ISSN: 0369-8963

# An Effective Hybrid Algorithm based on Modified Cuckoo Search and Firefly for Resource Description Framework based SPARQL Query language Optimization

**S.DHANALAKSHMI[1], N. MOHANASUNDARARAJU[2], T.JESUDAS[2] AND D.NOORULLAH[1]**

*[1]Government College of Engineering, Salem, Tamil Nadu - 636 101, India*
*[2]Mahendra Engineering College, Namakkal, Tamil Nadu - 637 503, India*

*[*]Corresponding Author: S.Dhanlakshmi*

## ABSTRACT

*Semantic web technology is the vision of the World Wide Web Consortium which enables people to create and maintain data stores in the internet. It refers to a web of linked data. Various technologies are available to retrieve information from the Semantic web using query languages. SPARQL (SPARQL Protocol and RDF Query Language) query language enables users to query the RDF data available from the Semantic web. Since the number of web pages increases from day to day in the internet, enormous volumes of data available from the Semantic Web must be queried efficiently with less amount of time. In this research work a Modified Hybrid of Cuckoo Search and Firefly (MHCSF) algorithm is proposed to handle the problem of SPARQL query optimization. It uses a modified hybrid of Cuckoo Search (CS) and Firefly algorithm (FA) in parallel to effectively query the Semantic web data store with less amount of time. The proposed algorithm is compared with existing algorithms CS and FA and the query execution times are documented. The experimental analysis shows that the proposed algorithm takes less execution time compared to the other algorithms.*

Keywords: *Semantic web; SPARQL queries; Query optimization; Cuckoo Search algorithm; Firefly algorithm; RDF triples.*

## INTRODUCTION

Semantic web is the concept initiated by the World Wide Web Consortium (W3C) in order to make the machines to understand the meaning of web pages [1]. It is an extension of the original web which tries to describe and link the web pages in a manner that can be understandable by the machines. The motivation behind semantic web includes automation, personalization, information retrieval, and data reuse and knowledge discovery. In the area of academics and industry, the most popular representation of Semantic web data in a variety of applications is the Resource description Framework (RDF) data [2]. It provides a common method to describe information about the semantic web so that it can be easily understandable by humans and computers. This model acts as a framework for denoting the Semantic web data. RDF is a data model designed to characterize information with less constraints and flexibility [3]. It is used in applications, in which it can be easily understood. The worth of information thus gets increased and it is accessible to many applications across the entire web. The primary query language used for querying RDF represented data is the SPARQL [4]. SPARQL is a recursive acronym for SPARQL Protocol and RDF Query Language. It uses Basic Graph Patterns (BGP) called a set of triples. The queries in SPARQL are similar to SQL (Structured Query Language) queries. The research work in this paper is organized as below. The latest research in SPARQL query optimization is analyzed and presented in Section 2. The proposed hybrid of Cuckoo Search and Firefly algorithm for SPARQL query optimization is discussed in Section 3. Section 4 presents the problem statement, encoding mechanism of query plans and fitness functions used in the

 https://doi.org/10.37896/pd91.5/91515

ISSN: 0369-8963

proposed algorithm. In the experimental Section, the investigational results for the optimizing a sample set of SPARQL queries with benchmark RDF datasets are discussed. A comparison of the proposed algorithm with the existing algorithms is also analyzed. Section 6 presents the conclusions and future work.

## BACKGROUND WORK

A new hybrid of Cuckoo search (CS) and firefly algorithm (FA) [5] optimization algorithm was focused in research. The hybrid approach combines the concepts from CS and FA and creates new solution in the new generation by combining random walk concept and the concepts of firefly algorithm. Evaluations were performed by comparing the classical algorithms and the hybrid algorithms to measure the performance. A parameter search based cuckoo search algorithm was proposed [6] in literature to solve the problem of searching parameters of the cascade controllers. The problem is complex since the search space for this problem is very broad. A heuristic algorithm based on a hybrid of Cuckoo and Tabu search was proposed in research to solve the multi join query ordering problem [7]. Exciting simulations results were obtained by applying the hybrid algorithm. Usually, the Cuckoo search algorithm uses the levy flight as the mutation operator to explore the search space. In research seven different mutations methods are replaced instead of levy flight with the Cuckoo search algorithm [8] and tested for the performance. Alami et al. proposed a multimodal optimization algorithm using fuzzy clustering on cultural algorithms are more dependable for solving multimodal optimization problems. The complete population set will be divided into equal number of parts of subpopulations using these fuzzy based clustering concepts. Also simultaneously, the cultural algorithm will be allowed for searching in each subpopulation. For solving multimodal optimization problems, the cultural exchange concepts have been introduced [9]. Zhang et al. developed a firefly model based algorithm of Return-Cost-based Binary Firefly Algorithm (Rc-BBFA) to effectively reduce the risk of premature convergence. The algorithm mainly concentrates on three phases. In the first phase, using the return-cost indicator the firefly's attractiveness over other fireflies was measured. In the second phase Pareto dominance based strategy was used to search the attractive fireflies on every other firefly. Finally the adaptive jump and return-cost attractiveness based binary movement operator was implemented to update the location of the firefly [10]. Selvakumar and Muneeswaran proposed a filter and wrapper based firefly model of MIFF for feature selection to detect intrusion in the network using KDDCUP 99 datasets. The MI based firefly is implemented to use voting based feature selection. They achieved considerable accuracy with minimal number of features [11]. Sainte and Alalyani 2018 proposed a novel firefly model for feature selection using OSAC real datasets to enhance the performance and reduce the execution time. Using Arabian documents the performance of the developed model is significant [12]. Fister et al., proposed a metaheuristic optimization algorithm on the flashing behavior of the fireflies to attract other fireflies. The brightness of the firefly decides the attractiveness of a firefly and less brightness firefly will be attracted by brighter firefly until the best one is identified in the population [13].

