Acta Scientific COMPUTER SCIENCES

Volume 4 Issue 1 January 2022

Review Article

TaQO: A Tabu Search Based SPARQL Query Optimization Approach

Tanvi Chawla*

Research Scholar, MNIT, Jaipur, India

***Corresponding Author:** Tanvi Chawla, Research Scholar, MNIT, Jaipur, India.

Received: August 04, 2021

Published: December 13, 2021

Abstract

Semantic Web is an emerging technology for information representation in web pages. This growth has furtheraccelerated with the Linked Open Data (LOD) movement. One ofthe commonly accepted standard for representing semantic web data is the Resource Description Framework (RDF). SPARQL Protocol and RDF Query Language (SPARQL) is the commonly used query language for querying data from the Semantic Web. Query Processing is one of the most important tasks ofany database and thus it requires optimal solutions. Query Optimization is one of the phases in query processing. This phase is crucial for generating an optimed version to a submittedquery. This optimized query will reduce the query executiontime depending upon the type of optimization solution used. The generally used solutions to Query optimizations like those used for relational databases can be directly applied to Semantic web frameworks. But these solutions have to be tailored according to RDF data and SPARQL.

Keywords: Semantic Web; RDF; SPARQL; Query Optimization; Selectivity

Introduction

Semantic Web represents linked data; this data can be easily accessed and understood. The main objective of Semantic Web is to make all information that is available on the World Wide Web (WWW) accessible and it should be understood byboth the humans and the machines. The Semantic Web has the potential to change the way how data on the web is collected, stored and analyzed. The potential of the Semantic Web isyet to be fully exploited. RDF is the standard model for data interchange on the Semantic Web and has also become popularbecause of its flexibility that is a result of its underlying graph based model [15].

Query optimization for distributed SPARQL processing isnecessary as Semantic web data is available over distributed sources. Moreover, with the proliferation in the amount of Semantic web data storage of RDF data on a single machine has become difficult. The centralized RDF storage systemsare not adequate to store and process this vast amount ofdata. Thus, come into picture the distributed RDF systems which can adequately store and process this large amount of RDF data. But these systems are limited by the large amount of processing and communication costs involved while processing SPARQL queries over this distributed RDF data. The objective of query optimization is to find a query execution plan that minimizes the processing cost of the query and the cost of data transmission for query execution in a distributed environment. The query execution plans can be structurally represented as left-deep, right-deep and bushy trees [12].

Join order is an important factor that affects the performance of query processing, because join ordering affects the size of intermediate results. A good query execution plan thus depends on the order of joins. The overall goal is findan optimal order for the patterns in the query that minimizethe size of intermediate results during each stage of query execution. For scalable manage-

ment of semantic web data it makes sense to spend time on query optimization for large datasets. As it will help to produce optimal query plans which in turn will reduce the query processing time, thus showing a considerable improvement in query performance. The order ofjoins mainly depends on the selectivity & the cost of the join [1]. The SPARQL query may be visualized as a query graph. The selectivity in the graph is computed for both the nodesand the edges. The edge selectivity or the join selectivity is computed as the product of selectivity of the two join patterns.

One of the parameters frequently used while SPARQL query optimization is selectivity of triple patterns. The triple patterns may be ranked by the size of their intermediate results.The selectivity of a triple pattern, sel(T) is the fraction of the triples which satisfy that pattern. The statistical information may be used to get an estimate of the selectivity of a triple pattern. The triple patterns may be ranked according to their estimated selectivity that is expected to reduce the size of the intermediate results [2]. Using the cost function for estimationof selectivity, the triples can be ranked in the increasing order of their selectivity. This cost function estimates the selectivity in the range [0,1]. The overall cost of a triple pattern is;

$$c(t) = c(s) * c(p) * c(o) \text{---------- (1)}$$

Where t is the triple pattern, and s, p, o are the subject, predicate and object in the triple pattern.

