Modified Firefly Algorithm for Optimizing Biomedical Breast Cancer Queries

Gomathi Ramalingam (gomsbk2020@gmail.com)
Bannari Amman Institute of Technology Department of Computer Science and Engineering

Original Research

Keywords: Biomedical queries, breast cancer, Firefly algorithm, query optimization, Semantic Web, triples

Posted Date: February 2nd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-171211/v1

Abstract

Querying and retrieving Semantic Web data is a challenging task due to the increment in its volume. Many query languages were designed to retrieve Semantic Web data. A popular querying method of communication in Semantic Web is SPARQL. The query languages were designed with some optimization strategies, and it was found in literature that these query languages were not able to handle large volume of data efficiently. In this research, a Modified Firefly Algorithm (MFA) is applied to optimize the SPARQL queries so that it can retrieve data from a large Semantic Web repository efficiently by reducing query execution time. Every query will have multiple query plans generated with different cost values. The challenge is to choose the best query plan which reduces the query cost and query execution time. The proposed algorithm uses the best query plan in the previous iteration to calculate the distance between two query plans using the radius parameter. The proposed algorithm generates a query plan which is a global optimal solution. MFA is evaluated using the BioPortal dataset with triples containing breast cancer. Experimental analysis is conducted to identify the significant improvement in performance of the proposed work with the existing nature inspired query optimization algorithms. The efficiency of MFA is compared with other algorithms in terms of query execution time and the performance is evaluated.

1. Introduction

In biomedical field, biology- and medical-related information is combined to provide information related to different types of diseases. Biomedical data from knowledge bases are well represented using Resource Description Framework (RDF) triples. Large-scale health-related datasets from various heterogeneous domains are linked together and represented as Semantic Web data. There is a necessity to provide absolute results to the patients from querying such vast amount of data. Several research initiatives have been undertaken to query these bulk data depending on technologies like natural language processing[4], evolutionary optimization algorithms[7] and heuristic algorithms[9]. Owing to the bulky size and complexity of the data, efficient retrieval mechanisms are necessary in today's era as biomedical queries are challenging to search engines [1].

RDF is one of the common ways to represent the Semantic Web data, and SPARQL is the popular endpoint to query such data. The results of SPARQL queries can be retrieved in diverse formats. There is a requirement for novel architectures, which manage bulk RDF data and queries [2]. Federated query engines have been proposed in literature to access the biomedical data sources [3]. Several researchers have been working on the access to the decentralized contents of biomedical data.

Query optimization is an important step in the process of processing a query. Every query issued may have multiple query plans generated and each query plan will be executed with a different query cost and varying query execution time. The issue with processing large volume of semantic web data is the increase in query execution time with a scalable increase in large volume of data. Usually traditional methods of query optimization are applied to generate the optimal query plans. The new way of

optimizing and generating the optimal query plan among the varying plans available is the application of nature inspired and evolutionary methods.

The nature inspired methods uses behaviour of different nature related aspects to the problem of interest and produces the result. Many natures inspired and evolutionary algorithms are available and applicable to solve this problem. Based on the parameters they depend on an acceptable nature inspired algorithm is chosen to solve the problem of query optimization.

The article is arranged as follows. Section 2 discusses an examination of the most recent works on the query optimization. Section 3 discusses the methodology of the Modified Firefly Algorithm (MFA) to optimize breast cancer queries. Section 4 describes the query optimization, problem statement, proposed encoding representations of query plans, and fitness functions. In Section 5, the implementation results for the optimization of biomedical queries using breast cancer dataset are discussed and compared with those of the other algorithms and also their efficiencies are analysed. Section 6 presents the conclusions.

2. A Review Of Literature

Hamon *et al*. [4] proposed a method for converting questions in natural language into SPARQL queries. It was a rule-based method, which depends on linguistic and semantic annotation of questions using Semantic Web resources and RDF triples. Oren *et al*. [5] proposed a technique for querying RDF data through evolutionary algorithm. The solutions were evaluated using fingerprint and Bloom filters. The trade-off between computation time and results of the query was addressed in the research. Hees *et al*. [6] devised an evolutionary pattern learner to identify patterns in a dataset using evolutionary algorithm.

Hogenboom *et al.* [7] adapted an efficient method to optimize chain queries in a distributed environment. Various evolutionary optimization algorithms such as Ant Colony Optimization (ACO), two-phase optimization, and Genetic Algorithm (GA) for optimizing chain queries were compared in the research. Saharan *et al.* [8] invented a solution to address the querying of scattered data using Differential Evolution. The algorithm uses the policy of reorganization of the order of triples pattern. The results were compared with other heuristic algorithms and were analysed.

