International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:8, No:8, 2014

A Hybrid Nature Inspired Algorithm for Generating Optimal Query Plan

R. Gomathi, D. Sharmila

***Abstract*—**The emergence of the Semantic Web technology increases day by day due to the rapid growth of multiple web pages. Many standard formats are available to store the semantic web data. The most popular format is the Resource Description Framework (RDF). Querying large RDF graphs becomes a tedious procedure with a vast increase in the amount of data. The problem of query optimization becomes an issue 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 the proposed SAPSO algorithm which is a hybrid of Simulated Annealing (SA) and Particle Swarm Optimization (PSO) algorithms. The proposed SAPSO algorithm has the ability to find the local optimistic result and it avoids the problem of local minimum. Experiments were performed on different datasets by changing the number of predicates and the amount of data. The proposed algorithm gives improved results compared to existing algorithms in terms of query execution time.

***Keywords*—**Semantic web, RDF, Query optimization, Nature inspired algorithms, PSO, SA.

I. Introduction

THE information provided by the web pages can be easily understood by people but not by machines. The semantic web provides a way for machines to interpret the information in the web pages. By using the semantic web technology, machines can easily perform searching, grouping and manipulating on information in the web. The data in semantic web can be represented using many formats like the RDF, Web Ontology Language (OWL), and Extensible Mark-up Language (XML). The most popular of them is the RDF format.

RDF is a data model to represent information in the web. It uses XML syntax and uses Uniform Resource Identifiers (URI) to identify resources. Each resource can be described using property and property values in RDF. A language for querying data represented using RDF is the SPARQL protocol and RDF Query Language (SPARQL). This query language can be used to query data from different 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 several hundred million years, and it has been providing many imaginative solutions to solve many problems. Nature-inspired algorithms often use many interacting agents. The sources of nature can be based on swarm intelligence, biological systems, physics based or chemistry based systems. A subset of metaheuristcs is often referred to as Swarm Intelligence (SI) based algorithms. The 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 behaviour of birds and fish [10], the Firefly Algorithm (FA) based on the blinking pattern of tropical fireflies [13], and Cuckoo Search (CS) algorithm [14] inspired by the brooding parasitism of some cuckoo species. Physics and chemistry based algorithms include Harmony Search, Simulated Annealing [2] and so on. Nature inspired algorithms can be applied for solving large scale problems. Hybrids of these algorithms are also possible to produce best results.

The paper is organised as follows: Section II reviews the methods used for query optimization in literature; Section III makes a study of the PSO algorithm and SA algorithm; Section IV explains the proposed SAPSO algorithm; Section V describes the datasets used and the experimental results; Section VI sketches the conclusions and future works.

R. Gomathi is with the Bannari Amman Institute of Technology, Erode, Tamil Nadu, India (e-mail:gomsbk@gmail.com).

II. Review of Related Works

The elementary concepts associated with efficient processing [1] of SPARQL queries was studied in literature. The study was performed on i) the complexity analysis of all operators in SPARQL query language ii) equivalences of SPARQL algebra iii) algorithm for optimizing semantic SPARQL queries. The complexity analysis shows that all fragments of SPARQL fall into the category of NP.

To handle the large amount of RDF data, there is a need to optimize the join of the partial query results. An Ant Colony System (ACS) [3] was proposed in literature to query effectively in semantic web environment. The improvement in solution costs was compared with the existing algorithms like Genetic Algorithm (GA) and two phase optimization (2PO). It was proved that the ACS approach outperforms the existing approaches.

To optimize a special class of SPARQL queries called RDFchain queries, a new genetic algorithm was devised called the RCQ-GA[4].The algorithm optimizes the chain queries by finding the order in which joins are to be performed. The performance of the algorithm was compared with two phase optimization and the results show the quality of the solution and consistency of the solution.

A new join ordering algorithm based on cardinality estimation [5] was proposed in literature for SPARQL query

optimization. Experiments were conducted on star queries and arbitrary queries and the results show that optimal query plans were found and executed in less time.

A new set of PSO algorithms to optimize queries in distributed databases [6] was presented in research. The particles representing solutions are moved with respect to a probability distribution. The algorithm searches for the near optimal quality execution plans. By varying the parameter settings of PSO the execution times can be adjusted.

A parallel join algorithm was designed to handle RDF [7] and its query language SPARQL.The algorithm is a combination of three join algorithms and it processes multiple queries in an interleaved fashion. The performance results were discussed for different data sizes.

