Journal of Visual Languages and Computing 28 (2015) 71–82

Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

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

View Points

GO: A cluster algorithm for graph visualization ☆

Xiaodi Huang [a,*], Weidong Huang [b]

[a] School of Computing and Mathematics, Charles Sturt University, Albury, NSW 2640, Australia
[b] School of Engineering and ICT, University of Tasmania, Newnham, Tasmania 7248, Australia

ARTICLE INFO

Article history:
Received 19 December 2013
Received in revised form
14 May 2014
Accepted 19 December 2014
Available online 30 December 2014

Keywords:
Information visualization
Graph clustering

ABSTRACT

As we are in the big data age, graph data such as user networks in Facebook and Flickr becomes large. How to reduce the visual complexity of a graph layout is a challenging problem. Clustering graphs is regarded as one of effective ways to address this problem. Most of current graph visualization systems, however, directly use existing clustering algorithms that are not originally developed for the visualization purpose. For graph visualization, a clustering algorithm should meet specific requirements such as the sufficient size of clusters, and automatic determination of the number of clusters. After identifying the requirements of clustering graphs for visualization, in this paper we present a new clustering algorithm that is particularly designed for visualization so as to reduce the visual complexity of a layout, together with a strategy for improving the scalability of our algorithm. Experiments have demonstrated that our proposed algorithm is capable of detecting clusters in a way that is required in graph visualization.

1. Introduction

Graphs have many applications, particularly in networks [28–30,35]. Graph data becomes large in a big data age. This fact creates the great challenge for visualizing large graphs. A number of techniques have been developed such as filtering [7] and distortion display [10,27]. Among them, clustering graphs serve as an efficient alternative method for reducing visual complexity. A clustered graph is produced from a large graph by two main steps. A clustering algorithm is first used to find highly-connected nodes and their incident edges in a graph that are grouped to form sub-graphs. Each of these sub-graphs is then replaced with a meta-node as a cluster. In this way, a coarse, clustered graph is obtained by replacing all sub-graphs with their corresponding meta-nodes. In order to reduce the visual complexity, however, the following requirements should be satisfied:

- Cluster sizes: identified clusters not only are densely connected sub-graphs, but also contain the sufficient number of nodes. Obviously, we are not interested in the small sub-graphs such as a sub-graph with three fully connected nodes. Replacing the small sub-graphs with their meta-nodes does not reduce visual complexity significantly.
- Overlapping: different clusters should not share too many nodes as their elements.
- The number of clusters: the number of clusters should be automatically detected. For a large graph, a user is not able to specify the accurate number of clusters as the input parameter of a clustering algorithm.
- Real-time: the clustering algorithm for visualization is normally used for user exploration of a graph. If the algorithm takes too long to find the resulting clusters, it will be impossible for such exploration.

Although a number of algorithms on clustering graphs [1,3,11,12,14,16] are available, they cannot completely satisfy

☆ This paper has been recommended for acceptance by Shi Kho Chang.

* Corresponding author.
E-mail addresses: xhuang@csu.edu.au (X. Huang), Tony.huang@utas.edu.au (W. Huang).

http://dx.doi.org/10.1016/j.jvlc.2014.12.007

the above requirements of visualizing graphs. Most of them focus on detecting densely connected sub-graphs while ignoring other requirements. This is because these algorithms are originally developed for vector data in other areas, rather than for graph data in graph visualization.

In graph mining, frequent sub-graphs cannot meet the requirements of graph visualization, either; graph partitioning [6,9] has traditionally attracted more interest. The graph partitioning methods divide a graph into a predefined number of partitions in an optimal way. In the context of graph visualization, it is, however, generally impossible to specify this number in advance.

Recently, a number of community detection algorithms are available. A community is equivalent to a cluster in a graph. However, most of these algorithms do not take into account the sizes of the identified clusters, while others focus on detecting the overlapping communities [15].

