A STUDY OF ALGORITHMS FOR DETECTING COMMUNITY IN NETWORKS

Nguyen The Vinh[1*], Tran Thanh Thuong[1], Trieu Xuan Hoa[2], Hoang Thi Hong Hanh[1]
[1]Thai Nguyen University, [2]University of Agriculture and Forestry - TNU

ABSTRACT

Detecting community structure in networks is highly desirable in many application domains, such as finding proteins with similar functionality in a biological pathway or automatically grouping relevant people in a social network. However, this is still a daunting task due to the network sizes, as well as the complicated relations between entities. This paper provides a study on algorithms to find communities in a network using edge betweenness centrality. We have implemented the modularity in order to compute the suitable network structure on weighted undirected networks. We also discuss the pros and cons of existing techniques in detecting community structures. To highlight the benefits of the selected techniques, we demonstrate their applications on various datasets, including Victor Hugo's Les Misérables, the movies's network of actors, the author's collaboration network in visualization publications, and the protein interaction network.

Keywords: *algorithms, detecting community, networks, community structure, edge betweenness centrality*

INTRODUCTION

Network is an important representation in data visualization and visual analytics. A network consists of vertices and edges representing individual entities and relationships between them. Dense connections between related nodes form cliques/communities. However, visually detecting such communities is challenging even with a small network. Many solutions [4], [14], [15] have been proposed to automatically detect community structure in networks. One effective technique is based on edge betweenness centrality [14], the sum of the fraction of all-pairs' shortest paths that pass through a given edge. However, this is still an on-going challenge due to the increasing amount of data, especially the complicated relationships between entities. One instance of such real-world data is the biological network of protein interactions which may contain millions of proteins and billions of connections between them [6]. Additionally, edges may have different weights to indicate the levels of correlations between related entities (and usually shown by the thicknesses of edges in the graph). These are weighted networks.

The research presented in this paper focuses on some available techniques on edge betweenness centrality algorithm, focusing on the undirected weighted network. The main contributions of this paper include:

- We integrate virtual nodes method into Newman edge betweenness algorithm [14]. We then apply a sampling technique [12] on edge betweennness centrality which significantly reduces the computing time.

We demonstrate the usefulness of our revised technique on various real-world data. The data and community detection techniques (implemented in forms of javascripts library) are freely provided to research community.

We conduct a quantitative study on networks with different features. The results from this study can serve as a guideline for selecting the community detection techniques for a given network.

The overview structure of this paper is as follows. In Section 2, we describe related work. Section 3 and 4 give a brief summary of algorithms for detecting community structures based on edge betweenness centrality. Next, we compare the performance of these techniques on real-world datasets of different sizes. Finally, we discuss the

*Email: vinhnt@tnu.edu.vn

implementation and scalability of the discussed techniques.

RELATED WORK

A variety of research aims at detecting the network structures [19], [2], [18]. Some popular techniques include Brandes's [4] using betweenness centrality and Newman's [14] using edge betweenness These algorithms can implement on the network using hierarchical clustering. Note that the centrality betweenness can be used in edge betweenness with an equivalent time of O(mn) and O(nm + n^2logm) on unweighted and weighted networks, respectively, which needs O(m + n) space where n is number of vertices and m is number of edges. More recently, Yang and Chen [21] introduced an evolution of Brandes' algorithm using virtual nodes on a weighted network with less running time than original ones, O($\overline{w}\overline{D}n^2$) and O(n + ($2\overline{w}$ − 1)m) where $\overline{w}$ is the average weight of edges and $\overline{D}$ is the average degree of network (we use these nota-tions for the rest of the paper) We use one method based on sampling techniques [17, 1, 20] to reduce computational time. According to the edge betweenness algorithm [14], the general form of detecting communities structure is as follows:

1 Calculate pair dependencies value for all edges.

2. Remove the most "betweenness" communities (the edge with the highest pair dependencies value) from network

3 Update the network. and recalculate betweenness for all remaining edges

4. Repeat from step 2 until no edge remains.

As the results, all edges are presented on a hierarchical structure, known as the dendrogram. Random Walks [15] follows a different approach for creating dendrogram but it costs more spaces for calculation O(n^2) Figure 1 demonstrates an example of grouping ten vertices in a network into three different groups. In particular, Figure 1(a) shows the dendrogram sliced by a vertical dotted slider. Figure 1(b) shows three dynamic communities corresponding to the current cut-off value of the slider. The number of communities in networks reduces as we move the slider to the left and vice versa In other words, by using a slider to control the cut-off level of the dendrogram, we can merge communities into a larger community or separate a larger community into smaller ones.

