Application of SAPSO Algorithm to Optimize SPARQL Queries with OPTIONAL Blocks

Gomathi R¹, Gomathy B²
¹²Department of Computer Science and Engineering, INDIA

ABSTRACT

Semantic Web technology is an emerging technology developed as a result due to the rapid development of multiple web pages. Semantic web stores facts in many different formats. One of the popular formats is the Resource Description Framework (RDF) format. When the amount of semantic web data increases, querying large RDF graphs becomes a tiresome process. Also the problem of query optimization becomes a concern in querying large RDF graphs. Choosing the best query plan reduces the amount of query execution time. To address this problem, nature inspired algorithms can be used as an alternative to the traditional query optimization techniques. In this research, the optimal query plan is generated by applying the SAPSO algorithm which is a hybrid of Simulated Annealing (SA) and Particle Swarm Optimization (PSO) algorithms. In this research, the SAPSO algorithm is applied to queries having OPTIONAL blocks. The SAPSO algorithm finds the local optimistic solution and it avoids the problem of local minima. Experiments were performed on datasets with varying number of predicates. The algorithm applied in this research gives improved results compared to existing algorithms in terms of query execution time.

Keywords— Nature Inspired algorithms, Query optimization, Particle swarm optimization, Semantic web, Simulated Annealing

I. INTRODUCTION

Computers cannot easily understand the data provided by the web pages but human beings can. A solution for the machines to interpret the information in the web pages is by the development of the semantic web technology. This technology allows machines to easily search, combine and manipulate information in the web. Information in the semantic web can be represented using many formats like the RDF, Web Ontology Language (OWL), and Extensible Mark-up Language (XML). The most common of them is the RDF format. RDF is a data model to characterize information in the web. It uses the syntax of XML and Uniform Resource Identifiers (URI) to identify resources. Each resource can be expressed using property and property values in RDF.

A query language for querying data in the semantic web is the SPARQL protocol and RDF Query Language (SPARQL). This query language can be used to query data from diverse sources, whether the data stored in native RDF format or it is viewed as RDF by means of any middleware. Nature has been growing for quite a lot of years, and it has been providing many ingenious solutions to solve many problems. There are many sources of nature which includes swarm intelligence, biological systems, physics based or chemistry based systems. A subset of metaheuristics is often referred to as Swarm Intelligence (SI) based algorithms have been developed by imitating the so called swarm intelligence characteristics of biological agents such as birds, fish, humans, and so on. Examples include, Particle Swarm Optimization (PSO) based on the swarming behavior of birds and fish [10], the Firefly Algorithm (FA) based on the blinking pattern of tropical fireflies [11], and Cuckoo Search (CS) algorithm [12] inspired by the brooding parasitism of some cuckoo species. Physics and chemistry based algorithms include Harmony Search, Simulated Annealing [13] and so on. Nature inspired algorithms are applied for solving large scale problems. Hybrids of these algorithms are also promising to produce best results. The paper is organized as follows: Section II reviews the methods used for query optimization in literature; Section III makes a study of the SAPSO algorithm; Section IV explains the application of the SAPSO algorithm to optimize the SPARQL queries with OPTIONAL blocks; Section V describes the datasets used and the experimental results; Section VI sketches the conclusions and future works.

II. LITERATURE REVIEW
The basic concepts connected with efficient processing [1] of SPARQL queries was studied in research. The study concentrates on i) the investigation of complexity analysis of all operators in SPARQL query language ii) equivalences of SPARQL algebra iii) optimizing SPARQL queries. The complexity analysis showed that all fragments of SPARQL fall into the class of NP.

There is a necessity to optimize the join of the partial query results. An Ant Colony System (ACS) [2] was proposed in literature to query the semantic web environment effectively. The enhancement in solution costs was compared with the existing algorithms like Genetic Algorithm (GA) and two phase optimization (2PO). It was shown that the ACS approach outperforms the existing approaches.

To optimize a specific class of SPARQL queries called RDF chain queries, a new genetic algorithm was devised called the RCQ-GA[3].The algorithm optimizes the chain queries by finding the order in which joins are to be carried out. The research work compares the performance of the algorithm with two phase optimization and the results demonstrate the quality of the solution and reliability of the solution.

A new join ordering algorithm based on cardinality assessment [4] was proposed in literature for SPARQL query optimization. Experiments conducted on star queries and arbitrary queries show that optimal query plans were found and executed in less amount of time.

A new set of PSO algorithms to optimize queries in distributed databases [5] was offered in research. The particles representing solutions are moved with respect to a probability distribution. The algorithm looks for the near optimal quality execution plans. By varying the constraint setting of PSO, the execution times can be accustomed.

A parallel join algorithm was deliberated to handle RDF [6] and its query language SPARQL. The algorithm combines three join algorithms and it processes multiple queries in an interleaved manner. The performance results for a variety of data sets were discussed.

A language for querying graphs was projected in research [7] called the G-SPARQL. A hybrid query engine which divides the query plan and its parts are pushed inside the relational database and some parts are processed using memory based algorithms. The effectiveness and scalability of the proposed approach was discussed.

To evaluate SPARQL queries, a RDF query engine was designed in literature [8]. The evaluation mechanism uses an index structure to index the RDF triples. A tree shaped optimization algorithm was designed to convert the SPARQL query graph into an optimal query plan. Experiments were conducted on both real and synthetic datasets to compute the performance.

