TBSGM: A Fast Subgraph Matching Method on Large Scale Graphs

Fusheng Jin, Beijing Institute of Technology, Beijing, China
Yifeng Yang, Beijing Institute of Technology, Beijing, China
Shuliang Wang, Beijing Institute of Technology, Beijing, China
Ye Xue, Northwestern University, Evanston, USA
Zhen Yan, Technical University of Munich, Munich, Germany

ABSTRACT

Subgraph matching, which belongs to NP-hard, faces significant challenges on a large scale graph with billions of nodes, and existing methods are usually confronted with greater challenges from both stability and efficiency. In this article, a subgraph matching method in a distributed system, tree model-based subgraph matching method (TBSGM) is proposed. The authors provide a transformed efficient query tree as a replacement for a query graph. In order to get the tree, they present a cost evaluation model which may help to generate the efficient query tree according to network communication-cost and calculation-cost evaluation. Also, a key set based indexing strategy for intermediate results is given to simplify the matching results during network communication. Extensive experiments with real-world datasets show that TBSGM significantly outperforms other methods in the aspects of scalability and efficiency.

KEYWORDS
Efficient Query Tree, Indexing Strategy, Large Scale Graph, Subgraph Matching

1. INTRODUCTION

Graph data have been increasing exponentially in recent years. In 2015, the number of active users on Facebook in a month has reached 1.3 billion (http://www.statisticbrain.com/facebook-statistics/). It is unfeasible to handle these large-scale graphs that have more than billion nodes on a single machine. That is, in the context of large scale graph, processing graph in distributed systems become an inevitable trend. Among all graph algorithms, subgraph matching has been widely used in many applications, such as finding isomorphism in organic polymer and protein matching (Zhao & Han, 2010), knowledge bases (Kasneci, Suchanek, Ifrim, Ramanath, & Weikum, 2008; Wu, Li, Wang, & Zhu, 2012) and program analysis (Zhang, Yang, & Jin, 2010; Zhao & Han, 2010). Many methods like (Cordella, Foggia, Sansone, & Vento, 2004; Han, Lee, & Lee, 2013; He & Singh, 2010; Shang,
Zhang, Lin, & Yu, 2008; Sun, Wang, Wang, Shao, & Li, 2012; Ullmann, 1976; Zhang, Li, & Yang, 2009; Zhao & Han, 2010) have been given to deal with the subgraph matching efficiently.

However, dealing with subgraph matching on large data graph is still an unresolved issue. When dealing with large scale graphs, all existing methods are either unstable or inefficiency. Matching order and redundant results are the most common reasons that result in instability and inefficiency (Han et al., 2013; Lee et al., 2012).

1.1. Subgraph Matching

In this paper, when handling subgraph matching problem of large scale graph with more than billion nodes in distributed systems, the subgraph matching problem is defined as follows: For a data graph G and a query graph Q, retrieve all subgraphs of G that are isomorphic to Q (Sun et al., 2012). Subgraph matching approaches are mainly categorized into two groups under different implementing environment, centralized approaches and distributed approaches:

1. **Centralized Approaches:** (Cordella, Foggia, Sansone, & Vento, 2004; Han, Lee, & Lee, 2013; He & Singh, 2010; Shang, Zhang, Lin, & Yu, 2008; Ullmann, 1976; Zhang, Li, & Yang, 2009; Zhao & Han, 2010). Given a matching order of query nodes, it compares each query node with the data node and check the edges that connect to this query node, until every query node has been matched. Centralized approaches are usually efficient when the scale of graph data is small. However, when it comes to large scale graph data, these approaches are very low efficiency, because the data graph cannot be stored in computer memory completely, and it takes a lot of extra resources, such as visiting external store or communicating on network.

2. **Distributed Approaches:** Sun Z (Sun et al., 2012) is a typical example. It transforms query graph to many STwigs and matching STwigs with data nodes. Then it performs the operation of Join to get the final result. However, the problem is that different matching orders and excessive intermediate results will significantly influence the efficiency.

1.2. Our Contribution

In this paper, a novel approach TBSGM is proposed to deal with subgraph matching. It focuses on the efficient query tree and the key set-based indexing strategy for intermediate results.

In order to ensure the stability in a distributed system, first TBSGM transforms query graph to query trees by splitting nodes. Then a cost evaluation model based on network communication cost and calculation cost evaluation is put forward to choose the best one which results in least communicational and computational costs from all the query trees. Tree model ensures query nodes order and performance stability (Han et al., 2013), and we don’t need extra iterations to check if some data edges exist or not in data graph, therefore it reduces the times of iteration and network communication cost.

By using efficient query tree, we can also filter redundant intermediate matching results as soon as possible. For example, a node \( u \) in query graph has three child query nodes \( (u_{c1}, u_{c2}, u_{c3}) \) which get their set of matching results respectively. With tree structure, we can merge these three sets to one and filter useless matching result of \( u_{c1}, u_{c2} \) and \( u_{c3} \) in the process of matching of query node \( u \), before we send them to \( u \)'s parent. As a result, we don’t need more network communication resources to process edges between these three sets in later procedures and eventually avoid the extra costs of network communications.

Furthermore, in order to reduce network communication cost. TBSGM uses a brief version of intermediate results (the key set-based indexing) to communicate instead of the complete set of results which have to be sent through network, therefore it reduces the costs of network communication greatly. However, using the brief results to communicate will cause information lost, so we add another phase to get the complete results in our approach. Compared with other methods that only
A Survey of Parallel and Distributed Data Warehouses
www.igi-global.com/article/survey-parallel-distributed-data-warehouses/1826?camid=4v1a

A Cached-Based Stream-Relation Join Operator for Semi-Stream Data Processing
www.igi-global.com/article/a-cached-based-stream-relation-join-operator-for-semi-stream-data-processing/168484?camid=4v1a