Tsialiamanis *et al.* [9] proposed a new Heuristic SPARQL Planner to take advantage of the syntax and structural variations in triples. The proposed algorithm chooses an optimized query plan without depending on a cost model. The planner was implemented using an open source column-store and the quality of the query plan was compared with benchmarks. Abbas *et al.* [10] proposed an optimization algorithm to optimize the Conjunctive SPARQL queries, which is based on considering the ShEx constraints.

Wang *et al.* [11] proposed a skeleton to optimize the working of distributed SPARQL queries. The proposed algorithm is based on graph traversal and its performance is evaluated based on comparison with non-graph traversal algorithms. Ibragimov *et al.* [12] proposed MARVEL (Materialized Rdf Views with Entailment and incompleteness), a method that depends on the RDF cost model. This algorithm rewrites

SPARQL queries based on materialized RDF views and view definition syntax. The investigational analysis proved that the MARVEL system improved the query retort time efficiently.

A comparison of evolutionary algorithms applied to the problem of query optimization in distributed databases was [15] discussed in literature. The research focussed on bio-inspired computational algorithms used in optimizing distributed database queries. A new genetic algorithm based query optimizer was proposed in literature which compares the performance of existing algorithms with random algorithms. The analysis proves that genetic algorithm seems to be a feasible approach to handle large scale distributed systems. A multi colony ant algorithm was proposed based on MIN-MAX ant system to optimize distributed database queries. The proposed algorithm handles scalability and modularity. The algorithm proves that it takes less computation time to optimize the queries. A hybrid of particle swarm optimization algorithms was developed which depends on the probability distribution of movement of particles. Memetic algorithms which includes the local search process to refine the search of best optimal query plan generation was proposed in research. The literature study shows that there are significant amount of bio-inspired algorithms available for generating optimal query plans.

The different cost models associated with the big data systems were studied in research [16]. The currently available cost models depending on the cardinality measure and the learned cost model depending on operators, common sub expression were also discussed. The basic and derived features of the cost model were studied and feature selection mechanisms were proposed in literature.

The following table 1 shows the compares [17] the existing evolutionary algorithms and their corresponding features.

Table 1: Comparison of existing evolutionary algorithms

S.No	Algorithm /Feature	PSO	GA	FA
1	Behaviour	Social behaviour	Biological genetic process	Behaviour of fireflies
2	Parameters	Population size, number of generations, particle velocity, current position	Population size, number of generations, mutation rate, crossover rate	Population size, Attractiveness, visibility
3	Locality	Population based search with position and velocity	Population based search	Population based search
4	Flexibility	Flexible	Not flexible	Flexible

3. Mfa: A Modified Firefly Algorithm

Nature inspired algorithms are the most popular algorithms to solve the optimization problem. Meta-heuristic algorithms are designed to generate good solutions to an optimization problem by making a few assumptions. Firefly algorithm is one such meta-heuristic algorithm developed by Yang [13] for solving optimization problems. The algorithm was configured such that it inspires the behaviour of a firefly. The three behaviours of fireflies include the following:

- Every firefly creates a centre of attention on another firefly using its brightness.
- Fireflies having superior brightness have elevated attractiveness to the other fireflies.
- Fireflies having inferior brightness move to other fireflies with superior brightness.

The second behaviour is analogous to the fact that newly produced solutions are based on previous solutions having an enhanced fitness function. Theoretically, for each query, there may be either one or more new possible query plans.. This depends on the query cost opinion between solution and other solutions among the existing population of query plans.

In the standard firefly algorithm the best query plan generated is the global optimal solution. If there is a movement of this firefly as in standard firefly algorithm, the brightness of the firefly decreases which affects the performance of generating the query plan with less execution time. In conventional firefly algorithm, the distance is calculated by considering the current solution and another better solution. Instead of doing this, MFA uses the best solution value in the previous iteration to calculate the radius. This modification is done to make the firefly move towards the global optimal solution.

