Future Generation Computer Systems 74 (2017) 63–75

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Querying massive graph data: A compress and search approach

Chemseddine Nabti, Hamida Seba [*]

Université de Lyon, CNRS, Université Lyon 1, LIRIS, UMR5205, 69622, Villeurbanne, France

HIGHLIGHTS

- A new subgraph isomorphism search algorithm is proposed for massive graph data.
- Simplifying graph querying known to be NP-complete especially for massive data.
- Querying compressed graph data without decompression.
- Deal with compressed graphs that are smaller and simpler than the original ones.
- Graphs are compressed by modular decomposition.

ARTICLE INFO

Article history:
Received 16 September 2016
Received in revised form
29 March 2017
Accepted 1 April 2017
Available online 13 April 2017

Keywords:
Subgraph isomorphism
Graph queries
Massive graph databases
Graph compression
Modular decomposition

ABSTRACT

Querying graph data is a fundamental problem that witnesses an increasing interest especially for massive graph databases which come as a promising alternative to relational databases for big data modeling. In this paper, we study the problem of subgraph isomorphism search which consists to enumerate the embedding of a query graph in a data graph. The most known solutions of this NP-complete problem are backtracking-based and result in a high computational cost when we deal with massive graph databases. We address this problem and its challenges via graph compression with modular decomposition. In our approach, subgraph isomorphism search is performed on compressed graphs without decompressing them yielding substantial reduction of the search space and consequently a significant saving in processing time as well as in storage space for the graphs. We evaluated our algorithms on nine real-word datasets. The experimental results show that our approach is efficient and scalable.

1. Introduction

In the article, "Is Graph Theory the Key to Understanding Big Data?" [1], the authors show how graph theory can be precious as a tool to organize and understand massive data. Flexible and open structures, graphs are natural and practical solutions for at least two of the Vs of massive data: Velocity for its streaming or dynamic nature and Variety for its different forms (heterogeneity) [2,3].

Graphs are a well-known mathematical concept that proved its flexibility and efficiency as a modeling tool in various domains and applications [4]. A graph is a collection of vertices connected by edges. Both vertices and edges can piggyback information called labels or attributes. Appreciated by their flexibility and their graphical representation, graphs have been used to formalize problems since the work of *Euler* on the problem of the *Seven Bridges of Königsberg* [5]. Representing data with graphs dates back to 1969 with the work of Bachman on the *Network* database model [6] but graphs have been overshadowed by the powerful relational modeling paradigm. Today, graphs come back and are projected to the frontend as a data modeling tool with several propositions of graph-based database management systems [7] and graph query languages [8]. This can be widely explained by the emergence of services and applications that rely on structured data such as social networks, crime detection and genoms. But, it is also related to the issues and challenges raised by massive data and for which graphs seem to be a viable solution.

Using graphs as means of representing data cannot be successful without developing effective ways to search and query this kind of data. However, querying massive graph data is a challenging issue. In fact, the main task involved in querying graph data is subgraph isomorphism search which is an NP-complete problem [9]. The subgraph isomorphism problem is the problem of finding the embedding of a query graph into a target graph called a data graph. The data graph is generally larger than the query and may contain several occurrences of it. The objective is then to

* Corresponding author.
E-mail address: hamida.seba@univ-lyon1.fr (H. Seba).

http://dx.doi.org/10.1016/j.future.2017.04.005
0167-739X/© 2017 Elsevier B.V. All rights reserved.

enumerate all the occurrences of the query graph within the data graph in a reasonable time. Fig. 1 depicts an example where the data graph contains two occurrences of the query graph. Given a query graph Q and a data graph G, a straightforward solution to enumerating the occurrences of Q into G is to directly compare the vertices of the query with the vertices of the data graph. This comparison constructs a search tree where each vertex of the tree corresponds to a mapping between a vertex from the query with a vertex from the data graph. Fig. 2 presents a part of the obtained recursion tree for the query graph and the data graph of Fig. 1. Exploring this recursing tree is the Achilles heel of subgraph isomorphism search. So, existing backtracking based solutions do not explore the whole tree, but use filtering methods that prune unpromising branches of the tree. Nevertheless, even with pruning functions, this method rises two main challenges:

- It is memory consuming: besides storing the data graph which can have a significant size, exploring the search tree has a high memory consumption and involves complex data structures to support backtracking.
- It is time consuming: backtracking and testing the possible mappings between vertices has a high computational cost. It is exponential in function of the number of vertices in the involved graphs.