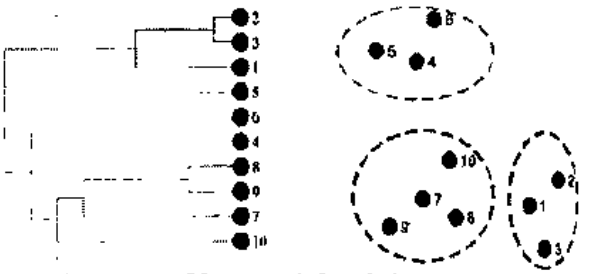

Figure 1. *Using a slider (left) to control community formation (right). The above graph is generated using in D3.js*

This raises another question. what is the cut-off value which gives the best estimation for community structure in a given network? Girvan and Newman [14] use modularity measure for this purpose Regarding running time of this algorithm, the worst-case scenario is O(m^2n) or O(n^3) on sparse graphs. In the next publication, Newman [13] proposes an adjusted algorithm which promotes O((m+n)n) in running-time or O(n^2) on sparse graphs. Figure 2 illustrates the community configurations for unweighted (left) and weighted graph (right) at the maximum modularity Q of each network.

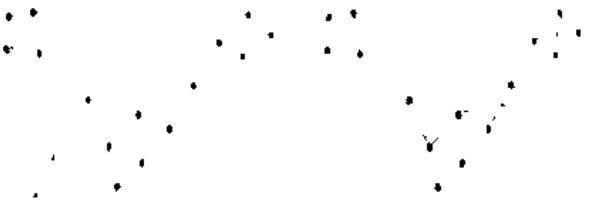

Figure 2. *Community structures of unweighted graph (left) and weighted graph (right): Thicker links represent edges with more weights*

METHOD

We implement algorithms in javascript and D3.js library [3] for discovering community structures of weighted network.

Edge betweenness

An important part of computing edge betweenness centrality is how we score the edge values (the pair dependencies values).

We suggest to use Newman algorithm [14] with Breadth First Search (BFS) and virtual nodes (thereafter, called VS) for weighted network. There are two steps of scoring edge value:

1. Use BFS to find the predecessors and descendants of each node.

2 Calculate pair dependencies value.

Virtual nodes

By adding additional nodes (VS) into the original network, the algorithm turns the weighted network into an unweighted network. An example of inserting VS is depicted in Figure 3. The strategy is to add nv virtual nodes between i and j where $n_v = w_{ij}/\overline{w} - 1$ (rounded to the nearest integer).

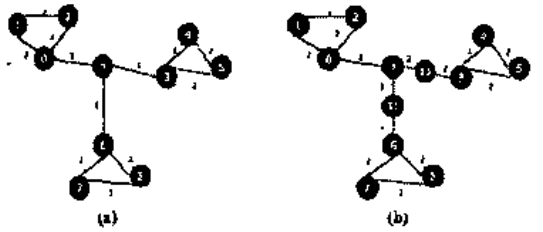

Figure 3 *An example of VS· (a) Original network with 10 nodes, (b) New network with two more virtual nodes. In this example,* $\overline{w}$ *= 2 33 (or 28/12), therefore* n_v *= 1 for the edge 6-9 and the edge 3-9. Orange are actual nodes while gray are virtual nodes*

For finding shortest path betweenness, we now use BFS on the modified network with virtual nodes (VS): if $n_v(i, j) > 1$, we add new VS nodes into the list of vertices. We start BFS at s (an actual vertex) with distance $d_s = 0$, weight $w_s = 1$.

1. For each neighbor *i* of *s*; we assign $d_i = d_s + 1$, weight $w_i = w_s$. Add s into *Pathu[i]* and if *i* is real node, we add *i* into *Pathd[s]*. *Pathu[s]* and *Pathd[s]* are the lists of predecessors and descendants of s respectively.