Current graph visualization systems make use of these existing clustering algorithms. It is challenging to develop a clustering algorithm that can meet all of the four requirements at the same time. Based on a community detection algorithm, the basic idea of our approach is to use an objective function derived from the well-known metric of modularity Q [13], and to provide a greedy solution to the function. According to this solution, we present an algorithm that clusters a graph for reducing the visual complexity of its layout. Since graph data becomes huge, it is still impossible to cluster a huge graph at once [5], even by using a clustering algorithm that meets the four requirements. The existing solution is to recursively cluster the graph [2,4,18,19]. A number of hierarchically clustered graphs are generated through a recursive clustering process. We call this approach as a vertical approach. Our solution uses a window approach, which is called the horizontal approach. A large graph is divided into a number of windows, and our algorithm is then applied to each window. In this way, users are able to explore the graph.

In this paper, we present a specific clustering algorithm for graph visualization by taking into account the above four requirements of visualization. The contributions of this paper are given below:

- A new algorithm for the clustering graphs, which is based on the structural patterns of graphs. The strategy for the scalabilities of the proposed algorithm is also given.
- An implementation of the algorithm and illustrative examples are provided.

The rest of this paper is organized as follows. In the following, we present related work. Section 3 presents the clustering algorithm for graph visualization, together with the strategy for improving the scalabilities of our algorithm. The experiment results are reported in Section 4, followed by the conclusion in Section 5.

2. Related work

In this section, we briefly review related work on cluster algorithms for visualization. Two fundamental questions on clustering graphs for visualization are as follows: one is how to detect clusters in a given graph, and another is about how to visualize the clustered graph. In this paper, we focus on the first question. For the second question, methods for drawing clustered graphs are presented by Eades et al. [2,11] and others [4,12].

In the area of graph visualization, clustering a graph refers to a process of grouping a set of nodes (or edges) in such a way that nodes (or edges) in the same cluster are more similar to each other than to those in other clusters. A node or an edge is of many characteristics. According to the types of characteristics that are used for measuring the similarities, we roughly classify approaches for graph clustering into two categories: geometric (topological features) abstraction and semantic (domain-specific attributes associated with nodes and edges) abstraction. Based on topological features such as proximities and linking patterns, geometric clustering can produce sub-graphs as social communities with distinctive structural features. Semantic clustering algorithms seek to retain the semantic attributes of nodes or edges, in order that the visualization is meaningful in itself. Nodes in a cluster have the common or similar attributes.

A number of geometric abstraction approaches have been reported in the literature. Using a distance measure between two clusters, van Ham et al. [17] cluster a graph in a bottom-up agglomerative way. The distance uses either the geometric distance or the edge length. Two clusters with the minimal distance is iteratively merged into one new cluster. The clustering quality function is a metric on the measure of the average link. Flow map layout [26] performs hierarchical, binary clustering on a set of nodes, positions, and flow data to route edges. Relying on a scalable technique for visualizing clustered hierarchies, ASK-GraphView [19] employs a series of increasingly sophisticated clustering algorithms in order to construct a hierarchy on a graph such as a tuned version of MCL [1]. The system provides facilities for on-demand navigation of cluster hierarchies for visualizing very large graphs. Other approaches utilize the geometric skeleton of a graph [7]. Introducing the distance metric, Botafogo et al. [8] construct a distance matrix that has as its entries the distances of every node to every other node, to identify hierarchies in an organization.

There are many semantic abstraction approaches. Di Giacomo et al. [21] present a system for visualizing a graph with an associated hierarchy generated from a query of the search engine. In particular, a search query produces a graph where the nodes denote a set of retrieved documents, and an edge exists between two documents if they are sufficiently semantically related. The strength of such a semantic relationship is represented as the edge weight. A hierarchy of the graph is formed by recursively applying a topological clustering algorithm. Cuts of this hierarchy are visualized by using orthogonal graph drawing algorithms, as users explore the hierarchy. The Clovis system [22] supports interactive clustering of an input graph based on querying the attribute values associated with nodes and edges of the graph. The graph hierarchy is superimposed on the graph using containment. The system, however, limits in hierarchies with a small number of