Big Data Research 10 (2017) 44–52

Contents lists available at ScienceDirect

Big Data Research

www.elsevier.com/locate/bdr

Community Detection Algorithm for Big Social Networks Using Hybrid Architecture

Rahil Sharma *,[1], Suely Oliveira [1]

University of Iowa, Dept. of Computer Science, Iowa city, IA-52246, USA

ARTICLE INFO

Article history:
Received 9 December 2016
Received in revised form 28 July 2017
Accepted 7 October 2017
Available online 26 October 2017

Keywords:
Community detection
Parallel distributed algorithms
Big data
Social networks

ABSTRACT

One of the most relevant and widely studied structural properties of networks is their community structure. Detecting communities is of great importance in social networks where systems are often represented as graphs. With the advent of web-based social networks like Twitter, Facebook and LinkedIn. community detection became even more difficult due to the massive network size, which can reach up to hundreds of millions of vertices and edges. This large graph structured data cannot be processed without using distributed algorithms due to memory constraints of one machine and also the need to achieve high performance. In this paper, we present a novel hybrid (shared + distributed memory) parallel algorithm to efficiently detect high quality communities in massive social networks. For our simulations, we use synthetic graphs ranging from 100K to 16M vertices to show the scalability and quality performance of our algorithm. We also use two massive real world networks: (a) section of Twitter-2010 network having $\approx 41M$ vertices and $\approx 1.4B$ edges (b) UK-2007 (.uk web domain) having $\approx 105M$ vertices and $\approx 3.3B$ edges. Simulation results on MPI setup with 8 compute nodes having 16 cores each show that, upto $\approx 6X$ speedup is achieved for synthetic graphs in detecting communities without compromising the quality of the results.

1. Introduction

One of the most relevant and widely studied structural properties of networks is their community structure. A community in a network is a set of nodes that are densely connected with each other and sparsely connected to the other nodes in the network. Community detection in a network extracts the structural properties of the network [1] and the various interactions in the network [2]. Detecting communities in social networks is of great importance because social networks consists of patterns which can be viewed as independent components, with each component having distinct features and can be detected based on network structure. Some major applications of community detection in social networks are a follows: (i) help to target users for marketing purposes, (ii) provide recommendations to users to connect with other users, join communities or forums, (iii) assist in market basket analysis to help group products likely to be sold together, (iv) help to generate user targeted advertisements.

The increasing size of social networks like Facebook, Twitter, LinkedIn, etc. has made community detection more difficult, with data size which can reach up to billions of vertices and edges. For example, Facebook has $\approx 1.1B$ users and LinkedIn has $\approx 500M$ users. As a result the ability to process this large graph-structured data in memory of a single machine is infeasible due to time and memory constraints. Most of the research in community detection has been focused on shared memory based algorithms on SMP machines and a thorough review of the same is presented in [3]. Where as some fast scalable community detection algorithms [4], [5], [6] which have been developed can only tackle network sizes which can be stored in the RAM of one machine. All of these algorithms adopt sequential, parallel shared-memory and non-distributed architectures. Processing networks with hundreds of millions of vertices and billions of edges require several hundred gigabytes of RAM. To address this challenge, parallel distributed community detection algorithms are necessary. *To avoid any confusion, we use the term cluster only for computer cluster; a part of the computer cluster will be denoted as machine or node, the objects in a network will be denoted as vertex and groups of vertices will be denoted as communities.*

In this paper, we modify and extend our multi-level multi-core (MCML), shared-memory based community detection algorithm [5] also explained in Section 3, to distributed memory parallel frame-

* Corresponding author.
E-mail addresses: rahil-sharma@uiowa.edu (R. Sharma), suely-oliveira@uiowa.edu (S. Oliveira).
[1] Equal contributor.

https://doi.org/10.1016/j.bdr.2017.10.003
2214-5796/

work using Message Passing Interface (MPI). This hybrid (shared + distributed memory) algorithm can process massive social networks to extract high quality communities efficiently. The main challenges we encountered were (1) the initial partitioning of the network and assigning each of these parts to different nodes in the parallel computers in such a way that, when community detection algorithm is applied on each individual node, it should not incur high communication overhead, (2) each node in the parallel computers should intelligently reduce the size of the network partition assigned to it such that, after merging, the entire network should fit in memory of one machine and quality of the communities detected is not compromised.