2. For each neighbor *j* of *i*, we consider two main cases

a If (d_j == null) (or *j* has not yet visited by any nodes) then $d_j = d_i + 1$, $w_j = w_i$. If *i* is a real node, we add *i* into *Pathu[j]* and if *j* is real, we add *j* into *Pathd[i]*. If *i* is a VS, we add previous real node into *Pathu[j]* and if *j* is real, we add *j* into *Pathd* of the previous real node

b. If ($d_j == d_i+1$) (or j has already visited), we set $w_j = w_j + w_i$. If i is real, we add *i* into *Pathu[j]* If i is a VS, we add the previous real node into *Pathu[j]*. And if *j* is real, we add *Pathd[j]* into *Pathd* of the previous real node

3 Repeat step 2 until all vertices have been visited and scored

Figure 4 illustrates the algorithm using BFS combined with VS (starting at vertex *s*).

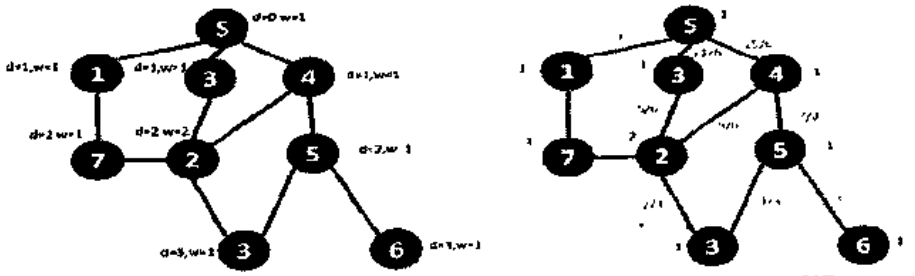

Figure 4. *Example network (orange and green are real vertices while gray is a VS vertex, green (leaf nodes) have no shortest path from s going through): (a) VS 7 is added into the network (b) edge scores. Notice that we do not calculate the accumulated scores for edges connecting the VS 7*

Eliminate edge

We first insert the edge with highest score into *distance* array and remove this edge from the network. We then recalculate edge betweenness centrality. We restart the algorithm in Section 3.2 until no edge remains. The *distance* array is used to build the dendrogram.

Compared to the original work of Newman [14, 21], we made some modifications in step 2 of the algorithm in Section 3.2 Besides, an additional property of VS nodes defines for fast query. *Pathu* and *Pathd* are used for fast access of edge values. The running time of this process is about $O(m(\overline{w}\overline{D}n^2 + m))$ for weighted network.

Detecting network structure

Modularity is a main factor for deciding cluster formation. Along with building a dendrogram by removing edge from distance array, we recompute the modularity Q based on the following equation.

$Q = \sum_i (e_{ii} - a_i^2)$ (1)

where i and j are communities, e_{ij} is the fraction of edges with one end vertices in community i and the other in community j, and a_i is the fraction of ends of edges that are attached to vertices in community i

RANDOM SAMPLING METHOD

Yang and Chen [21] show that VS algorithm does not outperform Brandes' algorithm when $\overline{w} > 2$. In other words, VS algorithm is not a preferable choice In addition, real world networks, such as social networks, biological networks, and communication networks are so complicated which makes exhaustively edge betweenness centrality (for all pairs of nodes) so exponentially expensive and unpractical Therefore, Matteo and Evgenios [12] recently introduced a sampling algorithm which estimates the betweenness centrality based on vertex-diameter (and independent from the network size). Given an undirected unweighted network, the sample size is calculated as follows:

$r = \frac{c}{\epsilon^2}([\log_2(VD(G)-2)]+1+\ln\frac{1}{\delta})$

where the two parameters $\varepsilon.\delta \in (0,1)$, and c is an universal positive constant. *VD(G)* in this case can be described as 2-approximation vertex-diameter:

1 Vertex v is selected at random

2 Calculate all shortest paths from vertex v to all other vertices

3. 2-approximation vertex-diameter *VD(G)* is the sum of the length of two shortest paths with maximum size from v to two other distinct vertices u and w

Random sampling is one of three techniques studied in the next Section.

