Information Systems 50 (2015) 51–68

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

An ant colony optimisation approach for optimising SPARQL queries by reordering triple patterns

Elem Guzel Kalayci [a], Tahir Emre Kalayci [b,*], Derya Birant [c]

[a] Dokuz Eylul University The Graduate School of Natural and Applied Sciences, 35390 Izmir, Turkey
[b] Celal Bayar University Computer Engineering Department, 45140 Manisa, Turkey
[c] Dokuz Eylul University Computer Engineering Department, 35390 Izmir, Turkey

ARTICLE INFO

Article history:
Received 8 February 2013
Received in revised form
23 December 2014
Accepted 29 January 2015
Recommended by: J. Van den Bussche
Available online 7 February 2015

Keywords:
SPARQL
Query optimisation
Reordering triple patterns
Ant colony optimisation
Ant system
MAX–MIN ant system

ABSTRACT

Processing the excessive volumes of information on the Web is an important issue. The Semantic Web paradigm has been proposed as the solution. However, this approach generates several challenges, such as query processing and optimisation. This paper proposes a novel approach for optimising SPARQL queries with different graph shapes. This new method reorders the triple patterns using Ant Colony Optimisation (ACO) algorithms. Reordering the triple patterns is a way of decreasing the execution times of the SPARQL queries. The proposed approach is focused on in-memory models of RDF data, and it optimises the SPARQL queries by means of Ant System, Elitist Ant System and MAX–MIN Ant System algorithms. The approach is implemented in the Apache Jena ARQ query engine, which is used for the experimentation, and the new method is compared with Normal Execution, Jena Reorder Algorithms, and the Stocker et al. Algorithms. All of the experiments are performed using the LUBM dataset for various shapes of queries, such as chain, star, cyclic, and chain–star. The first contribution is the real-time optimisation of SPARQL query triple pattern orders using ACO algorithms, and the second contribution is the concrete implementation for the ARQ query engine, which is a component of the widely used Semantic Web framework Apache Jena. The experiments demonstrate that the proposed method reduces the execution time of the queries significantly.

1. Introduction

Today, the Web is the primary environment for information sharing and exchange. As a consequence, the information on the Web grows substantially every day. This increase entails an important problem: processing the excessive amounts of information. The Semantic Web is a solution that is offered by the scientists and developers for this important problem. It is a paradigm for understanding and responding to user requests and understanding, interpreting and deducing Web content by computers [1]. It is also an effort for providing a common framework (based on a Resource Description Framework (RDF)) that allows the sharing and reuse of data across applications, enterprises and community boundaries [2]. Despite the popularity and extensive usage of the Semantic Web, there are some challenges when addressing issues such as ontology matching, ontology aligning, query optimisation and reasoning, and performance improvement. The current literature proposes various solutions for these challenges.

Query optimisation is a mature and comprehensive research area. Various deterministic and probabilistic techniques have been applied to the query optimisation problem in

* Corresponding author. Celal Bayar Universitesi Bilgisayar Muhendisligi Bolumu Muradiye Kampusu, 45140 Manisa, Turkey. Tel.: +90 2362012105.
E-mail address: tahir.kalayci@cbu.edu.tr (T.E. Kalayci).

http://dx.doi.org/10.1016/j.is.2015.01.013
0306-4379/© 2015 Elsevier Ltd. All rights reserved.

various domains (e.g., Relational DBMS [3–5], Object-Oriented DBMS [6,7], and XML [8,9]). Although there have been remarkable studies about query optimisation in RDF databases [5,10–17], the query optimisation area has not reached a sufficient maturity level in the RDF domain yet.

Ant Colony Optimisation (ACO) is a metaheuristic optimisation technique that is inspired from real ant colonies and their methods of finding food. Artificial ants in a colony cooperate to find good solutions to difficult discrete optimisation problems [18]. Hence, cooperation is a key component of the ACO algorithms [18]. ACO is also accepted as a paradigm for designing metaheuristic algorithms to solve combinatorial optimisation problems [18]. In ACO, computational resources are allocated to a set of simple agents (artificial ants), which communicate indirectly by stigmergy [18]. This communication is mediated by the environment with the help of pheromone intensity [18]. The main underlying idea of ACO is that of "a parallel search over several constructive computational threads based on local problem data and on a dynamic memory structure containing information on the quality of previously obtained result", and this idea is "loosely inspired by the behaviour of real ants" [19]. The interaction of different search threads creates a collective behaviour that has been proven to be effective in solving combinatorial optimisation problems [19].