The remainder of this paper is organized as follows. In Section 2, the some of the work done in context of cost-based and heuristic-based SPARQL optimization is discussed.Section 3, presents the a brief overview about the tabu search approach and its applications. Section 4, presents the proposed approach and the fitness function used for the Tabu Search. Finally in Section 6, we conclude by discussing the parameters used in our approach and future work that can be done by extending the proposed approach.

Related work

P. Tsialiamanis., *et al*. [3] proposed a Heuristics based SPARQL optimization approach. The authors analyzed the drawbacks of a cost based optimization approach considering the large join order space. The reason being that most of the times the statistics are missing. The importance of a heuristic based optimization approach was emphasized even withan option to partially use cost-based statistics. The authors proposed Heuristic SPARQL planner (HSP) that is based on computing a query execution plan without using nay costmodel. The SPARQL query optimization problem is reduced toa maximum independent set problem. During SPARQL query optimization it is difficult to estimate the statistics due to the complex nature of RDF data. Also using cost based optimization means to have prior knowledge of the underlying dataset. This is not the case in case of Heuristics based optimizationas no prior knowledge of dataset of the dataset is required. SPARQL query optimization is targeted at minimizing theintermediate results size and determining an optimal query execution plan. To achieve optimization another solution isto determine an optimal join ordering and maximizing thenumber of merge joins as they have less cost as comparedto the Hash joins. The heuristic based optimisation techniques proposed by the authors can be used in both the centralizedas well as the distributed settings like in the cloud. The first heuristic is based on the selectivity of the triple patterns; and the second heuristic is based on the distinct position of the joins. There are three more heuristics that are concluded after studying the RDF graphs. The authors have compared their HSP with the cost-based dynamic programming (CDP) plannerof RDF-3x. the HSP is implemented on top of the MonetDB.

K. Anyanwu [4] presents a vision for translating SPARQL queries to another data model and algebra i.e. the Nested Triple Group Data Model and Algebra (NTGA). This model has been proposed for evaluating SPARQL queries over MapReduce. There are several challenges which need to be solved while implementing traditional optimization techniqueson the MapReduce platform. One of them being collectingdata statistics and building indexes around that data. The traditional cost based optimization techniques cant be directly implemented on MapReduce as they are not aware of theMapReduce setting. Other techniques like the Heuristics- based SPARQL optimization can be considered. One of the considerations while implementing this technique is reducing the length of MapReduce workflows. While implementing SPARQL optimization over MapReduce it is important toreduce the number of cycles and to process a query withina single MapReduce job. Also the communication overhead incurred while processing queries must be minimized.

B. Quilitz and U. Leser [5] proposed DARQ, an engine for federated SPARQL queries, this engine uses query rewriting and cost-based query optimization techniques to speed up the process of

query execution. The results obtained depict that the optimization techniques used help to achievea great improvement in query performance despite of the fact that only a small amount of statistics may be available. The authors have proposed a query optimization algorithm that builds a cost-effective query plan. In the second stage of DARQ i.e. the Query Planning stage, the query is decomposed into multiple sub-queries. In the third stage i.e. Optimization, the Query optimizer takes the sub-queries from the previous stage and builds an optimized query execution plan. Some of the statistical information that is available is the number of triples with predicate p, the selectivity of a triple pattern with predicate p is the subject is bound etc. The authors have used iterative dynamic programming for optimization. Among all the computed query execution plans the best one is computed by comparing all the plans using a cost model. The cost factor that is used for the sub-queries is the expected result size ofthe queries. The cost of the nested loop and the bind joins are computed individually, and the result size of a query will be estimated based on the statistics. This optimization algorithm is implemented over distributed sources.