EVALUATION

In this section, we study the running time of community detection algorithms on four weighted real-world networks of various sizes, namely: Victor Hugo's Les Misérables dataset [10, 11], the movie's network of actors available on the IMDB website [9], the author's collaboration network in Visualization publications [8], and the protein interaction network [5]. In the last three datasets, we extracted the sub-networks (from the original data) of 250, 500, 750, and 1,000 highest-degree vertices for testing purposes. Table 1 displays prominent features of networks in our study. The last two columns are colored differently since these attributes are specific for VS and sampling techniques

Table 1. *Prominent features of datasets in our study*

Dataset	n	m	$\overline{w}$	$\overline{D}$	Q	#VS	#sample
Victor Hugo	77	254	3 21	6.6	0 51	170	215
IndexCards	250	579	1 07	4.6	0 83	295	265
	500	1,108	1 01	4 4	0 81	562	115
	750	1,433	1 04	3 8	0.91	816	315
	1,000	1,761	1 04	3.5	0.92	1,074	315
VisPublication	250	663	1 88	5 3	0 85	457	265
	500	1,436	1.73	5 7	0 72	838	265
	750	2,178	1 59	5 8	0 90	1,226	265
	1,000	2,894	1 53	5 7	0 93	1,483	315
IMDB	250	721	2 62	5 7	0 55	371	215
	500	1,126	2 49	4 5	0 69	638	265
	750	1,461	2 40	3 9	0 72	888	265
	1,000	1,772	2 35	3 5	0 76	1,146	265

Victor Hugo's Les Misérables network is depicted in Figure 5. In particular, the top left panel shows modularity graph by the number of clusters (which guides the selection of number of clusters using a slider). Below it, we show the network dendrogram. The current selection (the black dotted line) of cluster numbers is also reflected on the dendrogram. By default, we set the number of clusters at the maximum Modularity Q. In this example, we have 7 communities (for 77 vertices) at the maximum Modularity Q = 0.51. As the number of communities changes (by setting the slider), the graph layout gets updated dynamically to better reflect the community structure changes

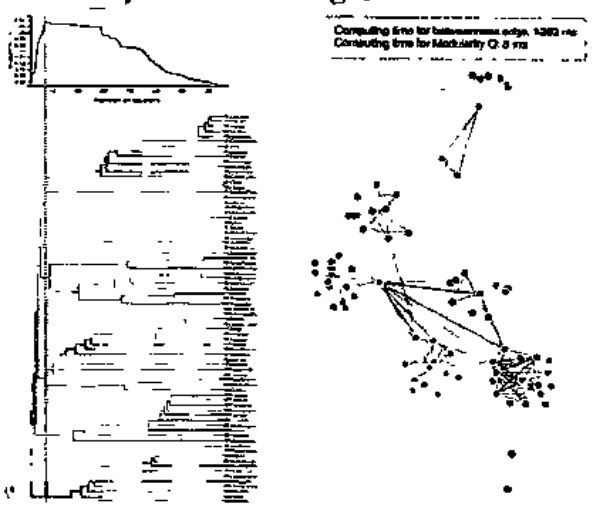

Figure 5 *Complete interface of our web application for the Victor Hugo dataset: number cluster selected by a slider (left) and the associated network visualization (right) where nodes are colored by cluster*

Figure 6 illustrates the results from our study on running times of community detection algorithms. All tests are performed on a 3.2 GHz Intel Core i5, macOS Sierra, 8GB RAM iMac PC running JavaScript and D3.js [3]. As noticed, all networks in Table 1 are weighted. However, random sampling technique only works on an unweighted network. We convert the weighted networks into unweighted networks by adding VS.

Here are some observations from empirical analysis:

Brandes and VS running times exponentially increase with network sizes but random sampling technique is almost linear

• As depicted in Figure 6 (a) at the network size of 250, the sampling method (green) is slower than other two algorithms. This rare case occurs on smaller networks with higher vertex-diameter. The number of iterations (or sample size of the first row for the Indexcards dataset in Table 1) to calculate edge betweenness centrality is greater than network size (265>250).

For small and highly weighted networks, we suggest using Brandes' algorithm due to high accuracy and smaller computational time. But for larger networks, VS with random sampling technique reduces running time significantly.