Suppose that for every solution *i*, X_i is a location of *i* th firefly at the present iteration. When the objective function value of solution *i* is superior than that of solution *j*, the distance between the two fireflies *i* and *j* is obtained using the following equation:

$$r_{ij} = \text{sqrt}(X_i - X_j)^2 \qquad (1)$$

Then, the updated distance is used into Equation (2) to work out a new attractiveness. The new position for the *i*th firefly can be calculated matching to the generation of a new solution of the *i*th solution. The formula of generating a new solution is as follows:

$$\beta = \beta_0 e^{-\gamma rij2} \qquad (2)$$

$$X_{ij\text{new}} = X_i + \beta \,\text{rand}\, \Delta X_{ij} + \text{rand} \qquad (3)$$

Where r and is any random number of solution *i*; β_0 is the attractiveness at zero distance and normally set to 1; γ is set to 0 which indicates high visibility of query plans in this case. X_j is a solution having lower

fitness function than X_i and ΔX_{ij} is a revised step size, which is calculated by the following equation:

$$\Delta X_{ij} = X_j - X_i \qquad (4)$$

The flowchart for the proposed work is given below. In Figure 1 Flowchart of proposed MFA and Figure 2 *Algorithm for proposed MFA.*The steps for the proposed MFA are as follows.

3.1 RDF Query Plans

Semantic Web facts stored in the RDF version can be retrieved using the popular SPARQL language. In the context of query optimization, every SPARQL query must be imagined by means of a query tree. The leaf nodes in the query tree stand for the triples. The internal nodes are meant to join the triples. Query trees can be represented in different formats that include bushy trees, left-deep trees, and right-deep trees. The nodes in a single query tree can be restructured in many dissimilar ways to turn out the same results. The sequence in which methodologies are executed to regain the requested data is known as query plan. In this line of investigation, the trees used to represent the query paths are left-deep trees [14]. Left-deep trees are chosen because these trees can process the joins by applying cost-reducing pipelining methods.

3.2 Solution Space

Generally, solution space consists of a set of all processing trees that can produce a result for a given query. In the context of query optimization problem, the solution space contains query execution plans as solutions. There are *n*! feasible means to assign *n* triples to the leaf of the tree. The leaves of the tree consist of triples and the inner nodes are used to join these triples. The *n*! solutions are obtained by applying the conversion set of laws.

3.3 Encoding Methodology

For every optimization algorithm, we need to choose an encoding methodology, a suitable encoding for the solutions in the solution space. In this research, the solutions are query plans and they are symbolized using left-deep trees. To encode left-deep trees, we choose ordered list. Solutions are represented using an ordered list [14] of leaves. After generating all possible query plans for the given SPARQL query, the trees are encoded using the ordered list format. For example, consider the query tree with nodes and joins in the following order: ((((R1⋈ R2) ⋈R3) ⋈ R4) ⋈R5) is encoded as “12345”.

3.4 Objective Function

To determine RDF query path, we must first come to a decision on the fitness function. In this research work, the fitness function refers to the cost model of the left-deep tree. The following formula is used to define the cost model of the query tree:

Cost of single query tree = Sum of the costs of each operator involved in the tree

The cost of a particular operator depends on the cardinality and selectivity estimation.

The intention of this research is to reduce the query execution time to retrieve data from a large volume of Semantic Web data. After deciding the aforementioned parameters such as encoding methodology, cost model, and solution space, they are given as input to MFA. The algorithm takes different query paths as input, converts the query plans to some encoding format, and applies cost function to determine the cost of the query plans. After executing MFA with all the inputs defined, the algorithm returns the best query plan that reduces the execution time.

4. Experimental Results With Breast Cancer Queries

This section identifies the performance of the proposed work with breast cancer datasets and also compares the performance of the proposed and existing algorithms.

4.1 Datasets

BioPortal [4] is a dataset that consists of a repository of biomedical ontologies in many different formats including RDF. It consists of 190M triples and manually and automatically generated cross-ontology mappings. The American Cancer Society have mentioned in their analysis Breast cancer facts and figures that “Breast cancer affects one in eight women during their lives”. The proposed algorithm is applied to sample SPARQL queries and with 1,00,000 breast cancer triples for testing. The RDF data can be queried using SPARQL endpoint.

4.2 Performance Comparison

Figures 3–5 show the fitness values obtained by applying MFA to SPARQL queries of breast cancer datasets.

The main performance metric used to assess the working of the proposed algorithm is the query execution time. The query execution times obtained for different queries are compared with algorithms such as Particle Swarm Optimization (PSO) and GA. Clearly, the proposed MFA reduces query execution time. Table 2 shows the comparison of sample query execution times of proposed and existing algorithms.