The most widely used nature inspired concepts includes the Firefly and cuckoo search algorithms. They are famous because of its simplicity and less computational cost. A new hybrid algorithm is suggested in literature [14] by combining the firefly algorithm and the cuckoo search algorithm.

This hybrid algorithm can solve constrained optimization problems (COPs) and engineering optimization problems in real time. The proposed algorithm Hybrid Firefly Algorithm and Cuckoo Search (HFFACS) algorithm brings a balance between the exploration and the

 https://doi.org/10.37896/pd91.5/91515

ISSN: 0369-8963

exploitation process. The proposed algorithm is applied to benchmark constrained optimization problems and engineering optimization problems and comparisons were performed.

## A HYBRID ALGORITHM – MODIFIED CSFA (Cuckoo search – Firefly Algorithm)

### Defining the Solution Space and Solutions

In any optimization algorithm, there is a solution space which consists of a set of all solutions that satisfies the problems constraints. With respect to the problem of SPARQL query optimization, the solution space is a place which contains the set of all possible query plans represented as a query tree. Any SPARQL query can be represented using different query trees which produce the same result. The query tree consists of nodes connected by edges. Different varieties of representations are possible with query trees. In this proposed algorithm, query plans are represented using the right deep trees [15-18]. Each query plan is a right deep tree and there is 'n!' possibilities of solutions placed in the solution space. All the possible query plans constitute the solution space. The possible set of solutions can be framed by applying transformation rules available in literature. The general schematic on calculation on fitness value is shown in the Figure 1.

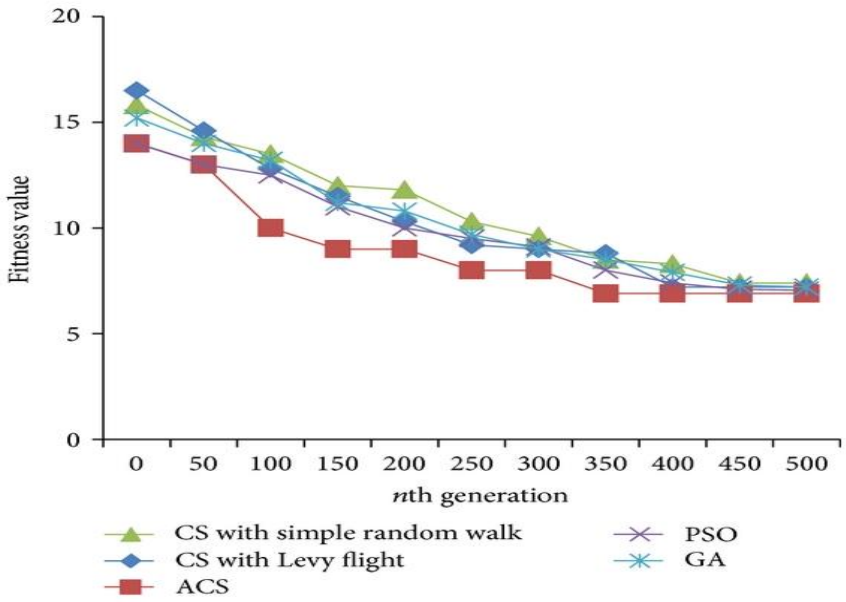


Fig. 1: The general calculation on fitness value

### The general Algorithm on Query Language

| **Input**<br>A collection of RDF files containing subject and predict<br>Query & Generations | **Output**<br>Optimal plan |
|---|---|
| ***Step 1*** **-** Random initial N generation<br>***Step 2*** **-** Evaluation<br>***Step 3*** **-** Sort the query plans based on fitness value<br>**Step4 -** Select the random generation | |


https://doi.org/10.37896/pd91.5/91515

ISSN: 0369-8963

| |
|---|
| ***Step 5 -*** Compare the fitness generation |
| ***Step 6 -*** Keep the best query |
| ***Step 7 -*** Rank them |
| ***Step 8 -*** Pass the best possible query to the next phase |

**Solution for Encoding**

Every solution in the solution space will be in some format that cannot be understood by the optimization algorithm. So, it must be encoded to a format that can be processed by the optimization algorithm. Since right deep trees are chosen to represent the solutions, ordered list encoding is used to encode the query trees. For example, consider the right deep query tree with nodes and joins in the order (R1⋈ (R2 ⋈ (R3 ⋈ (R4 ⋈R5)))). It can be encoded as "12345".