P. Obermeier and L. Nixon [6] proposed a cost model that is a sub-component of a query optimizer for distributed SPARQL processing. This cost model will serve as a cost indicator for the other components of the query optimizerlike for query rewriting, decomposition as well for choosing a join algorithm that decides on the join order. One of themost popular SPARQL query engine is Jena ARQ. This cost model estimates the cost for each individual operation in the query and then computes a total of these costs. These costs are distinguished on the number of resources used like CPU instructions, the number of I/O operations etc. It is presumed that the SPARQL query is given as a graph model i.e. a SPARQL query graph model (SQGM). This query graph model is a planar rooted directed graph in which the operators are the nodes and the data flow is represented as edges. The proposed cost model is based on a recursive cost function that in turn relies on a recursive cardinality function. Firstly,a method is used to assign cost and cardinality functions to each basic SPARQL operation. Cardinality, CPU and I/O cost functions are assigned to each SPARQL operation. This SQGM is evaluated in a distributed environment.

A. Hogenboom., *et al*. [7] introduced RDF chain query optimization for a distributed environment. Some of the meta- heuristic nature inspired methods like genetic algorithm (GA),ant-colony optimization (ACO) etc. have been used for query optimization but they have only been evaluated in a centralizedenvironment. The authors have proposed a technique to adapt these techniques in a distributed environment and assess the performance of these methods in this environment. The objective is to optimize the join order of the results of the SPARQL chain queries. The authors have also analyzed the effects of the different join methods e.g. nested-loop join, bindjoin etc. while querying RDF data the triples are matched against the patterns in the SPARQL queries. The join orderof the results obtained from the subqueries may be optimized and later joined to compute the final results of the query. The sequence of joins in the query can be represented as trees

i.e. left-deep, right-deep or bushy trees. These nodes in the query tree can be ordered in multiple ways and are referredto as the query paths. The total execution time of the queryis dependent on the ordering of these joins of the subqueries. The solution space for the process of SPARQL query path optimization consists of multiple right-deep and bushy trees. The number of possible solutions for right deep trees is n! while for bushy trees it is

$$\frac{2(n-1)}{(n-1)}(n-1)! \qquad (2)$$

The challenge in query path optimization is to find an optimal query path that minimizes the query executioncosts [15,16]. The authors propose a cost model for executingSPARQL queries in a distributed environment. For multiple join operations cardinality is estimated, cardinality is estimatedby counting the number of triples that match a pattern. The cardinality of a non-base join set pj in distributed databases is estimated by using bloom filters or histograms. The different optimization algorithms 2PO (RCQ-2PO), GA (RCQ-GA) [10], and ACO (RCQ-ACO) [11] are implemented on multipledistributed sources and the results are compared based on the execution time for the optimization process and the quality of the optimized solution. From the results obtained it can be concluded that irrespective of the cost function used, RCQ- ACO algorithm performs faster for smaller queries i.e. that consist of upto 10 joins, whereas the RCQ-GA algorithm isthe fastest for larger queries.

A major amount of work has been done in employing these nature inspired metaheuristic algorithms for SPARQL optimization. Apart from the Genetic (GA) and Ant colony optimization (ACO) algorithms one another such algorithm i.e. an Cuckoo search algorithm has also been used for SPARQL optimization where the

best query plan is generated using the fitness function [13]. In another such work, optimal query plans are generated using SAPSO algorithm which is a hybrid of the Simulated Annealing (SA) and Particle Swarm Optimization (PSO) algorithms [14].

Fuqi Song and Olivier Corby [8] proposed query plan- ning methods for SPARQL query optimization, since query planning is one of the most essential tasks performed by a query optimizer. Query Planning (QP) component enumerates all the possible query plans and selects the best one. Thetwo main issues involved in SPARQL query planning are;the representation of SPARQL statements and evaluating the cost of queries. To solve the first issue a SPARQL querygraph may be constructed for the statements. And to solvethe second issue the most frequently used approaches include pre-computed statistics and heuristics. The first approach i.e. pre-computed statistics certain summary data is obtained on the available RDF data. This summary data may be obtained using histogram-based methods and then this data is utilizedto calculate the cost of the query plans. With the heuristic- based methods certain heuristics are defined by observing the RDF data, and then these heuristics are applied to estimate thecost. The statistic-based approaches are expensive in terms of resources like time and space that is used but relatively they aremore accurate. On the contrary, the heuristic-based approaches are easy to implement, are less costly but they may be less efficient for some particular datasets. Also, heuristic-basedapproaches are more suitable to be applied in a distributed environment as in such an environment it is difficult to obtain statistics. The authors have used heuristic-based approachesas these approaches have shown to depict promising results for SPARQL query planning. The heuristic-based methods areused with a cost model to formalize the heuristics and to calculate the cost of the query plan. Two of the main things essential in a query planning process requires include; computed query plans and a cost estimation model or technique.A desirable query optimizer must generate query plans havinglow cost and apply accurate cost estimation techniques. The authors have used the Extended SPARQL query triple pattern Graph (ESG) for representing their SPARQL statements. The cost can be computed in terms of the execution time andspace consumed by the results after evaluating a SPARQL expression in a query. Both the time and space can be evaluatedby computing the intermediate results returned from the RDF dataset. This means if the intermediate result size is more then,the cost of evaluating the query in terms of execution time and storage space is also more. SPARQL optimization intends at minimizing this intermediate result size to reduce the query processing time.