A language for querying graphs was proposed [8] called the G-SPARQL. A hybrid query engine was proposed in literature which splits the query plan and parts of them are pushed inside the relational database and some parts are processed using memory based algorithms. The efficiency and scalability of the proposed approach was discussed.

A RDF query engine was designed to evaluate SPARQL queries [11]. The evaluation mechanism uses an index structure to index the RDF triples. A tree shaped optimization algorithm was developed to convert the SPARQL query graph into an optimal query plan. Experiments were conducted on both real and synthetic datasets to measure the performance.

To reduce the query responding time, a cost model was developed [12] using Map Reduce framework to explore the scalability of RDF data. The search space is reduced by using a new algorithm called All-Possible-Join tree (APJ-tree) algorithm. A hybrid joins and a bloom filter was applied to speed up the processing of joins. Tests were conducted and the results were compared with the state of art solutions.

III. Existing Approaches

A. Particle Swarm Optimization Algorithm

The PSO [10] algorithm simulates the behaviours of bird flocking. Suppose the following scenario: a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know where the food is. The effective strategy to find the food is to follow the bird which is nearest to the food.

PSO learned from the scenario and used it to solve the optimization problems. In PSO, each single solution is a "bird" in the search space. We call it "particle". All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest. When a particle takes part of the population as its topological neighbours, the best value is a local best called lbest.

After finding the two best values, the particle updates its velocity and positions with following (1) and (2):

$$v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[]) \quad (1)$$

$$present[] = present[] + v[] \quad (2)$$

v [] is the particle velocity, present [] is the current particle (solution).
pbest [] and gbest [] are defined as stated before.
rand () is a random number between (0, 1).
c1, c2 are learning factors, usually c1 = c2 = 2.

The pseudo code of the procedure is as follows

```
For each particle
   Initialize particle
END

Do
   For each particle
      Calculate fitness value
      If the fitness value is better than the best fitness value (pBest) in
history
         set current value as the new pBest
   End

   Choose the particle with the best fitness value of all the particles
as the gBest
   For each particle
      Calculate particle velocity according equation (1)
      Update particle position according equation (2)
   End
While maximum iterations or minimum error criteria is not attained
```

Particles' velocities on each dimension are clamped to a maximum velocity Vmax. If the sum of accelerations would cause the velocity on that dimension to exceed Vmax, which is a parameter specified by the user,then the velocity on that dimension is limited to Vmax.

B. Simulated Annealing Algorithm

Simulated annealing is a kind of global optimization technique based [2] on annealing of metal. It can find the global minimum using stochastic searching technology from the means of probability. Simulated annealing algorithm has a strong ability to find the local optimistic result and it can avoid the problem of local minimum.

SA is a relatively straightforward algorithm which includes Metropolis Monte Carlo method. The Metropolis Monte Carlo algorithm is well suited for this simulation since only energetically feasible states will be sampled at any given temperature. The Simulated Annealing algorithm is therefore a Metropolis Monte Carlo simulation that starts at a high temperature. The temperature is slowly reduced so that the search space becomes smaller for the Metropolis simulation,

and when the temperature is low enough the system will hopefully have settled into the most favourable state.

Simulated Annealing can also be used to search for the optimum solution of the problems by properly determining the initial (high) and final (low) effective temperatures which are used in place of kT in the acceptance checking, and deciding what constitutes a Monte Carlo step.

The initial and final effective temperatures for a given problem can be determined from the acceptance probability. In general, if the initial Monte Carlo simulation allows an energy increase of dE_i with a probability of P_i, the initial effective temperature is $kT_i = -dE_i/\ln(P_i)$.

A similar expression can be used to determine the final effective temperature. It is important to realize that Simulated Annealing is strictly a minimization procedure. To determine the initial effective temperature the acceptance probability of a given increase in the cost needs to be chosen.

The below algorithm describes the process of Simulated Annealing:

Start with the system in a known configuration, at known energy E
T=temperature =hot; frozen=false;
While (! frozen) {
repeat {
Perturb system slightly (e.g., moves a particle)
Compute E, change in energy due to perturbation
If(ΔE < 0)
Then accept this perturbation, this is the new system config
Else accept maybe, with probability = exp (-ΔE/KT)
} until (the system is in thermal equilibrium at this T)
If(ΔE still decreasing over the last few temperatures)
Then T=0.9T//cool the temperature; do more perturbations
Else frozen=true
}
return (final configuration as low-energy solution)

IV. Proposed Approach

A. RDF Query Plans

The semantic web data represented by RDF format can be queried using the SPARQL query language. To generate the query plan we should convert the given SPARQL query into a query tree. Each SPARQL query can be depicted as a tree structure. The leaf nodes of the query tree represent any of the triples and the intermediate nodes are used to join the triples. Different forms of tree structures like left deep trees, right deep trees, bushy trees are available. In this research, left deep trees [9] are used.