**Deciding the Objective Function**

For any optimization problem, Objective function is one which is used to maximize profits or minimize the number of losses based on some parameters. So it is important in deciding an objective function for the context of query optimization [16-17]. The accepted objective function for query optimization problem is the cost of the query plan. Since a single query can be solved in many different ways, the cost of the query tree decides the final query plan and so it is chosen as an objective/fitness function.

**Solution for encoding**

Every solution in the solution space will be in some format that cannot be understood by the optimization algorithm. So, it must be encoded to a format that can be processed by the optimization algorithm. Since right deep trees are chosen to represent the solutions, ordered list encoding is used to encode the query trees. For example, consider the right deep query tree with nodes and joins in the order (R1⋈ (R2 ⋈ (R3 ⋈ (R4 ⋈R5)))). It can be encoded as "12345".

**Deciding the objective Function**

For any optimization problem, Objective function is one which is used to maximize profits or minimize the number of losses based on some parameters. So it is important in deciding an objective function for the context of query optimization. The accepted objective function for query optimization problem is the cost of the query plan. Since a single query can be solved in many different ways, the cost of the query tree decides the final query plan and so it is chosen as an objective/fitness function.

The formula to calculate the cost of a query plan is as below:

Cost of a query tree= Summation of the cost of each branch of the query tree

In the branches of the query tree there may be some operators whose cost is calculated depending upon the cardinality and selectivity estimation. The objective of this analysis is to minimize the query execution time in the process of retrieving bulk volume of data from the Semantic web repositories. With the above basic decisions, the optimization algorithm is applied to the query. The coding part is implemented with Java programming language with Jena API platform and the query execution time obtained results are noted.

| ***Algorithm (Step-by-step)*** |
|---|
| ***Step: 1*** Sort the given solutions in the solution space. |
| ***Step: 2*** Divide the sorted solutions in the solution space into two groups. |
| ***Step: 3*** Initialize the population of group1 and group2. |
| ***Step: 4*** Evaluate the fitness of solutions in each group. |
| ***Step: 5 R***epeat |

 https://doi.org/10.37896/pd91.5/91515

ISSN: 0369-8963

*Execute the following two steps in parallel.*

*Apply Cuckoo search algorithm to group1.*

*Apply firefly algorithm to group2.*

***Step: 6***Find the best solution from the two group after applying optimization algorithms.

***Step: 7*** Combine the two groups and regroup it into two groups randomly.

***Step: 8*** Evaluate the fitness of each solution.

Until a terminate condition is met.

***Step: 9*** Find the query plan with best query cost and execution time.

## RESULTS AND DISCUSSION

This segment calculates the performance of the proposed algorithm with benchmark RDF datasets and also compares the proposed algorithm with the other existing algorithms

### Datasets

LUBM (Lehigh University Benchmark) dataset is a benchmark dataset which includes University domain ontology. The RDF data about any number of universities can be generated randomly and the dataset generated is also scalable. The term LUBM (N, S) identifies the number of university data in which N stands for the number of universities and S mentions the seed value for the dataset generator. For example, if the N is 1 and S is 0, the LUBM dataset generator generates data for one university named as University0. If the N is 2 and seed is 0, the dataset generator generates data for two universities named as University0 and University1. The benchmark also provides us with 14 test queries named starting from Q1 to Q14.

### Performance Comparison

The performance of the proposed algorithm is examined using the LUBM (3, 0) dataset which contains 3, 25,400 triples, 54 classes and 30 properties. The optimization algorithm is implemented in a Microsoft Windows environment on an i5 machine containing a 4GB RAM configuration. Java programming language is used along with Jena API to code the algorithm. To find out the effectiveness of the algorithm, several tests are conducted with queries with different number of predicates. The execution of the optimization algorithm is iterated greater than 100 times in order to discover the effectiveness of the produced results. The proposed work is also compared with existing Firefly and CS algorithms and the results are documented. The benchmark LUBM dataset consists of 14 test queries. These queries are named from Q1 to Q14. The queries Q1, Q3, Q5, Q10 and Q11 includes 2 predicates, Q2 and Q9 includes 6 predicates, Q4 and Q8 includes 5 predicates, Q6, Q13 and Q14 includes 1 predicate, Q7 and Q12 includes 4 predicates.