Wang., *et al*. [9] presented a Minimum-Spanning-Tree-based (MST-based) algorithmic approach for distributed SPARQL processing. Some previous research has shown that methods from the graph theory can be applied to optimizethe performance of distributed SPARQL. The early approachesfor SPARQL optimization include query rewriting, selectivity- based triple pattern reordering etc. These optimization techniques for SPARQL queries are still not much developed as they lack the required efficiency. One of the methods for con- structing query plans for distributed SPARQL queries includessearching the minimum weight path using a graph traversing algorithm. Using this approach a query execution plan (QEP) can be constructed for distributed SPARQL queries. Using some graph traversal algorithm an optimal query plan may be constructed that corresponds to a MST. The authors have used the Prims algorithm to find the MST. The evaluation frame- work designed by the authors consists of 5 components i.e. a data generator, data splitter, statistics collector, data distributorand a test driver. One of the many drawbacks in distributed SPARQL optimization is lack of an efficient and accuratecost model. The authors have proposed and implemented this DSP engine that is based on the Prims algorithm. They have compared this DSP engine with the DARQ engine, from the results it is concluded that this DSP engine performs better than the DARQ in terms of both the query processing timeand system resource usage.

RDF-3X is considered to be the most popular tool for scalable management of Semantic web data; the query times are quite fast for this engine. To enable high performance SPARQL queries on the semantic web RDF-3X makes use of histograms, summary statistics, and some query optimization techniques. Thus, RDF-3X can generally outperform the other solution for queries. But the performance of RDF-3Xs degrades if the selectivity factor of queries is low even for the queries with simple joins. The performance of RDF-3X is poorfor such queries and in most of the cases it aborts before the completion of the query. RDF-3X achieves best performance only for queries that have a high selectivity and contain boundobjects. RDF-3x utilizes the dynamic programming approach to enumerate all the possible query plans. The plans may be computed in bottom-up or top-down manner. The timecomplexity

of the approach is $O(n^3)$, where n is the numberof triple patterns in the query [1].

Tabu search

Tabu search is a metaheuristic local search algorithmthat has been used to solve various optimization problems. Tabu search uses a neighborhood search procedure. In this way, it iteratively moves from one possible solution x to another possible solution x´ which is in the neighborhood of X This process continues until a stopping condition is metor some threshold is reached. The tabu search will explorethe neighborhood of each solution carefully with progressing search. In this algorithm, a memory structure is used knownas the tabu list (T_L). This memory structure is used to record the search history. So a new improved solution is determined using this memory structure as the search progresses. The tabulist generally consists of a short term list of solutions that havebeen visited in the past. The memory structures used in this tabu search can be categorized into short-term, intermediate and long-term. Like the short-term structure consists of a listof solutions that were recently considered.

Algorithm 1 Pseudo code for Tabu Search

```
1: Begin
2: t ← 0
3: initialize tabu search;
4: while (t < t_max) do
5: t t+1;←
6:     Search the neighborhood;
7:     Evaluate candidate solutions;
8:     Update tabu list;
9: End
```

In tabu search strategic restrictions and aspiration levels are introduced as a means for exploiting the search spaces.