Table 2: Comparison of query execution times in milliseconds

Query	PSO	GA	MFA
Query 1	21.4	19.56	19.47
Query 2	47.2	45.0	44.86
Query 3	19.6	18.0	17.88
Query 4	59.6	57.2	57.0
Query 5	43.66	41.9	41.36

The following paragraph could just summarize the differences obtained with the MFA in relation to the other algorithms.The query execution time for sample Query 1 is 21.4 ms for PSO and 19.56 ms for GA. After optimizing using MFA, the query execution time reduces to 19.47 ms. The query execution time for sample Query 2 is 47.2 ms for PSO and 45 ms for GA. After optimizing using MFA, the query execution time reduces to 44.86 ms. The query execution time for sample Query 3 is 19.6 ms for PSO and 18 ms for GA. After optimizing using MFA, the query execution time reduces to 17.88 ms. The query execution time for sample Query 4 is 59.6 ms for PSO and 57.2 ms for GA. After optimizing using MFA, the query execution time reduces to 57 ms. The query execution time for sample Query 5 is 43.66 ms for PSO and 41.9 ms for GA. After optimizing using MFA, the query execution time reduces to 41.36 ms.

The advantage of the modified firefly algorithm(MFA) is that it increases the visibility to search for the optimal query plan from the available plans. This mechanism increases the possibility of choosing the best plan with minimum execution cost as well as execution time. The choice of choosing the left deep tree representation of the query plan and the ordered list encoding methodology also contributes to the working of the proposed algorithm to produce the best results.

The comparison of the query execution times is shown graphically in Figure 6.

5. Conclusions

1. In this research, MFA – a Modified Firefly Algorithm – is used to optimize the SPARQL query plans.
2. The algorithm takes query plans as input and uses cost of query plans as a fitness function. The proposed algorithm applied a modified version of Firefly algorithm called MFA, which uses best query plan in previous iteration to calculate the radius value. This modification optimizes the query and computes the best query plan.
3. Sample queries of BioPortal datasets are executed and the query execution times are recorded. The query execution times are compared with those of already existing conventional algorithms such as PSO and GA. The proposed algorithm is found to reduce the query execution time.
4. The proposed algorithm is designed to handle large semantic web datasets and its corresponding queries. The advantage of the proposed system resides in generating the optimal query plan among

many plans with different cost.

5. The cost model used with the proposed system depends upon the heuristics and statistics information like cardinality. With larger datasets, the cost model may be improved by considering hardware and software configurations of the system.
6. Experimenting nature inspired algorithms for optimization with different cost models may be implemented as a future contribution.

References

1. N. Senthilselvan, V. Subramaniyaswamy, and K.R. Sekar, “Information –-retrieval for biomedicine applications through linked open data based optimization model,” *J. Biomed. Semantics*, vol. 8, pp. 7867-7870, 2017.
2. J. Liu, M. Yang, L. Zhang, and W. Zhou, “An effective biomedical data migration tool from resource description framework to JSON,” *Database*, vol. 2019, 2019.
3. Y. Khan and R. Sahay, “SPARQL Query Federation over Biomedical Data,” Insight Centre for Data Analytics, National University Literature, 2019.
4. T. Hamon, N. Grabar, and F. Mougin, “Querying biomedical linked data with natural language questions”, *Semantic Web*, vol.8, pp.581-599, 2017.
5. E. Oren, C. Guéret, and S. Schlobach, “Anytime query answering in RDF through evolutionary algorithms,” *International Semantic Web Conference*, pp. 98-113, 2008.
6. J. Hees, R. Bauer, J. Folz, D. Borth, and A. Dengel, “An evolutionary algorithm to learn sparql queries for source-target-pairs,”, European Knowledge Acquisition Workshop, 2016, pp. 337-352.
7. A. Hogenboom, A. E. Niewenhuijse, M. Jansen, F. Frasincar, and D. Vandic, “RDF chain query optimization in a distributed environment,” *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pp. 353-359, 2015.
8. S. Saharan, J. S. Lather, and R. Radhakrishnan, “Optimization of different queries using optimization algorithm (DE),” *In J Comput Network Inf Secur.*, vol. 11, pp. 52, 2018.
9. P. Tsialiamanis, L. Sidirourgos, I. Fundulaki, V. Christophides, and P. Boncz, “Heuristics-based query optimisation for SPARQL,” *Proceedings of the 15th International Conference on Extending Database Technology*, pp. 324-335, 2012.
10. A. Abbas, P. Genevès, C. Roisin, and N. Layaïda, “Optimising SPARQL query evaluation in the presence of shex constraints,” *Gestion de Données - Principes, Technologies et Applications*, pp. 1-12, 2017.
11. X. Wang, T. Tiropanis, and H. Davis, “Evaluating graph traversal algorithms for distributed SPARQL query optimization,”, *Joint International Semantic Technology Conference*, pp. 210-225, 2011.
12. D. Ibragimov, K. Hose, T. B. Pedersen, and E. Zimányi, “Optimizing aggregate SPARQL queries using materialized RDF views”, *Proceedings of International Semantic Web Conference, Springer*, pp. 341-359, 2016.