The proposed algorithm is experimented with sample test queries Q2, Q8 and Q12 and taken into consideration for fitness evaluation.

### Performance of MHCSF Algorithm with LUBM (3, 0) Dataset

The fitness values obtained from LUBM test query Q2 are noted. Q2 retrieves the graduate students studying in the university. This query involves a triangular relationship between the objects. The fitness values obtained in various iterations of the proposed system are shown in Table 1. These fitness values are also compared with the fitness obtained when executing with the already existing algorithms Cuckoo Search (CS) and Firefly algorithm (FA).

 https://doi.org/10.37896/pd91.5/91515

ISSN: 0369-8963

Table 1: Fitness Values for Q2 with LUBM (3, 0)

| Iterations | Fitness Value | | |
|---|---|---|---|
| | FA | CS | MHCSF |
| 10 | 230500 | 230250 | 220560 |
| 30 | 230000 | 229800 | 219450 |
| 50 | 228500 | 228500 | 208320 |
| 66 | 227485 | 227885 | 205485 |
| 86 | 225530 | 215730 | 193254 |
| 95 | 224745 | 204641 | 185699 |
| 118 | 215890 | 195867 | 185699 |
| 127 | 215890 | 195687 | 185699 |
| 138 | 215890 | 195687 | 185699 |
| 150 | 215890 | 195687 | 185699 |

The query Q2 contains 6 predicates and so there is a possibility of '6!' Solutions would produce the same results. The convergence of the MHCSF algorithm starts from iteration 95 with 185699 number of triples. It remains constant till the 150$^{th}$ iteration and the plan corresponding to this cost becomes the optimal plan. The CS and FA algorithms produces plans with 195687 number of triples and 215890 number of triples whose execution times are greater than that produced by MHCSF algorithm. The below figure 2 shows the fitness chart for the test query Q2.

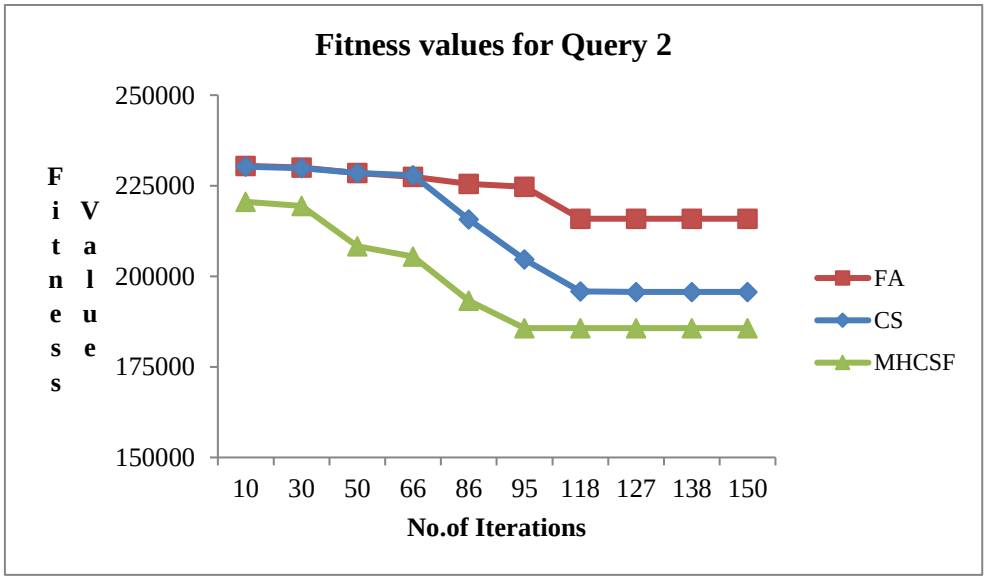


Fig. 2: Fitness Chart for Q2 with LUBM(3,0)

It is clear from the chart that the MHCSF algorithm generates optimal query plan comapred to CS and FA algorithms.The fitness values retrieved from LUBM test query Q8 are documented. Q8 consists of 5 predicates and thus the solution space consists of 5!=120 possible query plans as solutions. The fitness values recorded for different iterations are shown in Table 2.


https://doi.org/10.37896/pd91.5/91515

ISSN: 0369-8963

**Table 2:** Fitness Values for Q8 with LUBM (3, 0)

| Iterations | Fitness Value | | |
|---|---|---|---|
| | FA | CS | MHCSF |
| 10 | 250505 | 240256 | 230578 |
| 30 | 220000 | 215622 | 221689 |
| 50 | 218545 | 203456 | 218756 |
| 70 | 217488 | 205897 | 207894 |
| 88 | 195862 | 198745 | 171545 |
| 94 | 182258 | 178955 | 171545 |
| 120 | 172396 | 178955 | 171545 |
| 129 | 172396 | 178955 | 171545 |
| 133 | 172396 | 178955 | 171545 |
| 150 | 172396 | 178955 | 171545 |

The query Q8 is a complex query and so it produces query plans with more cost during the initial iterations with the MHCSF algorithm. But the convergence starts in the earlier iteration 88 with a number of triples as 171545. Comparing to the existing algorithms, the coat generated by the proposed system is lesser. Fig. 3 shows the fitness chart obtained for the benchmark test query Q8 compared with the existing algorithms FA and CS. The fitness values obtained for executing query Q12 with proposed algorithm is given in Table 3.

