Expert Systems with Applications 40 (2013) 1555–1563

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

Ant colony optimization for RDF chain queries for decision support

Alexander Hogenboom [a,*], Flavius Frasincar [a], Uzay Kaymak [b]

[a] Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands
[b] Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

ARTICLE INFO

Keywords:
RDF chain query optimization
Ant colony optimization
Genetic algorithm
Iterative improvement
Simulated annealing

ABSTRACT

Semantic Web technologies can be utilized in expert systems for decision support, allowing a user to explore in the decision making process numerous interconnected sources of data, commonly represented by means of the Resource Description Framework (RDF). In order to disclose the ever-growing amount of widely distributed RDF data to demanding users in real-time environments, fast RDF query engines are of paramount importance. A crucial task of such engines is to optimize the order in which partial results of a query are joined. Several soft computing techniques have already been proposed to address this problem, i.e., two-phase optimization (2PO) and a genetic algorithm (GA). We propose an alternative approach – an ant colony optimization (ACO) algorithm, which may be more suitable for a Semantic Web environment. Experimental results with respect to the optimization of RDF chain queries on a large RDF data source demonstrate that our approach outperforms both 2PO and a GA in terms of execution time and solution quality for queries consisting of up to 15 joins. For larger queries, both ACO and a GA may be preferable over 2PO, subject to a trade-off between execution time and solution quality. The GA yields relatively good solutions in a comparably short time frame, whereas ACO needs more time to converge to high-quality solutions.

1. Introduction

In today's information-driven society, decision makers need to process a continuous flow of data through various input channels, by extracting information, understanding its meaning, and acquiring knowledge by applying reasoning to the gathered information (Hogenboom, Hogenboom, Frasincar, Schouten, & van der Meer, 2012). Yet, an overwhelming amount of data is available at any given time, whereas decision makers – businesses and consumers alike – need a complete overview of their environment in order to enable effective, well-informed decision making. In order to address this issue, Semantic Web (Berners-Lee, Hendler, & Lassila, 2001) technologies can be utilized in expert systems for decision support, as has been demonstrated in recent research on knowledge management (Joo & Lee, 2009), annotation (Aksac, Ozturk, & Dogdu, 2012) and recommendation of data sources (Hsu, 2009), intelligent search (Batzios, Dimou, Symeonidis, & Mitkas, 2008; Lupiani-Ruiz et al., 2011; Vandic, van Dam, & Frasincar, 2012), and personalized user experiences in, e.g., tourism (Garcia-Crespo, Lopez-Cuadrado, Colomo-Palacios, Gonzalez-Carrasco, & Ruiz-Mezcua, 2011) or e-commerce (Blanco-Fernandez et al., 2010).

The rise of the Semantic Web facilitates an ever-growing amount of data to be stored in many heterogenous, yet interconnected sources. This data is commonly represented by means of the Resource Description Framework (RDF), a framework for describing and interchanging meta-data (Klyne & Carroll, 2004), which describes the context of data and thus enables machine-interpretability. Due to the interconnectivity of data rather than pages, the Semantic Web has the potential of addressing today's typical users' complex information needs in a more effective and efficient way than the current Web can.

Semantic Web technologies allow a user to explore many different data sources in order to address very specific information needs. A typical scenario here may be a query for features and reviews of a holiday destination and comparable alternatives, as well as prices and details of trips to any of these locations. Such queries can be executed on multiple RDF sources by means of the SPARQL Protocol and RDF Query Language (SPARQL) (Prud'hommeaux & Seaborne, 2008). In order for SPARQL queries to disclose the ever-growing amount of widely distributed RDF data to demanding users in real-time environments, fast RDF query engines are crucial.

Recent expert systems for querying distributed environments with multiple heterogenous information sources such as databases or repositories focus on the problem of identifying the information sources that are most relevant with respect to queries (Jung, 2010) or on the problem of combining the query results from multiple

* Corresponding author. Tel.: +31 (0) 10 408 2264; fax: +31 (0) 10 408 9031.
E-mail addresses: hogenboom@ese.eur.nl (A. Hogenboom), frasincar@ese.eur.nl (F. Frasincar), u.kaymak@ieee.org (U. Kaymak).

sources while optimizing coverage and search effectiveness (Amin & Emrouznejad, 2011; Batzios & Mitkas, 2012). However, to the best of our knowledge, expert systems dealing with optimizing the execution efficiency of a given query in a distributed environment have been relatively unexplored.

One of the problems today's RDF query engines face is the optimization of the order in which the distinct parts of a query are executed. The total execution times of queries depend on these query paths. A good algorithm for optimizing the execution order in a query path can thus contribute to efficient querying. As the number of possible query paths grows exponentially with the query size, the optimization of RDF query paths is far from trivial. Therefore, several soft computing techniques have been proposed to address this problem. For instance, Stuckenschmidt, Vdovjak, Broekstra, and Houben (2005) present a two-phase optimization (2PO) algorithm, consisting of an iterative improvement (II) method followed by simulated annealing (SA). More recently, a genetic algorithm (GA) has been shown to be a promising alternative when optimizing RDF queries (Hogenboom, Milea, Frasincar, & Kaymak, 2009).