A left deep tree is defined as a join tree with join operators as the inner nodes and relations as the leaf nodes. In the context of SPARQL query a left deep tree consists of any of the triples in the leaf nodes with join operators in the intermediate nodes. A sample left deep tree is given in the Fig. 1.

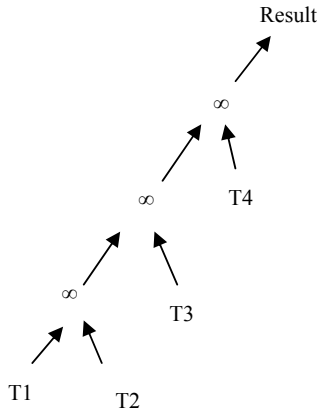

Fig. 1 A sample left deep tree

For example consider the sample SPARQL query,

```
Query1:
SELECT ?Person
WHERE {
? Person foaf: name ?name.
? Person foaf: workplaceHomepage ?Work.
? Person foaf: Publications ?Pubs.
}
```

Fig. 2 shows the corresponding left deep tree for the sample query Query1.

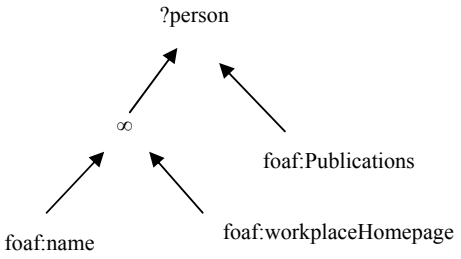

Fig. 2 A left deep tree for Query1

B. Solution Space

The solution space consists of many particles. In this research, the particles in the solution space can be represented as query execution plans. The number of particles in the solution space depends on the type of tree used to represent the query plan. Since we use a left deep tree, there are n! different query plans [9] can be generated for a query tree with n leaf nodes. The n! solutions can be obtained by applying the transformation rules like join commutativity, join associativity, left join exchange and right join exchange.

C. Encoding of Particles

In order to apply any optimization algorithm to solve a problem, a suitable encoding scheme must be chosen for the particles in the solution space. Two types of encoding are available for left deep trees [9]: i) ordered list and ii) ordinal number encoding. In this research, we choose ordered list for

encoding query plans. Solutions are represented as an ordered list [1] of leaves. For example, the query plan tree (((T1∞T2) ∞T3) ∞T4) is encoded as "1234".

The sample query query1 given in the previous section consists of three predicates. So there are 3! =6 different ways in which we can represent the query tree which gives the same result.

In Fig. 2, consider
foaf:name as 1
foaf: workplaceHomepage as 2
foaf: Publications as 3

The possible encoding is as follows:
123
132
213
231
312
321

D. Fitness Function

To determine optimal query plan, let us decide on the fitness function. The fitness function for the problem of query optimization refers to the cost of the left deep tree. The cost of a left deep tree depends on the selectivity and cardinality estimation. Cardinality of a triple pattern is the number of triples that match a particular pattern. Selectivity of a join between two triples T1 and T2 is defined as the number of triples satisfying both T1 and T2.Let R_i be the cardinality and $f_{i,j}$ be the selectivity. If $p_{i,j}$ is the join predicate between R_i and R_j, we can define selectivity as in (3):

$$f_{i,j} = \frac{|R_i \infty_{i,j} R_j|}{|R_i \times R_j|} \quad (3)$$

For a given join tree T, the resultant cardinality |T| can be recursively computed as in (4) and (5):

$$|T| = |R_i| \quad \text{if T is a leaf } R_i \quad (4)$$

$$|T| = \left(\prod_{R_i \in T1, R_j \in T2}, f_{i,j} \right) |T1||T2| \quad \text{if T=T1} \infty \text{ T2} \quad (5)$$

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

$$C_{out}(T) = 0 \quad \text{if T is a leaf } R_i \quad (6)$$

$$C_{out}(T) = |T| + C_{out}(T1) + C_{out}(T2) \quad \text{if T=T1} \infty \text{ T2} \quad (7)$$