In this work, we integrate an existing network partitioning algorithm in our hybrid algorithm's flow so that, it will partition the original network into chunks to be distributed across the network of parallel machines, incurring minimum communication overhead between them. In order to minimize the probability of distributing vertices belonging to the same community across different machines, we use a network partitioning algorithm which tries to minimize the inter-partition edges [7]. After network partitioning and distribution, we intelligently reduce the size of every network partition on each machine in such a way that, when merging all the partitions back in the master node, the entire network can fit into the memory of a single master node to which we apply our MCML algorithm to extract high quality communities. All our simulations are done using MPI and OpenMP implementation on the HPC Neon cluster at the University of Iowa. The main contribution of this paper are as follows:

1. We develop a hybrid (shared + distributed memory) community detection algorithm, a modification and extension of our shared memory based MCML algorithm [5], which utilizes multiple cores of multiple machines and scales to hundreds of millions of vertices and edges without compromising quality of the detected communities.
2. We showcase our algorithms' efficiency by using synthetic graphs ranging from 100K up to 16M vertices and also on real world networks like (a) section of Twitter-2010 network having $\approx 41M$ vertices and $\approx 1.4B$ edges (b) UK-2007-05 (.uk web domain) having $\approx 1.2B$ vertices and $\approx 3.2B$ edges.

The structure of the remaining paper is as follows: In Section 2, we present an overview of related work in graph partitioning, community detection and parallel community detection. In Section 3, we describe the proposed core MCML [5] and our hybrid algorithm for large scale community detection. Following this, in Section 4, we discuss and present our experimental environment, datasets used and results, followed by our conclusion in Section 5.

2. Related work

Network partitioning: It aims to divide the network into k-parts in such a way that edge cuts are minimized and each partition roughly has same number of vertices. Most of the network partitioning problems are *NP-Hard* [3]. One group of techniques in graph partitioning relies on optimizing an objective function which is defined as a ratio of number of intra-partition edges to number of inter-partition edges. Another group of partitioning techniques uses multi-level partitioner [7], [8] whose implementation is in METIS and PMETIS library respectively. There exists other partitioning algorithms which scales better than METIS [9], [10] but incur very high communication overhead leading to large runtimes. We plan to utilize parallel METIS to perform our initial graph partitioning, due to its low communication overhead, ease of use and wide availability. The parallel implementation was done using GNU C++ and MPI.

Community Detection: This is an interesting problem in the domain of *graph partitioning*. Interest in community detection problem started with the new *partitioning* approach by [1], [11]; where the edges in the network with the maximum betweenness are removed iteratively, thus splitting the network hierarchically into communities. Similar algorithms were proposed later on, where attributes like 'local quantity' i.e. number of loops of a fixed length containing the given edge [12] and a complex notion of 'information centrality' [13], are used to decide removal of edges. *Hierarchical clustering* is another major technique used for community detection, where based on the similarity between the nodes, an agglomerative technique iteratively groups vertices into communities. There are different existing methods to choose the communities to be merged at each iteration. Algorithms described in [14] and [15] start with all the nodes as individual communities and iteratively merge them to optimize the 'modularity' function. Many other algorithms in the literature of community detection, like ones proposed by [16] and [17] rely heavily on *modularity maximization*. *Label propagation* is another well known technique used for community detection, which finds communities by iteratively spreading labels across the network. Raghavan et al. [6] proposed an algorithm, where each node picks the label in its 1-neighborhood that has the maximum frequency. These labels are permitted to spread synchronously and asynchronously across the network until near stability is attained in the network. This method has some limitations, where large communities dominate the smaller ones in the network, this phenomenon is called 'epidemic spread'. This limitation is tackled in [18]. Liu et al. [19] used affinity propagation, which is a similar approach to label propagation, for finding communities/clusters in images. Some community detection algorithms use *random walks* as a tool. The idea is that, due to the higher density of internal edges, the probability of a random walk staying inside the community is greater than going outside. This approach is used in Walktrap [20] and Infomap [21] algorithms. A thorough review on community detection algorithms for networks is given in [3]. A study presenting evolution and management of interest-based communities formed by humans is shown in [22]. Another interesting application of community detection is shown in [23], where due to the emergence of smart grids which enable bidirectional energy, finding economically motivated Prosumers-Community Groups (PCG) is important.

Parallel Community Detection: Community detection algorithms is a well studied research area, but achieving strong scalability along with detecting high quality communities is an open problem. Most of the past research on community detection has focused on single threaded algorithms. There is a rich and vast literature of such algorithms and the ones based on modularity maximization being the most prominent amongst them [11]. The *Louvain* method which is based on modularity maximization [4] is the most widely used community detection algorithm which can scale to networks with millions of vertices. However, the quality of results obtained deteriorates as the size of the network increases [24]. It is observed that modularity maximization based algorithms are unable to detect small and well-defined communities in large networks [25] [26]. One of the recent parallel algorithms developed to detect disjoint community structures based on maximizing weighted network partitioning is given in [20]. A scalable community detection algorithm, which partitions the network by maximizing the Weighted Community Clustering (WCC), is proposed in [27] which uses community detection metric based on triangle analysis [28]. Some other works which focused on developing parallel implementation for existing community detection heuristics is given in [29]. Recently, [30] proposed a scalable parallel algorithm for community detection, based on label propagation, which is optimized for GPGPU architectures. This algorithm just works on local information which drives the high scalability of this algorithm.