This flexible memory structure T_L restricts the next solution choice to some subset of neighborhood of the current solution. Some of the parameters in tabu search are the neighborhood structure, aspiration conditions, the maximum size of the tabu list and the stopping rule. In tabu search a neighborhood is constructed to identify the adjacent solutions that can be reached from the current solution. Some of the main draw- backs of tabu search include that many parameters have to be determined and the iterations can be very large. Tabu searchis widely used in many applications such as Graph coloring and partitioning, Vehicle routing, Job shop scheduling etc.

Proposed approach

SPARQL query plan

The RDF data of the Semantic web can be queried using SPARQL. This SPARQL query can be represented as a query tree where the leaf nodes represent the triples and the internal nodes represent the join between these triples. These query trees can be represented as bushy trees, left-deep and right- deep trees. The nodes in these trees can be arranged in different ways to produce the same results. The order in which teh operation sin the tree are executed is referred to as the query plan. In a parallel setting or a distributed environment the bushy trees are used frequently. Our proposed approach uses the left deep trees.

Solution space

Each solution in the solution space represents a query plan. The size of this solution space depends on the type of query trees that are being used. As we are using here the left deep trees so there are n! possible solutions. These different solutions can be obtained by applying various transformation rules. The type of encoding used for these left-deep trees isthe ordered list. A query plan tree (T1∞ (T2∞ T3∞) T4) is encoded as "1234".

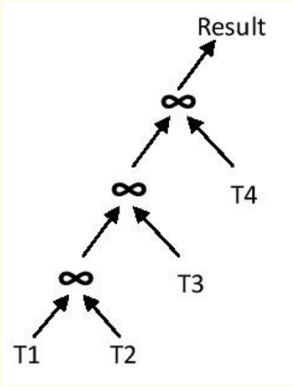

Figure 1: Left-deep trees.

Fitness function

The fitness function or the objective or cost functions refers to the cost of a query plan. The cost of the query plan is

the cost of thus left deep tree that depends on the selectivity and the cardinality estimation. This fitness function is defined assuming that the population contains n solutions. This fitness function represents the cost of the solution which is to be minimized. So among all the possible solutions the solutionwith the minimum fitness value will be chosen. Co_i represents the cost of a solution i, C_i represents the cardinality and $s_{i,j}$ is the selectivity.

$$s_{i,j} = min \quad \frac{|C_i|}{|C_i X C_j|}, \frac{|C_j|}{|C_i X Cj|}$$

The cardinality for a join tree T can be computed as,

$$C_T = C_i \text{ if T is a leaf}$$

$$C_T = |C(T1)||C(T2)| \text{ if T=T1} \infty \text{T2}$$

For a given tree T with multiple joins, the cost function Co_T can be defined as;

$$Co_T = 0 \text{ if T is a leaf}$$

$$Co_T = |C_T| + |Co(T1)| + |Co(T2)| \text{ if T=T1} \infty \text{T2}$$

Algorithm 2 Pseudo code for TaQO

1: Randomly generate population of N query plans.
2: Evaluate the cost of each plan based on cardinality as given in cost model.
3: Sort the query plans based on cardinality as given in the cost model.
4: Select the query plan q_p with minimum fitness value (query cost represented by fitness function and has to be minimized).
5: Set $q_{best}=q_p$.
6: Initialize an empty tabu list, T_i.
7: Find subset of $N(q_p)$, the neighbors of q_p that are not in the tabu list T_i.
8: Find the nearest query plan q_p' of the current selected query plan q_p. This can be found by calculating the minimum difference between fitness functions of the two query plans.

$$\Delta f = min[Co(q_p) - Co(q_p')]$$

9: The neighboring query plan q_p' in subset of N with minimum Δf is set as the best query plan i.e. $q_{best}=q_p'$.
10: In each iteration find the nearest unvisited node from current node until all nodes are visited.
11: Put q_p and q_p' in tabu list T_i. This tabu list will prevent their occurence for a limited period of time.
12: Continue this process until the maximum number of iterations are reached or a stopping condition is satisfied.