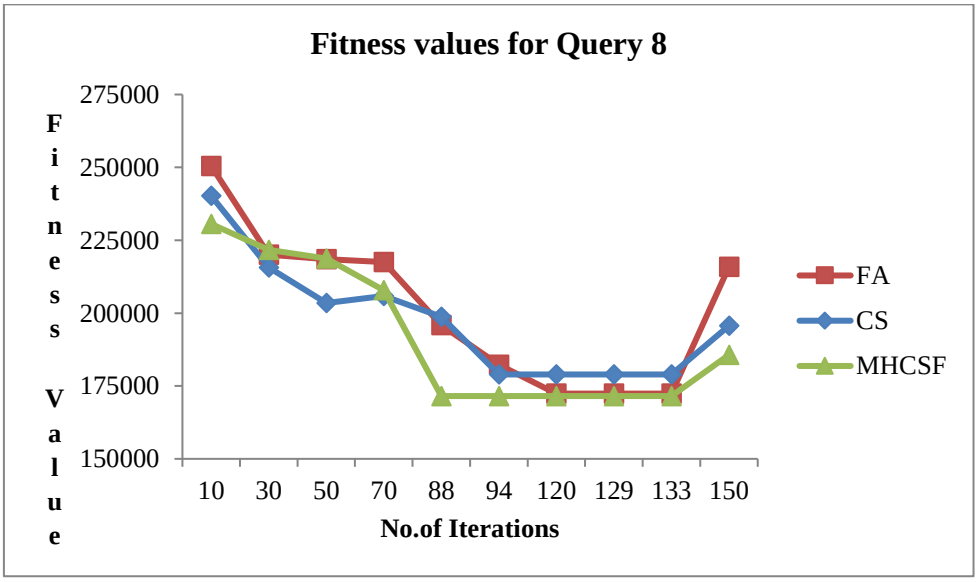


Fig. 3: Fitness Chart for Q8 with LUBM(3,0)

Table 3: Fitness Values for Q12 with LUBM (3, 0)

| Iterations | Fitness Value | | |
|---|---|---|---|
| | FA | CS | MHCSF |
| 10 | 290563 | 285555 | 302589 |
| 30 | 285975 | 278945 | 297859 |
| 45 | 275896 | 263456 | 278945 |


https://doi.org/10.37896/pd91.5/91515

ISSN: 0369-8963

| | | | |
|---|---|---|---|
| 70 | 272589 | 258977 | 267894 |
| 97 | 258956 | 235569 | 198576 |
| 120 | 248555 | 236845 | 198576 |
| 135 | 248555 | 236845 | 198576 |
| 139 | 248555 | 236845 | 198576 |
| 146 | 248555 | 236845 | 198576 |
| 150 | 248555 | 236845 | 198576 |

Fig. 4 shows the fitness chart obtained for the benchmark test query Q12 by proposed algorithm compared with existing algorithms.