As their design has been inspired by methods for optimization of query paths in traditional databases, existing soft computing approaches to RDF query path optimization have essentially been designed for more or less static environments – changes in the environment typically require the optimization to be run all over again. However, the Semantic Web is a complex and dynamic environment. Data changes, sources come and go, and latency between sources may be volatile. In this light, ant colony optimization (ACO) (Dorigo, Maniezzo, & Colorni, 1996, 2006) is a good alternative to the existing soft computing approaches in an RDF environment. ACO is a soft computing technique inspired by the foraging behavior of ant colonies. Its nature allows the algorithm to be run continuously and to adapt to changes in the environment in real time. Moreover, ACO has been shown to outperform GAs in solving complex problems such as scheduling (Merkle, Middendorf, & Schmeck, 2002) and sequential ordering (Gambardella & Dorigo, 2000).

As its characteristics render ACO an attractive alternative to existing soft computing techniques for RDF query optimization, we explore the applicability of ACO to query path optimization in an RDF environment. Furthermore, we compare the performance of existing soft computing techniques for RDF query path optimization with our novel ACO algorithm. We focus on the performance of the considered algorithms when optimizing a special class of SPARQL queries, the so-called RDF chain queries, on a single source.

The remainder of this paper is structured as follows. First, Section 2 provides a short introduction to RDF and chain queries. We then discuss the two existing soft computing techniques as well as our novel ACO algorithm for RDF chain query optimization in Sections 3 and 4. In Section 5, we evaluate the performance of our considered methods in terms of execution time and solution quality. Finally, we draw conclusions and propose directions for future work in Section 6.

2. RDF and chain queries

In an RDF model, facts are declared as a collection of triples, each consisting of a subject, a predicate, and an object. RDF triples can be visualized in a graph, which can be described as a node and directed-arc diagram, in which each triple is represented as a node-arc-node link (Klyne & Carroll, 2004). The relationship between a subject node and an object node in an RDF graph is defined using an arc which denotes a predicate.

When querying RDF sources, RDF triples are essentially matched with a series of patterns specified in a SPARQL query. In the specific subset of SPARQL queries we consider in our current endeavors, the `WHERE` statement only contains a set of node-arc-node patterns which are chained together such that the object of one predicate is the subject of the next predicate. Such RDF chain queries resemble chain queries in traditional relational databases, where a query path is followed by performing joins between its subpaths of length 1 (Stuckenschmidt et al., 2005). Randomized and genetic algorithms have proven to outperform heuristic methods in optimizing these types of queries in relational databases (Steinbrunn, Moerkotte, & Kemper, 1997).

A typical example of a chain query in an RDF environment is the following. Let us consider an RDF model of the CIA World Factbook generated by using QMap (Hogenboom et al., 2008). This model contains data about 250 countries, defined in over 100, 000 triples. Suppose a financial risk analyst wants to identify the export partners of The Netherlands that have dependent areas (i.e., areas that do not possess full political independence or sovereignty) that are involved in an international conflict. This query can be expressed in SPARQL as demonstrated in Fig. 1.

This query can be subdivided into four subqueries: a query for information on the export partners of The Netherlands (line 4), a query for countries associated with other countries as export partners (line 5), a query for dependent areas (line 6), and a query for international disputes (line 7). In order to resolve the complete query, the results of the individual subqueries can be joined in any order. Here, the number of statements resulting from a join is equal to the number of statements compliant with both operands' constraints.

A sequence of joins in such a query can be visualized as a tree. The leaf nodes of an RDF query tree typically represent inputs, whereas the internal nodes represent algebra operations, enabling one to specify basic retrieval requests on the inputs (Frasincar, Houben, Vdovjak, & Barna, 2004). We consider the leaf nodes of a query tree to be matches with the individual patterns of triples constituting an RDF chain query. The internal nodes represent join operations.

The nodes in a query tree can be ordered in many different ways, which are referred to as query paths. In an RDF context, bushy and right-deep query trees can be considered (Stuckenschmidt et al., 2005). In bushy trees, base relations (containing matches with one triple pattern) as well as results of earlier joins can be joined. Right-deep trees, which are a subset of bushy trees, require the

```
1. PREFIX c: <http://www.daml.org/2001/09/countries/fips#>
2. PREFIX o: <http://www.daml.org/2003/09/factbook/factbook-ont#>
3. SELECT ?partner
4. WHERE { c:NL o:exportPartner ?expPartner .
5.         ?expPartner o:country ?partner .
6.         ?partner o:dependentArea ?area .
7.         ?area o:internationalDispute ?conflict .
8.       }
```

Fig. 1. Example RDF chain query in SPARQL.