In this paper, we target the above challenges and propose a new framework that handles efficiently the problem of querying graph data with subgraph isomorphism search. Our solution aims to deal with subgraph isomorphism challenges in massive graph databases through graph compression. Of course, compression in itself is not a new idea. It is omnipresent in our digital world. In fact, compression is used almost every where especially in data transmission. The goal is usually to reduce the space occupied by data even if this means losing some information when decompressing the data. Compression algorithms are inextricably linked to decompression ones because we need to decompress data to be able to use it. However, the era of massive data is changing usages. The goal of compression has evolved. It goes without saying that we always want to compact data but we also want to be able to use the data in its compact format, i.e., without decompression. For graph data, this means less space for storing the graphs but also acceptable time processing of the compressed graphs. This work investigates this issue for the problem of subgraph isomorphism search. More precisely, we make the following contributions:

1. *A simple compressed representation of both the data and the query graphs*. We tackle the memory and the computational challenges by compressing the graphs. Compression not only reduces the amount of memory needed to store large graph data but also the time needed to process them because they are simpler and smaller. In fact, our compression reduces the number of vertices and edges of the graphs yielding a reduction of the search tree proportionally to the compression rate.
2. *An intuitive subgraph isomorphism enumerating algorithm that works on the compressed graphs without decompressing them.*
3. *Performance studies on nine real datasets.* We conduct extensive performance studies using nine real-world graphs with various graph properties. The experimental results demonstrate that our proposed algorithms can handle large graphs with significant performance on both computational time and storage space.

A preliminary version of the current paper appeared in [10]. The current paper has been significantly extended with respect to the underlying methodology and the experimental evaluation based on IoTBD 2016 selection.

2. Background

2.1. Preliminaries

We consider data graphs defined as simple vertex labeled graphs. Simple graphs are graphs with no edges involving a single vertex.

Definition 1. A data graph G is a 3-tuple $G = (V, E, \ell)$, where V is a set of vertices, $E \subseteq V \times V$ is a set of edges connecting the vertices, $\ell : V \rightarrow \Sigma$ is a function labeling the vertices where Σ is the sets of labels that can appear on the vertices.

In this paper, the notation $G = (V, E)$, with ℓ omitted means that we actually do not need the labels of the vertices but just their identifiers (i.e., indexes).

An undirected edge between vertices u and v is denoted indifferently by (u, v) or (v, u). For each $v \in V$, $d(v)$ denotes the degree of v, i.e., the number of neighbors of v, where a neighbor is a vertex adjacent to v. The label or set of labels of a vertex v is given by $\ell(v)$.

A graph that is contained in another graph is called a subgraph and can be defined as follows:

Definition 2. A graph $G_1 = (V_1, E_1, f_{V_1})$ is a subgraph of a graph $G_2 = (V_2, E_2, f_{V_2})$, if $V_1 \subseteq V_2, E_1 \subseteq E_2, f_{V_1}(x) = f_{V_2}(x) \forall x \in V_1$.

Graph isomorphism is defined as follows:

Definition 3. A graph $G_1 = (V_1, E_1, f_{V_1})$ and a graph $G_2 = (V_2, E_2, f_{V_2})$ are said to be isomorphic, if there exists a bijective function $h : V_1 \rightarrow V_2$ such that the following conditions hold:

1. $\forall x \in V_1 : f_{V_1}(x) = f_{V_2}(h(x))$
2. $\forall (x, y) \in E_1 : (h(x), h(y)) \in E_2$
3. $\forall (h(x), h(y)) \in E_2 : (x, y) \in E_1$.

Given a query graph Q and a data graph G, the subgraph isomorphism search of Q in G consists to find all the subgraphs of G that are isomorphic to Q.

2.2. Related work

In this section, we review the main algorithms proposed to deal with subgraph isomorphism search and point-out the mechanisms proposed to scale to large graphs. We also present the approaches that deal with graph compression.

2.2.1. Subgraph isomorphism search algorithms

Searching in graphs is an important problem studied in many domains related to classification, information retrieval, mining and pattern recognition. Subgraph isomorphism search that consists to find all the occurrences of a subgraph into a larger graph is a basic task in graph search. Subgraph isomorphism search is an NP-complete problem that has been extensively studied [11–13,8,14,15]. We will not describe all the existing solutions. We rather focus on the basic and the most recent ones. Detailed algorithms of the most known solutions are nicely surveyed and compared in several papers [16–18]. In our description, we mainly focus on how these algorithms deal with scalability, i.e., massive graphs.

Given a query graph Q and a data graph G, searching for Q in G is a complex task that involves a total or a partial traversal of a search tree (see Fig. 2 for an example). In the search tree, each vertex at level i maps a vertex of the query to a vertex of the data graph. Each path from the root to a leaf in the search-tree represents either a unsuccessful mapping between the query and a subgraph that have been dropped by the algorithm or a successful one that corresponds to a subgraph that is isomorphic to the query. Dealing