E. Implementation of the SAPSO Algorithm

The following algorithm 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 $\boldsymbol{X}_i$ in the search space is randomly placed. Its velocity vector is $\boldsymbol{V}_i = (v_{i1}, \ldots, v_{id} \ldots v_{iD})$, in which the velocity in the d^{th} dimension is V_{id}=rand()×$\boldsymbol{V}_{max}$, where rand() is the random number in the range [-1, 1].
2. Assign value for c1, c2, 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

$$V_{id} = V_{id} + c_1 \times rand() \times (P_{id} - X_{id}) + c_2 \times rand() \times (P_{id} - X_{id}) \quad (8)$$

5. Update the location of the particle and SA operator

$$X_{id} = X_{id} + V_{id} \quad (9)$$

6. Reduce the Temperature value using the following equation

$$T = T * 0.95 \quad (10)$$

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.

V. Experimental Analysis

A. Data Sets

Three different datasets, namely Leigh University benchmark (LUBM) dataset, Friend of a Friend (FOAF) dataset and the Central Intelligence Agency (CIA) World Fact book dataset are used in this research to test the proposed algorithm.

The LUBM benchmark is the most popular benchmark for semantic repositories. It uses a simple ontology to describe the structure of a university organization with synthetically generated datasets. It consists of a set of 14 test queries, and several performance metrics. The dataset we use here is LUBM5 which consists of about 645,649 triples.

The FOAF consists of millions of RDF documents which is used to describe attributes of people and relationships between them. The FOAF has become one of the most commonly used semantic web ontologies. It consists of about 201,612 RDF triples.

The CIA World Fact book contains data about 250 countries defined using RDF statements. Information about government, people, geography, economy, transportation and many more are provided by this dataset. It consists of more than 100,000 RDF statements.

The proposed algorithm is tested in a Microsoft Windows XP platform on an Intel Pentium 4 machine with 2GB RAM. From each of the three datasets, about 100,000 triples are taken and tested. The number of predicates is varied to

measure the performance of the algorithm. The algorithm is iterated for 100 times and the results obtained are recorded. The proposed SAPSO algorithm is compared with GA and PSO.

The average execution times obtained for three datasets for varying number of predicates is recorded.

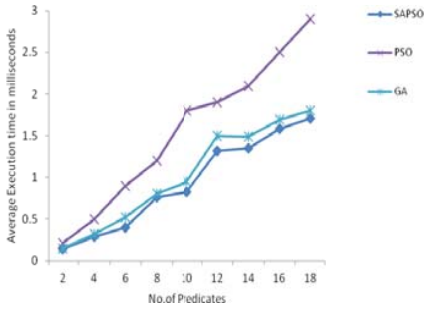

Fig. 3 Average Execution times (LUBM dataset)

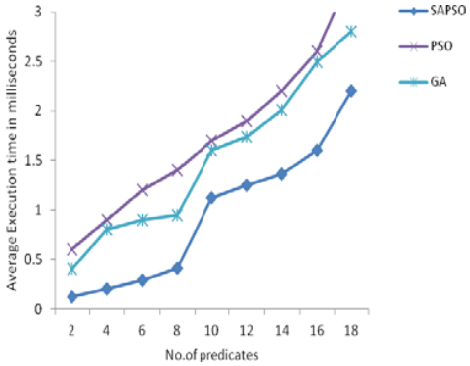

Fig. 4 Average Execution times (FOAF dataset)

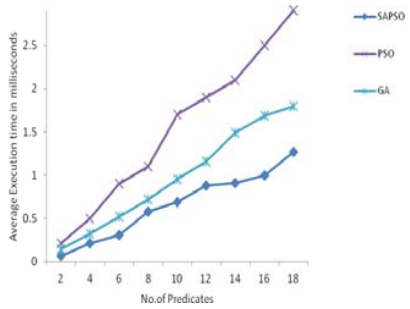

Fig. 5 Average Execution times (CIA World Factbook dataset)

Figs. 3-5 show the average execution times of the proposed SAPSO algorithm compared with PSO and GA. Execution time linearly increases when the number of predicates is increased. Comparing the three methods, SAPSO algorithm gives better execution time.