13. X. S. Yang, X. S, “Firefly algorithms for multimodal optimization,” *International Symposium on Stochastic Algorithms*, pp. 169-178, 2009.
14. M. Steinbrunn, G. Moerkotte, and A. Kemper, “Heuristic and randomized optimization for the join ordering problem,” *VLDB J*, vol.6, pp. 191-208, 1997.
15. Ali, Z., Kiran, H. M., & Shahzad, W. (2018). Evolutionary Algorithms for Query Optimization in Distributed Database Sys-tems: A review. Advances in Distributed Computing and Artificial Intelligence, Vol. 7 N. 3 (2018), 115-127
16. Siddiqui, T., Jindal, A., Qiao, S., Patel, H., & Le, W. (2020, June). Cost models for big data query processing: Learning, retrofitting, and our findings. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (pp. 99-113).
17. Garg, A., & Juneja, D. (2012). A Comparison and analysis of various extended techniques of query optimization. International Journal of Advancements in Technology, 3(3), 184-194.

Declarations

Competing interests

The authors declare no competing interests.

Author

Dr. R. Gomathi received the B.E. degree from Periyar University and M.E. degree from Anna University. She completed her Ph.D. in Computer Science and Engineering from Anna University, Chennai, in 2016. Her research interests include Query processing, Semantic Web, Optimization techniques, and database management.