Conclusion

In the past work has been done using various nature inspired algorithms like Cuckoo search and Ant colony optimization (ACO) for SPARQL optimization. In this work, we have presented another nature inspired optimization approach; TaQo for the problem of SPARQL query optimization. In the presented algorithm the cardinality of the triples in the dataset is estimated using the given cost model. The query plans differ from each other according to the order of their joins. The maximum number of iterations can be manually set and the stopping condition be the minimum value of **Δ**f. In the future work, this algorithm can be extended to develop a hybrid algorithm that is a combination of two metaheuristic algorithms. The hybrid algorithm can be a combination of cuckoo search and tabu search algorithms for solving the problem of SPARQL query optimization. This algorithm can also be adapted to work in a distributed environment.

Bibliography

1. P Yuan., *et al.* "Dynamic and fast processing of queries on large-scale RDF data". *Knowledge and Information Systems* 41.2 (2014): 311-334.
2. Bernstein C., *et al.* "OptARQ: A SPARQL Optimization Approach based on Triple Pattern Selectivity Estimation". Department of University of Zurich (2007).
3. P Tsialiamanis., *et al.* "Heuristics-based query optimisation for SPARQL". in the 15th International Conference on Extending Database Technology, Berlin, Germany (2012): 324-335.
4. K Anyanwu. "A vision for SPARQL multi-query optimization on MapReduce".in The 29th International Conference on Data Engineering Workshops (ICDEW) Brisbane, Australia (2013): 25-26.
5. B Quilitz and U Leser. "Querying distributed RDF data sources with SPARQL". in European Semantic Web Conference, Tenerife, Spain (2008): 524-538.
6. LNP Obermeier and L Nixon. "A cost model for querying distributed rdf repositories with sparql". in The Workshop on Advancing Reasoning on the Web: Scalability and Commonsense, Tenerife, Spain (2008).
7. Hogenboom E., *et al.* "RDF chain query optimization in a distributed environment". in the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain (2015): 353-359.

8. F Song and O Corby. "Extended Query Pattern Graph and Heuristics- based SPARQL Query Planning". *Procedia Computer Science* 60 (2015): 302-311.

9. X Wang., *et al.* "Evaluating graph traversal algorithms for distributed SPARQL query optimization". in Joint International Semantic Technology Conference, Hangzhou, China (2011): 210-225.

10. Hogenboom V., *et al.* "RCQ-GA: RDF chain query optimization using genetic algorithms". in International Conference on Electronic Commerce and Web Technologies, Linz, Austria (2009): 181-192.

11. EG Kalayci., *et al.* "An ant colony optimisation approach for optimising SPARQL queries by reordering triple patterns". *Information Systems* 50 (2015): 51-68.

12. O Gorlitz and S Staab. "Federated data management and query optimization for linked open data". *New Directions in Web Data Management* 1 (2011): 109-137.

13. R Gomathi and D Sharmila. "A novel adaptive cuckoo search for optimal query plan generation". *The Scientific World Journal* (2014).

14. R Gomathi and D Sharmila "A Hybrid Nature Inspired Algorithm for Generating Optimal Query Plan". World Academy of Science, Engineering and Technology. *International Journal of Computer, Electrical, Automation, Control, and Information Engineering* 8.8 (2014): 1519-1524.

15. T Chawla., *et al.* "Research issues in RDF management systems". in International Conference on Emerging Trends in Communication Technologies (ETCT), Dehradun, India (2016): 1-5.

16. T Chawla., *et al.* "A shortest path approach to SPARQL chain query optimisation". in International Conference on Advances in Computing, Communications, and Informatics (ICACCI), Udupi, India (2017): 1778-1783.

17. T Chawla., *et al.* "JOTR: Join-Optimistic Triple Reordering Approach for SPARQL Query Optimization on Big RDF Data". in 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bengaluru, India (2018): 1-7.