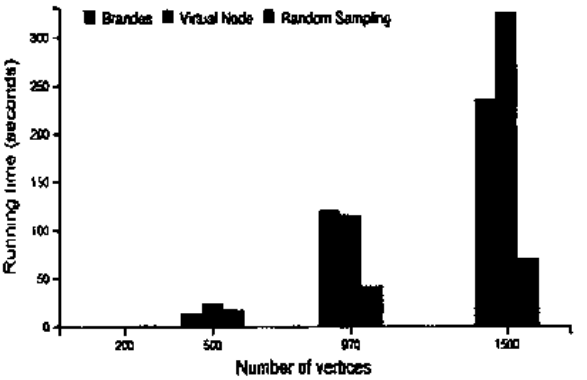

Figure 6. *Clustering times of algorithms on different network sizes. blue for Brandes' algorithms, orange for VS, green for random sampling. The last graph has a different scales on y-axis*

CONCLUSIONS

In this paper, we have applied VS and random sampling method for faster community detection on weighted networks. The running times of different algorithms are compared in Section 5. As depicted in Figure 5, Brandes' algorithm [14] does not scale well with the network size. VS [21] is even worse in most cases since it introduces many virtual nodes into the existing network. However, VS helps to convert weighted networks into unweighted ones for which we can apply random sampling [12] to reduce the running times. Notice that the random sampling method is independent from network size but depends on vertex-diameter of the given network. Although our empirical study is limited on a single machine (can not handle

large networks), Figure 5 clearly depicts that sampling technique combined with VS scales are better, compared the Brandes' algorithm.

The source code of all studied algorithms, online demos, sample data, and more examples are provided via our GitHub project repository, located at https: //github.com/iDataVisualizationLab/NetworkClustering.

One possible extension on visualization side is on multilayer clustering of large networks and supporting multi-stop navigation. In other words, we want to provide a "google map"-like tool for large networks The ability to automatically detect the community formation/change over time [16, 7] is also another interesting future research.

REFERENCES

1. Arun S. Maiya T Y B -W (2010), "Sampling community structure" *Proceedings of the 19th international conference on World wide web*, pp 701–710.

2 B. H. Good Y -A D M , Clauset A. (2010), "The performance of modularity maximization in practical contexts", *Phys. Rev. E*, 81, 046106

3. Bostock M., Ogievetsky V., Heer J. (2011), "D3 Data-driven documents". *IEEE Transactions on Visualization and Computer Graphics* 17, 12, pp. 2301–2309.

4 Brandes U. (2001), "A faster algorithm for betweenness. centrality". *Journal of Mathematical Sociology*, 25, pp. 163-177

5. Cerami E. G., Gross B E , Demir E., Rodchenkov I , Babur Ö., Anwar N , Schultz N., Baderg D , Sander C (2011), "Pathway commons, a web resource for biological pathway data", *Nucleic acids research*, 39, suppl 1, pp. D685–D690.

6. Dang T. N., Murray P., Forbes A G (2015), "Path-wayMatrix· Visualizing binary relationships between proteins in biological pathways", *BMC Proceedings 9*, 6 (2015), S3 10 1186/1753-6561-9-S6-S3.

7 Dang T N , Pendar N , Forbes A. G (2016), *TimeArcs: Visualizing Fluctuations in Dynamic Networks*, Computer Graphics Forum

8. Isenberg P., Heimerl F., Koch S , Isenberg T., Xu P., Stolper C., Sedlmair M., Chen J., Möller T , Stasko J. (2015), *Visualization publication dataset* Dataset: http.//vispubdata.org/, Published Jun. 2015

9 IMDB Imdb support community. Dataset· http //www.imdb.com/interfaces, accessed Jan, 2017

10. Knuth D. E. (1993), "The Stanford GraphBase· A Platform for Combinatorial Computing", *Addison-Wesley Reading*, vol. 37.

11 Kunegis J. (2013), "KONECT – The Koblenz Network Collection" *In Proc. Int. Conf. on World Wide Web Companion*, pp. 1343–1350.