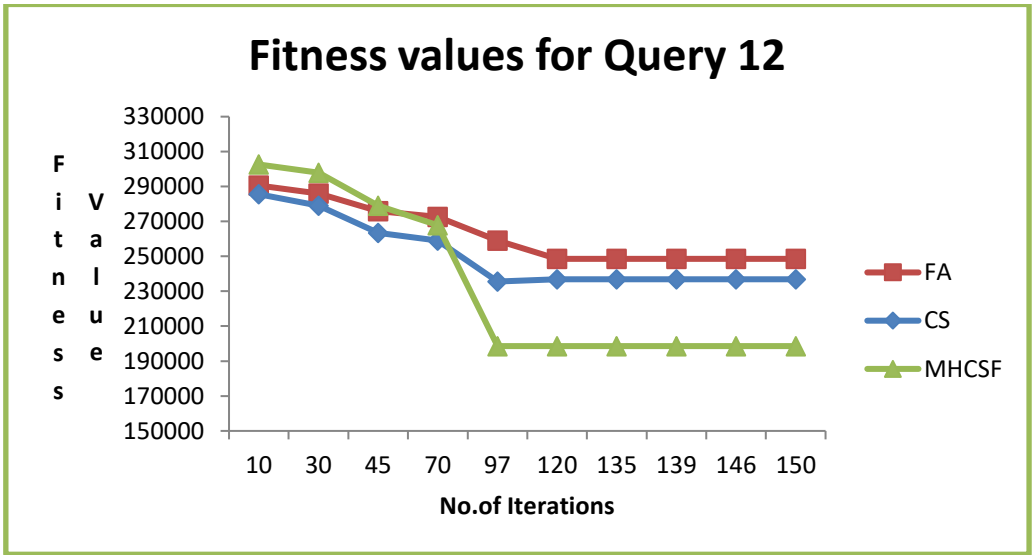


Fig. 4 Fitness Chart for Q12 with LUBM(3,0)

The query Q12 is based on the concept of realization. The query needs some inference and the input to the query is very small. The MHCSF algorithm generates a query plan with 198576 number of triples converging at the iteration number 97.The existing CS anf FA algorithms generates a query plan with 236845 and 248555 number of triples respectively.These plans have cost greater than the one generated by MHCSF algorithm.

**Comparison of Query Execution Time with LUBM(3,0) Dataset**

The execution times obtained with the LUBM(3,0) dataset for the proposed MHCSF algorithm is recorded and compared with existing algorithms as shown in Table 4. Sample queries with diverse number of predicates is considered for performance evaluation.

Table 4: Performance Comparison of MHCSF Algorithm

| **Dataset** | **Query** | **Query Execution Time in Seconds** | | |
|---|---|---|---|---|
| | | **FA** | **CS** | **MHCSF** |
| LUBM (3,0) | Q1 | 22.38 | 22.34 | 21.57 |
| | Q2 | 13.65 | 13.45 | 13.00 |


https://doi.org/10.37896/pd91.5/91515

ISSN: 0369-8963

| | | | | |
|---|---|---|---|---|
| | Q4 | 14.5 | 14.1 | 13.75 |
| | Q8 | 26.54 | 25.54 | 24.68 |
| | Q12 | 47.86 | 37.0 | 26.25 |

Query Q1 retrieves information about one class and property and it does not depend upon any inference or hierarchy. With the LUBM(3,0) dataset, the proposed MHCSF algorithm produces an execution time of 21.57 seconds, CS produces 22.34 seconds and FA produces 22.38 seconds to run the query. Query Q2 contains 3 classes and 3 properties, and so the complexity of solving the query increases. Also, there exist a triangular pattern of relationship between the objects.Hence for the LUBM (3, 0) dataset, MHSCF produces 13.00 seconds, CS produces 13.45 seconds and FA produces 13.65 seconds.

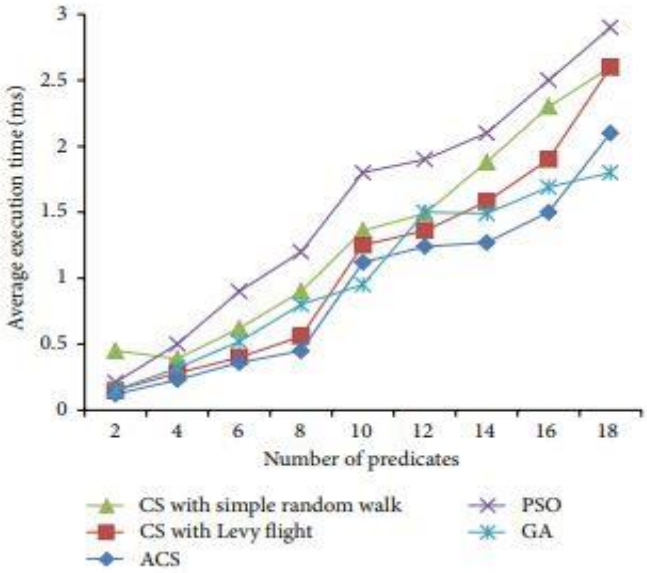


Fig. 5**:** Average execution time VS predicts

Query Q4 retrieves multiple properties within a single class. The execution time produced for LUBM(3, 0) dataset is 13.75 seconds by MHCSF algorithm, 14.1 seconds by CS and 14.5 seconds by FA.Query Q8 involves more complexity and so it takes an execution time of 24.68 seconds by the proposed MHCSF, 25.54 seconds by the CS and 26.54 seconds by the FA. The above Fig. 5 describes about the Average execution time VS predicts. Query Q12 is a complex query that involves the concept of realization and it takes an execution time of 26.25 seconds by MHCSF, 37.0 seconds by CS and 47.86 seconds by FA for the LUBM(3, 0) dataset. The Fig. 6 & 7 shows the execution times and plot between average execution and Fig. 8 predicts and retrieved in seconds for the benchmark test queries for the LUBM (3, 0) dataset compared with the existing systems.