Figures

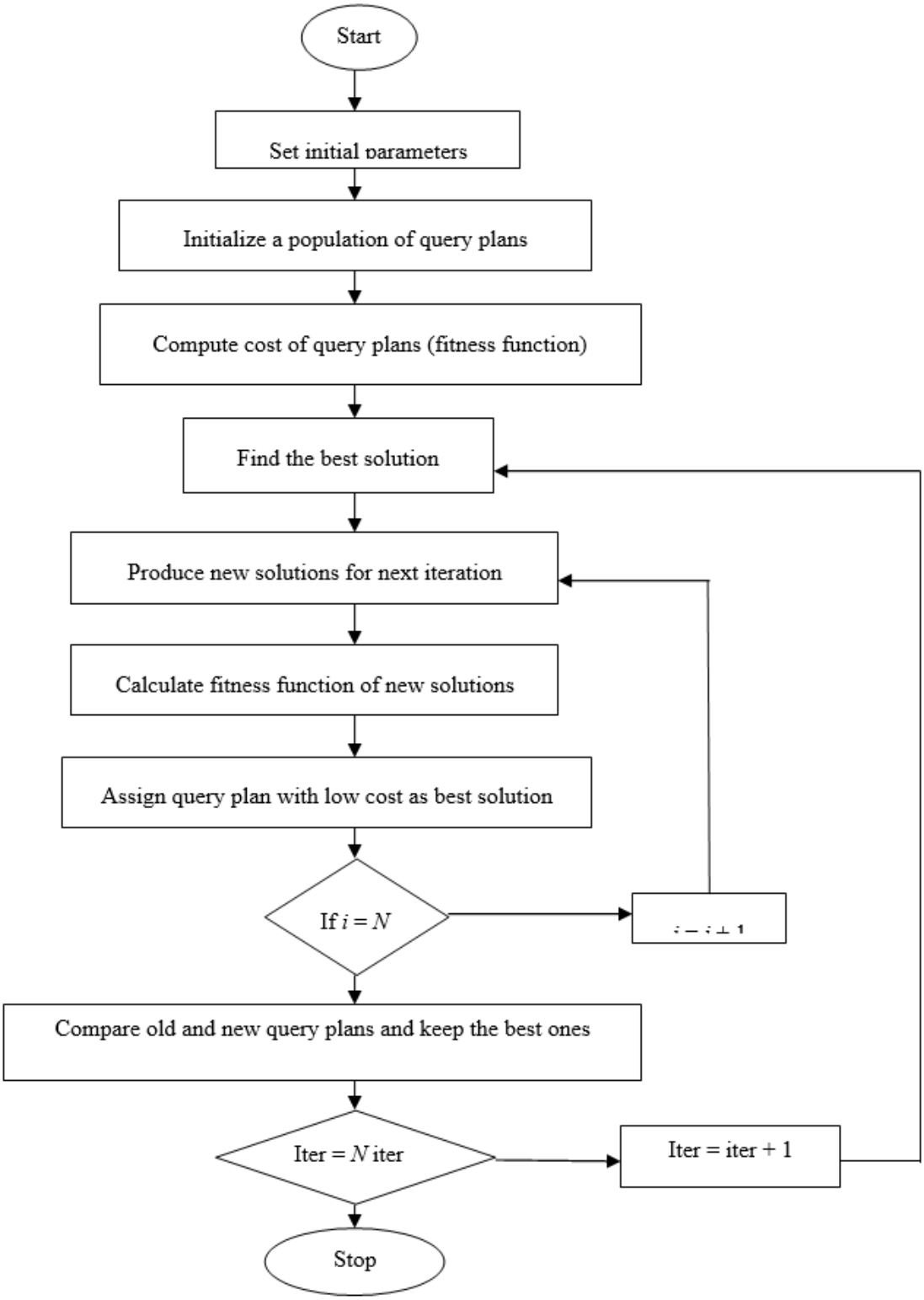

Figure 1

Flowchart of proposed MFA

Step 1: Set initial parameters

Step 2: Initialize a population of query plans

Step 3: Compute cost of query plans using fitness function

Step 4: Find the best solution

Step 5: Produce new solutions for next iteration