To reduce the query responding time, a cost model [9] using Map Reduce framework was explained to explore the scalability of RDF data. The search space is diminished by using a algorithm called All-possible-joint tree (APJ-tree) algorithm. A hybrid joins and a bloom filter to speed up the processing of joins was applied. Tests were performed and the results were compared with the state of art solutions.

III. SAPSO ALGORITHM

The following algorithm[14] shows the steps of the SAPSO algorithm for generating optimal query plan,

1. Initialize a population of N query plans and a Temperature value T. For the \(i^{th}\) query plan, its location \(X_i\) in the search space is randomly placed. Its velocity vector is \(V_i = (v_{i1}, \ldots, v_{id}, \ldots, v_{id})\), in which the velocity in the \(d^{th}\) dimension is \(V_{id} = rand() \times V_{max}\), where \(rand()\) is the random number in the range [-1, 1].

2. Assign value for \(c_1\), \(c_2\), and \(w\). \(pbest\) is the current best position of the particle and \(gbest\) is the current global best position of the particle.

3. Compare the evaluated fitness value of each query plan with its \(pbest\). If the current value is better than \(pbest\), then set the current location as the \(pbest\) location. Furthermore, if the current value is better than \(gbest\), then reset \(gbest\) to the current index in the particle array.

4. Change the velocity of the particle using the following equation
\[ V_{id} = V_{id} + c_1 \times rand() \times (P_{id} - X_{id}) + c_2 \times rand() \times (g_{id} - X_{id}) \] (1)

5. Update the location of the particle and SA operator
\[ X_{id} = X_{id} + V_{id} \] (2)

6. Reduce the Temperature value using the following equation
\[ T = T \times 0.95 \] (3)

7. Repeat Steps 3-5 until the number of iteration is greater than the allowable maximum iteration number \(T_{max}\) or till the optimum query plan achieved.

IV. APPLICATION OF SAPSO ALGORITHM TO OPTIMIZE QUERIES WITH OPTIONAL BLOCKS

The following algorithm [14] shows the steps of the application of SAPSO algorithm to the SPARQL queries with OPTIONAL blocks for generating optimal query plan.

1. Initialize a population of N query plans and a Temperature value T. For the \(i^{th}\) query plan, its location \(X_i\) in the search space is randomly placed. Its velocity vector is \(V_i = (v_{i1}, \ldots, v_{id}, \ldots, v_{id})\), in which the velocity in the \(d^{th}\) dimension is \(V_{id} = rand() \times V_{max}\)
where \( \text{rand()} \) is the random number in the range \([-1, 1]\).

2. Assign value for \( c1 \), \( c2 \), and \( w \). \( \text{pbest} \) is the current best position of the particle and \( \text{gbest} \) is the current global best position of the particle.

3. Compare the evaluated fitness value of each query plan with its \( \text{pbest} \). If the current value is better than \( \text{pbest} \), then set the current location as the \( \text{pbest} \) location. Furthermore, if the current value is better than \( \text{gbest} \), then reset \( \text{gbest} \) to the current index in the particle array.

4. Change the velocity of the particle using the following equation

\[
V_{id} = V_{id} + c1 \times \text{rand}() \times (P_{id} - X_{id}) + c2 \times \text{rand}() \times (P_{id} - X_{id})
\]  

5. Update the location of the particle and SA operator

\[
X_{id} = X_{id} + V_{id}
\]  

6. Reduce the Temperature value using the following equation

\[
T = T \times 0.95
\]  

7. Repeat Steps 3-5 until the number of iteration is greater than the allowable maximum iteration number \( T_{\text{max}} \) or till the optimum query plan achieved.

V. EXPERIMENTAL ANALYSIS

The dataset used in this proposed research is the Lehigh University benchmark (LUBM) dataset to test the proposed algorithm. The LUBM benchmark is the most popular benchmark for semantic web repositories. It uses a straightforward ontology to portray the structure of a university organization with synthetically generated datasets. The dataset used in this research is LUBM (5,0) which consists of about 645,649 triples. The proposed algorithm is tested in a Microsoft Windows XP platform on an Intel Pentium 4 machine with 2GB RAM. Queries with OPTIONAL blocks are created and tested for performance of the algorithm. The number of predicates is varied and the proposed algorithm is iterated for 100 times. The execution time is recorded before and after application of the optimization algorithm.

The average execution time obtained for five different queries with varying number of predicates are recorded and compared with the query execution time before optimization. Figure 5.1 shows the average execution time in seconds.

![Figure 5.1 Average Execution time](image)

The experimental results shows that the query execution time gets reduced compared to querying without optimization. When the number of predicates increases also the query execution time gets reduced with the proposed algorithm.

VI. CONCLUSION

In this work, the application of SAPSO algorithm to optimize SPARQL queries with OPTIONAL blocks is presented. The algorithm starts with a solution space consisting of all possible query plans. The fitness function for the optimization algorithm is defined which depends upon the cardinality and selectivity of the triples in the input dataset. The experimental results show the efficiency of the applied algorithm in terms of query execution time. The SAPSO algorithm is applied to queries with OPTIONAL blocks having different number of predicates and the best plan is generated which in turn reduces the execution time of the query.

REFERENCES