https://doi.org/10.37896/pd91.5/91515

ISSN: 0369-8963

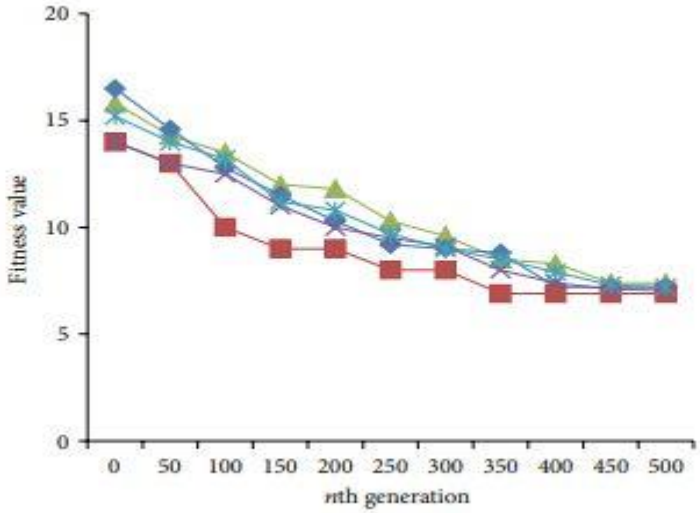


Fig. 6: Fitness value vs generation chart

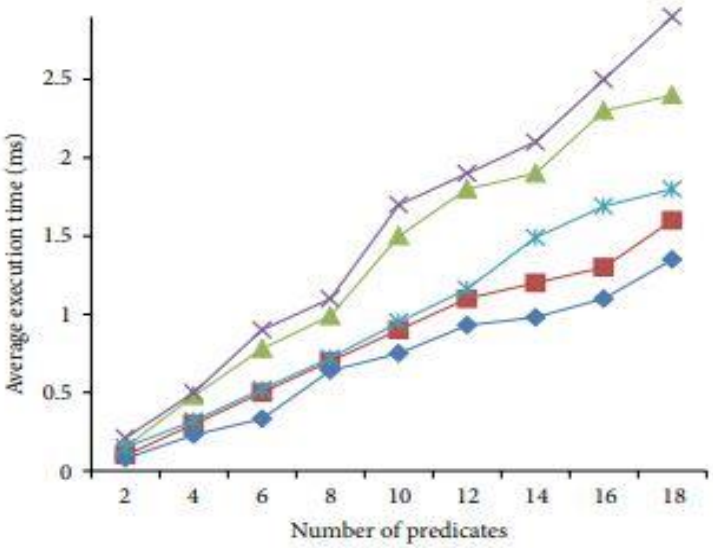


Fig. 7 Average Execution VS Predicts

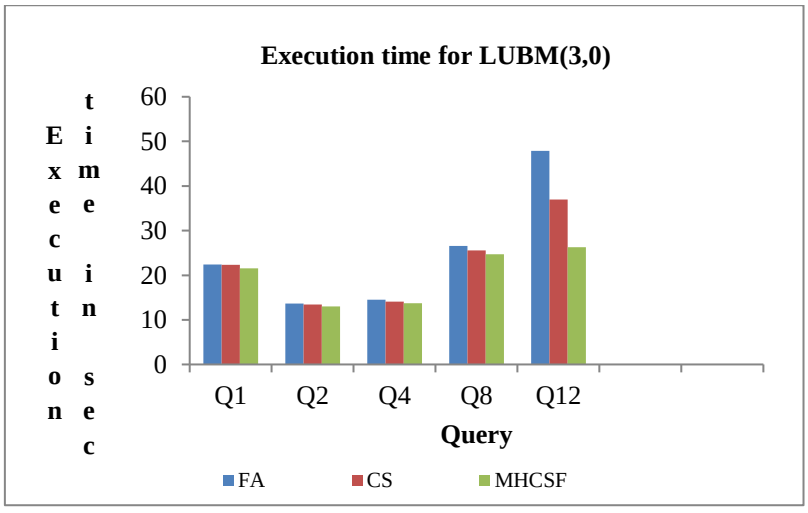


Fig. 8 Comparison of Query Execution Time for LUBM(3,0)

The above chart shows the performance of the proposed MHCSF algorithm in terms of its


https://doi.org/10.37896/pd91.5/91515

execution time for sample test queries. For all the sample queries, the proposed algorithm produces a lesser execution time compared to the existing algorithms.

## CONCLUSIONS

In this line of investigation, a modified hybrid of Cuckoo Search and Firefly algorithm is applied to solve the problem of SPARQL query optimization. The algorithm takes as input a set of query plans and finds the query plan with less cost as output. With the output query plan the execution time gets reduced. Benchmark dataset LUBM with three universities data are used. Sample queries are executed and the implementation records the query execution time. The query execution times are compared with the existing Cuckoo search and firefly approaches. The comparisons show that the proposed system reduces the query execution time and the system is suitable for querying large volume of Semantic web data.