Step 6: Determine the fitness values for new solutions

Step 7: Assign query plan with low cost as best solution

Step 8: Until the number of iterations is equal to N, repeat from step 5

Step 9: Compare old and new query plans and keep the best ones

Step 10 Repeat from step 4 until the best query plan is obtained

Step 11: Stop

Figure 2

Algorithm for proposed MFA

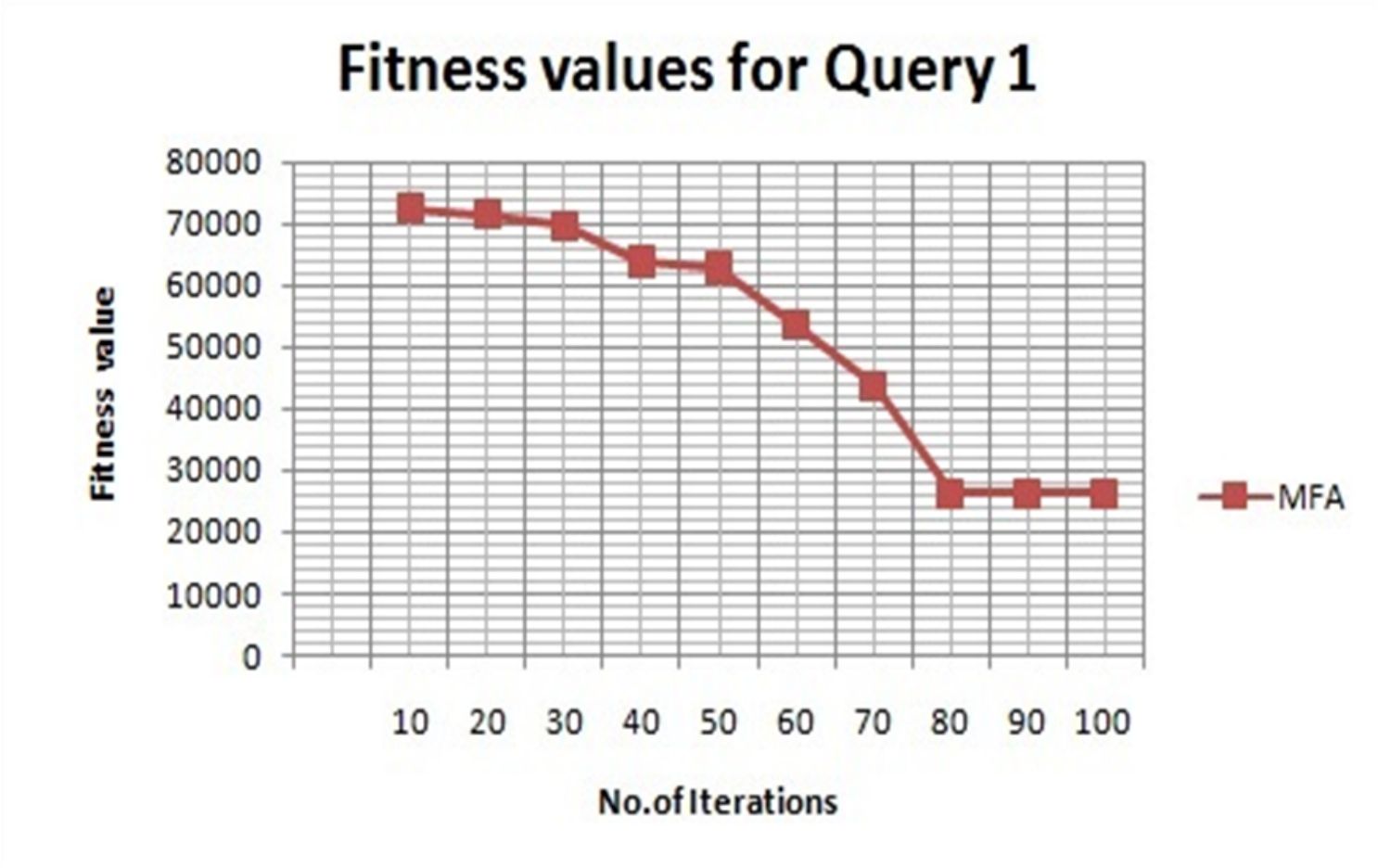

Figure 3

Fitness values for Query 1

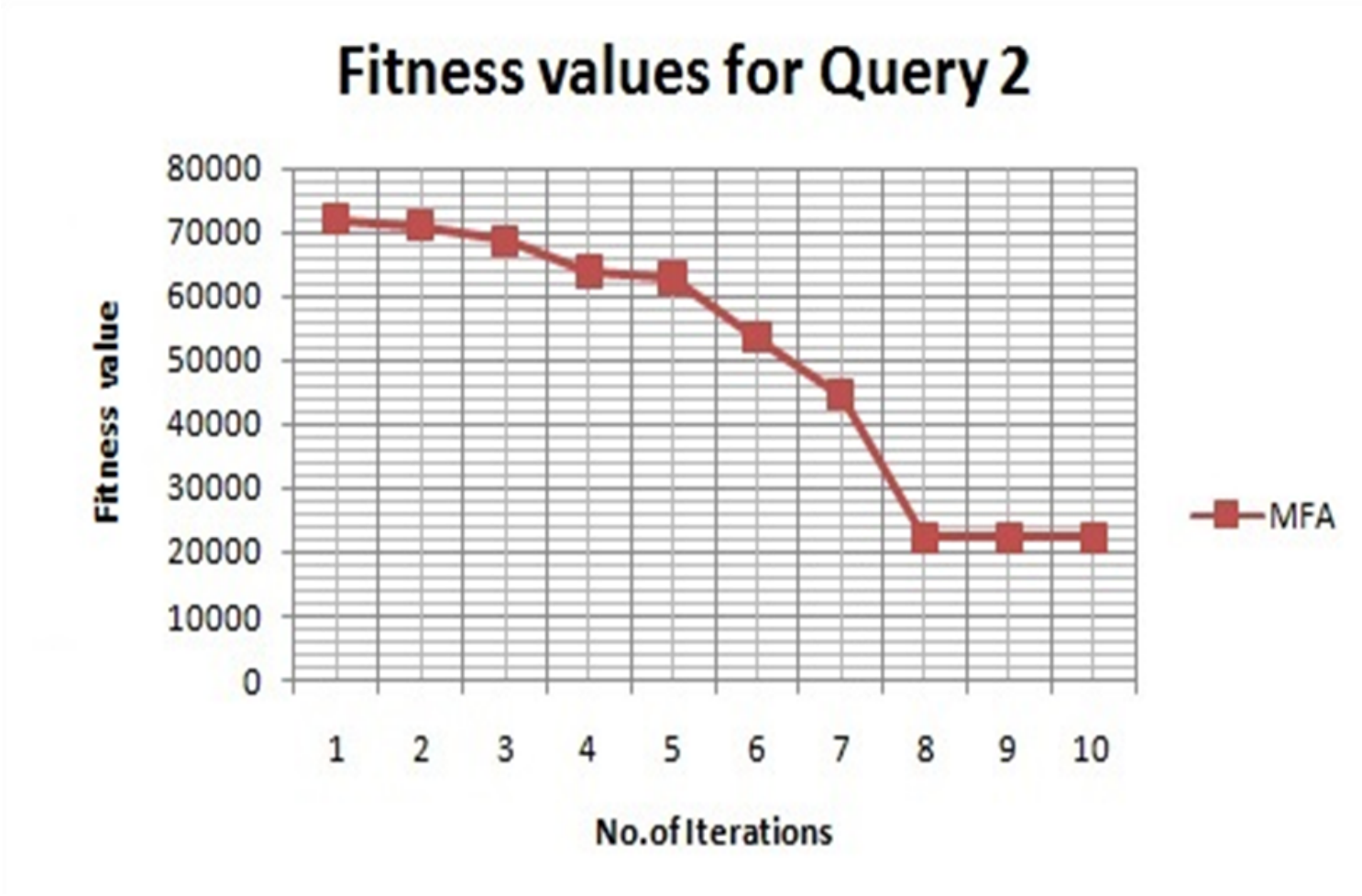

Figure 4

Fitness values for Query 2

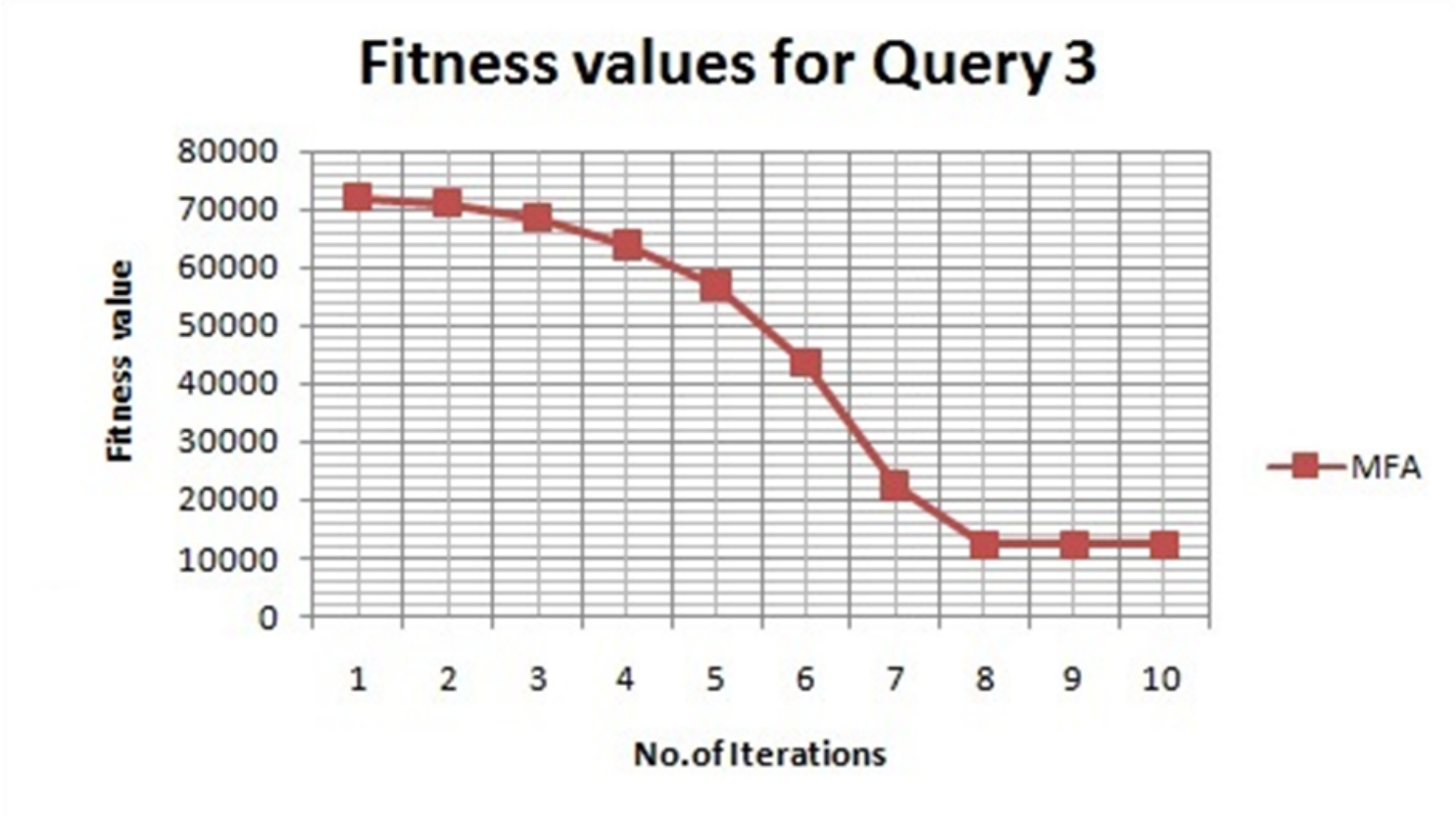

Figure 5

Fitness values for Query 3

Figure 6

Comparison of query execution times