm is compared with GA and PSO. The Figure 9, 10 and 11 shows the execution times of different queries for the three datasets compared with GA and PSO algorithms.

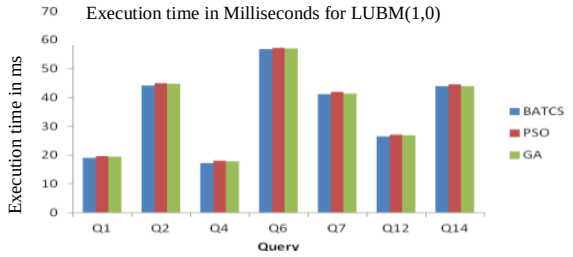

Figure 9. Execution time in milliseconds for LUBM(1,0) dataset.

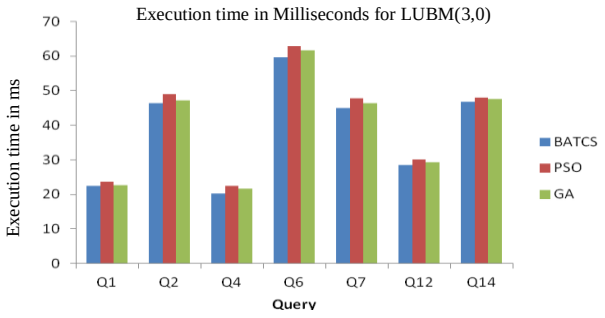

Figure 10. Execution time in milliseconds for LUBM(3,0) dataset.

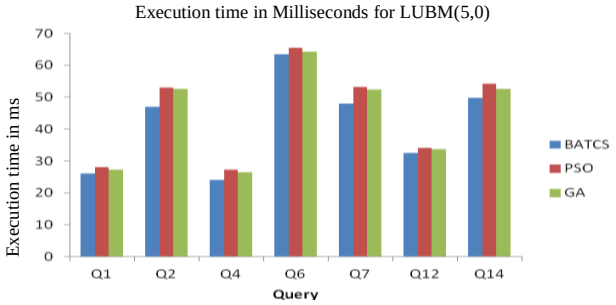

Figure 11. Execution time in milliseconds for LUBM(5,0) dataset.

7. Conclusions and Future Work

In this research, a hybrid algorithm called BATCS algorithm was presented to solve the problem of query optimization. The algorithm begins with a solution space consisting of all possible query plans. The query plans represents the bats and eggs of the bat and Cuckoo Search algorithms. The cost of the query plan is chosen as the fitness function which is calculated based on the cardinality and selectivity of the triples occurring in the dataset.

The experimental results show the efficiency of the algorithm in terms of query execution time. The BATCS algorithm has been applied to three datasets of varying sizes and the best query plan is found based on the fitness function and the execution time is recorded. The BATCS algorithm outperforms when compared to GA and PSO. To improve the correctness of the work, other hybrid nature inspired algorithms

can be applied and performance can be measured in the future.

References

[1] Dokeroglu T., Tosun U., and Cosar A., "Particle Swarm Intelligence as a New Heuristic for the Optimization of Distributed Database Queries," *in Proceedings of International Conference on Application of Information and Communication Technologies*, Tbilisi, pp.1-7, 2012.

[2] Gomathi R. and Sharmila D., "A Novel Adaptive Cuckoo Search for Optimal Query Plan Generation," *The Scientific World Journal*, vol. 2014, pp.1-7, 2014.

[3] Hogenboom A., Milea V., Frasincar F., and Kaymak U., "RCQ-GA: RDF Chain Query Optimization Using Genetic Algorithms," *in Proceedings of International Conference on Electronic Commerce and Web Technologies*, Linz, pp. 181-192, 2009.

[4] Lynden S., Kojima I., Matono A., Nakamura A., and Yui M., "A Hybrid Approach to Linked Data Query Processing with Time Constraints," *in Proceeding of LDOW 996*, Rio de Janeiro, 2013.

[5] Meiyappan Y. and Iyengar S., "Interactive Query Expansion using Concept-Based Directions Finder Based on Wikipedia," *The International Arab Journal of Information Technology*, vol. 10, no. 6, pp. 571-578, 2013.

[6] Schmidt M., Meier M., and Lausen G., "Foundations of SPARQL Query Optimization," *in Proceedings of the 13th International Conference on Database Theory*, Lausanne, pp. 4-33, 2010.

[7] Sinha M. and Chande S., "Query Optimization Using Genetic Algorithms," *Research Journal of Information Technology*, vol. 2, no. 3, pp. 139-144, 2010.

[8] Steinbrun M., Moerkotte G., and Kemper A., "Heuristic and Randomized Optimization for the Join Ordering Problem," *VLDB Journal*, vol. 6, no. 3, pp. 191-208, 1997.

[9] Yang X. and Deb S., "Cuckoo Search Via Levy Flights," *in Proceedings of the World Congress on Nature and Biologically Inspired Computing*, Coimbatore, pp. 210-214, 2009.

[10] Yang X. and He X., "Bat Algorithm: Literature Review and Applications," *International Journal of Bio-Inspired Computation*, vol. 5, no. 3, pp. 141-149, 2013.

[11] Yu J., Zhang L., Chen M., and Liu X., "Hybrid Ant Algorithm Based Query Processing with Multiagents in Sensor Networks," *International Journal of Distributed Sensor Networks*, vol. 9, no. 9, pp. 1-7, 2013.

[12] Yunus M., Zainuddin R., and Abdullah N., "Semantic Method for Query Translation," *The International Arab Journal of Information Technology*, vol. 10, no. 3, pp. 253-259, 2013.

[13] Zhou Z., "Using Heuristics and Genetic Algorithms for Large-scale Database Query Optimization," *Journal of Information and Computing Science*, vol. 2, no. 4, pp. 261-280, 2007.

Gomathi Ramalingam completed her under graduation in the year 2003 and post graduation in the year 2011. She is pursuing her Doctorate in Anna University, Chennai. At present she is working as an Assistant Professor (Sr.Grade) in the Department of Computer Science and Engineering at Bannari Amman Institute of Technology, Sathyamangalam, Erode Dt. She has over 12 years of teaching experience. She has published her papers in 4 International conferences, 6 National Conferences and 8 International Journals.

Sharmila Dhandapani completed her under graduation in the year 1996 and post graduation in the year 2004. She has been awarded Doctorate in the year 2010 from Anna University, Chennai. At present she is working as Professor and Head of Electronics and Instrumentation Engineering in Bannari Amman Institute of Technology, Sathyamangalam. She has over 18 years of teaching experience. She has published her papers in 2 National and 35 International Journals. She has also presented her papers in 10 National and 19 International Conferences.