12. Matteo Riondato E M. K. (2015), "Fast approximation of betweenness centrality through sampling". *Data Min Knowl Disc*, 30 (2015), pp 438–475

13. Newman M. E. J. (2004), "Fast algorithm for detecting community structure in networks" *Phys. Rev* 69 (2004), 066133

14. Newman M E. J., Girvan M (2003), *Finding and evaluating community structure in networks*. Preprint cond-mat/0308217.

15 Pons P., Latapy M (2006), "Computing communities in large networks using random walks", *Journal of Graph Algorithms and Applications* 10, 4, pp. 191–218

16. Reda K., Tantipathananandh C , Johnson A , Leigh J , Berger-Wolf T. (2011), "Visualizing the evolution of community structures in dynamic social networks" *Proceedings of the 13th Eurographics/ IEEE - VGTC Conference on Visualization (Chichester, UK, 2011)*, EuroVis'11, pp. 1061–1070.

17. Se-Young Yun A. P. (2014), *Community detection via random and adaptive sampling*, Social and Information Networks.

18. T Aynaud V. D. Blondel J -L. G. R. L (2011), "Multilevel local optimization of modularity". *John Wiley Sons*, pp 315-345.

19. Wakita K., Tsurumi T (2007), *Finding community structure in mega-scale social networks* e-print (2007) cs.CY/0702048.

20. Xin Yu. Jing Yang Z -Q X (2015), "A semantic overlapping community detection algorithm based on field sampling", *Expert Systems with Applications* 42, 1 (2015), pp 366–375

21. Yang J , Chen Y (2011), *Fast computing betweenness centrality with virtual nodes on large sparse networks*, PLoS ONE, 6·e22557

TÓM TẮT

NGHIÊN CỨU CÁC THUẬT TOÁN ĐỂ XÁC ĐỊNH CÁC CỘNG ĐỒNG TRONG MẠNG

Nguyễn Thế Vịnh[1*], Trần Thanh Thương[1], Triệu Xuân Hòa[2], Hoàng Thị Hồng Hạnh[1]

[1]Đại học Thái Nguyên, [2]Trường Đại học Nông Lâm – ĐH Thái Nguyên

Việc phát hiện cấu trúc nhóm trong các liên kết mạng đóng vai trò quan trọng và được ứng dụng trong nhiều lĩnh vực thực tiễn chẳng hạn như việc nhóm và phân loại các proteins có cấu trúc, chức năng tương tự với nhau hoặc nhóm những người có cùng sở thích hoặc mối quan hệ nào đó trên mạng xã hội một cách tự động để giúp các nhà phân tích, quản lý đưa ra các phương pháp chiến lược, hoạch định đối với từng nhóm. Tuy nhiên, việc tự động phát hiện cấu trúc trên vẫn còn gặp nhiều khó khăn do cấu trúc mạng vô cùng lớn cũng như mối quan hệ phức tạp giữa các thực thể với nhau. Một cá nhân có thể tham gia nhiều nhóm cùng lúc trong mạng. Trong bài báo này, tác giả nghiên cứu một số thuật toán tự động phát hiện cấu trúc nhóm trong mạng dựa trên phân tích các mối quan hệ giữa các nút mạng (hay còn gọi là edge betweenness centrality). Tiêu chí để so sánh các thuật toán với nhau dự trên giá trị modun (Modularity), giá trị này dùng để đánh giá chất lượng việc tự động phân nhóm. Ngoài ra, tác giả cũng đánh giá ưu điểm, nhược điểm của từng thuật toán đối với mỗi mạng có cấu trúc khác nhau và đưa ra gợi ý để lựa chọn thuật toán phù hợp. Để làm rõ tính ưu việt của từng thuật toán, nhóm tác giả tiến hành chạy từng thuật toán trên các tập dữ liệu khác nhau, bao gồm dữ liệu Les Misérables của Victor Hugo, dữ liệu về mạng lưới các diễn viên trong một bộ phim, dữ liệu về các tác giả trong cùng một bài báo và dữ liệu về mạng tương tác giữa các protein

Từ khóa: *thuật toán, xác định cộng đồng, mạng, cấu trúc nhóm, mối quan hệ giữa các nút mạng*

Ngày nhận bài:03/5/2017; Ngày phản biện: 17/5/2017; Ngày duyệt đăng: 31/5/2017

* *Email: vinhnt@tnu.edu.vn*