## References

[1] M. Arenas and J. Perez, "Querying semantic web data with sparql". in *Proce. of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM Digital Library*, pp. 305-316, 2011. https://doi.org/10.1145/1989284.1989312.

[2] Z. Kaoudi and A. Kementsietsidis, "Query processing for rdf databases", *Lecture Notes in Computer Science, Springer*, vol. 8714, 2014. DOI: 10.1007/978-3-319-10587-1_3

[3] E. Sirin and B. Parsia, "SPARQL-DL: sparql query for owl-dl", in *Proce. Sirin2007, SPARQLDLSQ, Computer Science, Owled*, vol. 258, pp. 1-10, 2007.

[4] M. Steinbrunn, G. Moerkotte and A. Kemper, "Heuristic and randomized optimization for the join ordering problem", *The International Journal on Very Large Data Bases*, vol. 6, no.3, pp.191-208, 1997. https://doi.org/10.1007/s007780050040

[5] M. Elkhechafi, H. Hachimi and Y. Elkettani, "A new hybrid cuckoo search and firefly optimization", Monte *Carlo Methods and Applications*, vol. 24, no. 1, pp. 71-77, 2018. DOI: 10.1515/mcma-2018-0003.

[6] M. A. Tawhid and A. F. Ali, "An effective hybrid firefly algorithm with the cuckoo search for engineering optimization problems", *Mathematical Foundations of Computing*, vol. 1, no. 4, pp. 349-368, 2018. doi: 10.3934/mfc.2018017

[7] G. Klyne, "Resource description framework (rdf): Concepts and abstract syntax. http://www. w3. org/TR/2004/REC-rdf-concepts-20040210/. 2004

[8] V. Stojanovic, N. Nedic, D. Prsic, L. Dubonjic and V. Djordjevic, "Application of cuckoo search algorithm to constrained control problem of a parallel robot platform," *The International Journal of Advanced Manufacturing Technology*, vol. 87. no. 9, pp. 2497 - 2507, 2016. DOI:10.1007/s00170-016-8627-z

[9] J. Alami, L. Benameur and A. A. El Imrani, "A fuzzy clustering based PSO for multimodal optimization", *International Journal of Computational Intelligence Research*, vol. 5 no.2, pp. 96 -107, 2009.

[10] Y. Zhang, S. Xian and D Gong, "A return-cost-based binary firefly algorithm for feature selection", *Information Sciences*, vol. 418 - 419, pp. 561-574, 2017. https://doi.org/10.1016/j.ins.2017.08.047

[11] B. Selvakumar and K. Muneeswaran, "Firefly algorithm based feature selection for network intrusion detection", *Computers and Security*, vol. 81, pp. 148-155, 2019. DOI:10.1016/j.cose.2018.11.005

[12] M. Sainte and N. Alalyani, "Firefly algorithm based feature selection for Arabic text classification", *Journal of King Saud University - Computer and Information Sciences*, vol. 32, no. 3, pp. 320–328, 2020. https://doi.org/10.1016/j.jksuci.2018.06.004.

[13] I. Fister, X. Yang and B. Janez, "A comprehensive review of firefly algorithms", *Swarm and Evolutionary Computation*, vol. 13, pp. 34-46, 2013. https://doi.org/10.48550/arXiv.1312.6609

[14] M. Joshi, and P.R. Srivastava, "Query optimization: an intelligent hybrid approach sing cuckoo and tabu search, *International Journal of Intelligent Information Technologies*, vol. 9, no. 1, pp. 40-55, 2013. DOI:10.4018/jiit.2013010103

[15] S. Debnath, K. Ramalakshmi and M. Senbagavalli, "Multimodal authentication system based on audio-visual data: a review", in *Proce. 2022 International Conference for Advancement in Technology (ICONAT), IEEE, 21660941*, Goa, pp. 1-5, 2022. DOI:10.1109/TENCON.2019.8929592

[16] B. H. Abed-Alguni and D. J. Paul, "Hybridizing the cuckoo search algorithm with different mutation operators for numerical optimization problems, *Journal of Intelligent Systems*, vol. 29, no. 1, pp. 1043-1062, 2018. https://hdl.handle.net/1959.11/26644.

[17] M.Babu and T.Jesudas, "An artificial intelligence-based smart health system for biological cognitive detection based on wireless telecommunication", *Computational Intelligence*, Vol. 38, no. 4, pp. 1365-1378, 2022.

[18] Jesudas T, Ramesh S, Arunachalam RM and Senthilkumar K, "Study and optimization of overcut on most influencing process parameters of micro edm, *International Journal of Applied Engineering Research*, vol. 10, no. 50, 13678-13683, 2015.