Sample Queries:

LUBM Query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ub: <http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
SELECT? X ?Y ?Z
WHERE
{? X rdf:type ub:GraduateStudent .
 ? Y rdf:type ub:University .
 ? Z rdf:type ub:Department .
 ? X ub:memberOf ?Z .
 ? Z ub:subOrganizationOf ?Y .
 ? X ub:undergraduateDegreeFrom ?Y}
```

FOAF Query:

```
SELECT? Person
WHERE {
? Person foaf: name? name.
? Person foaf:workplaceHomepage ? Work.
? Person foaf:Publications ?Pubs.
}
```

CIA World Factbook Query:

```
PREFIX c: <http://www.daml.org/2001/09/countries/fips#>
PREFIX o: <http:www.daml/org/2003/09/factbook/factbook-ont#>
SELECT? Partner
WHERE {
c: SouthAfrica o: importPartner ?imppartner.
?impPartner o: Country? Partner.
? Partner o: border? border.
?border o: Country?neighbour.
?neighbour o: internationalDispute? dispute.
}
```

VI. Conclusion

In this work, we have presented the SAPSO algorithm which is a hybrid of PSO and SA for solving the problem of query optimization. The algorithm starts with a search space consisting of all possible query plans. The fitness function for the algorithm is the cost of the query plan which is calculated based on the cardinality of the triples occurring in the dataset. The cost of the different query plans depends on the cardinality and selectivity of triples.

The experimental results show the effectiveness of the algorithm in terms of query execution time. The SAPSO algorithm has been applied to datasets of varying sizes and the best query plan is found based on the fitness function. The query execution time is recorded for three different datasets of varying sizes. The SAPSO algorithm outperforms when compared to GA and PSO. To improve the accuracy of the results, other hybrid nature inspired algorithms can be applied and performance can be measured.

References

[1] Michael Schmidt, Michael Meier, Georg Lausen," Foundations of SPARQL Query Optimization", Proceedings of the 13th International Conference on Database Theory, pp.4-33, 2010.

[2] Kirkpatrick, S, Gelatt Jr, C. D, Vecchi, M. P, "Optimization by Simulated Annealing". Science vol.220 No.4598, pp. 671–680, 1983.

[3] Alexander Hogenboom, Ewout Niewenhuijse, Frederik Hogenboom, and Flavius Frasincar," RCQ-ACS: RDF Chain Query Optimization Using an Ant Colony System", IEEE/WIC/ACM International Conferences On Web Intelligence and Intelligent Agent Technology, vol.1, pp.74-81, 2012

[4] Alexander Hogenboom, Viorel Milea, Flavius Frasincar, and Uzay Kaymak," RCQ-GA: RDF Chain Query Optimization using Genetic Algorithms", Lecture notes in Computer Science, vol.5692, pp.181-192, 2009.

[5] Andrey Gubichev, Thomas Neumann," Exploiting the query structure for efficient join ordering in SPARQL queries", Proceedings of the 17th international conference on Extending Database Technology, 2014.
[6] Tansel Dokeroglu, Umut Tosun, Ahmet Cosar,"Particle Swarm Intelligence as a New Heuristic for the Optimization of Distributed Database Queries", 6th International Conference on Application of Information and Communication Technologies (AICT), pp.1-7, 2012.
[7] Juerg Senn," Parallel Join Processing on Graphics Processors for the Resource Description Framework", 23rd International Conference on Architecture of Computing Systems (ARCS), pp.1-8, 2010
[8] Sherif sakr, Sameh Elnikety, Yuxiong He, "Hybrid query execution engine for large attributed graphs", Journal of Information Systems, vol.41, pp.45–73, 2014.
[9] Michael Steinbrunn, Guido Moerkotte, Alfons Kemper,"Heuristic and randomized optimization for the join ordering problem", The VLDB Journal, vol.6, pp.191-208, 1997.
[10] Kennedy, J.; Eberhart, R."Particle Swarm Optimization". Proceedings of IEEE International Conference on Neural Networks vol.IV. pp. 1942–1948, 1995.
[11] C.Liu, H.Wang,"Towards efficient SPARQL query processing on RDF data", Tsinghua science and technology, vol.15, No.6, pp.613-622, 2010.
[12] X. Zhang, L. Chen, and M. Wang, "Towards efficient join processing over large RDF graph using map reduce," in Scientific and Statistical Database Management,vol.7338 of Lecture Notes in Computer Science, pp. 250–259, 2012.
[13] Xin-She Yang, Nature-Inspired Metaheuristic Algorithms. Frome: Luniver Press. ISBN 1-905986-10-6, 2008.
[14] X.-S. Yang; S. Deb, "Cuckoo search via Lévy flights", World Congress on Nature & Biologically Inspired Computing (NaBIC 2009). IEEE Publications. pp. 210–214, 2009.