In contrast to previous studies, this study introduces the use of ACO metaheuristic for the query optimisation problem of SPARQL (SPARQL Protocol And RDF Query Language) *SELECT* queries that have different graph shapes. In the proposed approach, the ACO metaheuristic implementations Ant System (AS), Elitist Ant System (EAS) and MAX–MIN Ant System (MMAS) are used for the optimisation of the queries.

The contributions of this study are the following:

- Real-time optimisation of Basic Graph Patterns (i.e., optimising the order of triple patterns) of SPARQL *SELECT* queries that have different shapes by using ACO algorithms.
- Concrete implementation for the Apache Jena ARQ query engine to optimise in-memory RDF data.

This paper is organised as follows: Section 2 introduces important concepts about SPARQL query optimisation. Section 3 presents previous studies about query optimisation. Section 4 explains the steps of the proposed approach and the ACO algorithm implementations. Section 5 presents the performed experiments and a discussion. Last, the paper concludes with outcomes of the study in Section 6.

2. SPARQL query optimisation concepts

Query optimisation is a critical issue for DBMSs, and it has been addressed for decades. Although there is a substantial amount of knowledge in the literature about query optimisation, it is still a recent research area for RDF databases.

Query optimisation can be defined as searching for an optimal query execution strategy. For example, for a single query, there could be different query execution plans with different execution times that produce the same results sets. The performance difference between query execution plans of the same query can be enormous. Therefore, finding optimal or nearly optimal execution plans is a very important task for query engines.

Resource Description Framework (RDF) is a language [2] with directed, labelled graph data [20] for representing information about resources on the World Wide Web. Although it is intended to represent meta-data about Web resources, it is also used to represent information about things that are identified on the Web [21]. SPARQL is a query language for RDF, and it is used to express queries across diverse data sources [20]. Jena[1] is a Semantic Web framework that is used to manipulate and store RDF graphs in memory or in persistent storage. ARQ[2] is a query engine of Jena that is used for querying ontologies by SPARQL. Internally, ARQ uses iterators extensively, and the evaluation of an operation is achieved by feeding the stream of results from the previous stage into the evaluation.[3] ARQ uses a strategy that is similar to an index-join, which is a type of nested-loop join,[4] that proceeds by substituting the results from the expressions already generated into the next triple pattern to be solved.[5] Apache Jena ARQ has been used for the implementation of the approach proposed in this paper.

There are many studies that address query optimisation by join ordering [3,5,22–24]. These studies draw attention to the effect of join ordering on the query execution time. For this reason, this study is focused on reordering triple patterns of SPARQL queries that correspond to the join ordering process for SQL queries. Determining the order of the triple patterns is a key factor in optimising joins [25]; thus, it is a key factor for decreasing the execution time of the queries. The purpose of reordering the triple patterns of the SPARQL queries is *to find the fastest (optimum) query execution plan* (the plan that returns the result set with the minimum execution time) compared to other execution plans.

SPARQL is based on graph pattern matching, and the *Basic Graph Pattern* (BGP) is the set of triple patterns [26]. BGP is constructed from *WHERE* clause predicates of the SPARQL query. A triple pattern (TP) is a structure that consists of three components (the subject, the predicate and the object), which could be concrete (i.e., bound) or variable (i.e., unbound) [10]. Sets of triple patterns (BGP) are fundamental to SPARQL queries because they specify access to the RDF data [10] and this approach constitutes "a little logic for extracting subsets of related nodes in an RDF graph" [27].

An example BGP (i.e., the *WHERE* clause of the corresponding SPARQL query) is given in Table 1. This example BGP is presented to explain the importance of the triple

[1] http://jena.apache.org/
[2] http://jena.apache.org/documentation/query/
[3] http://jena.apache.org/documentation/query/arq-query-eval.html
[4] http://informatik.hu-berlin.de/forschung/gebiete/wbi/teaching/archive/sose05/dbs2/slides/09_joins.pdf
[5] http://markmail.org/message/w32y7glxgxxa4u4e, http://answers.semanticweb.com/questions/2481/sparql-basic-graph-pattern-matching/2495