Clustering Networks by Structures

Xiaowei Xu
University of Arkansas at Little Rock
xwxu@ualr.edu

Nurcan Yuruk
University of Arkansas at Little Rock
{nxyuruk@ualr.edu}

Thomas A. J. Schweiger
Acxiom Corporation
Tom.Schweiger@acxiom.com

ABSTRACT

This paper presents some new ways of uncovering underlying structures, including the roles that vertices play in the network. We begin by defining a property we call structural similarity. We then define a structure-connected cluster (SCC) as a collection of vertices they have strong structural similarity. Finally, we propose an algorithm that only finds SCCs that represent groups of peers in networks, but also identify vertices that play special roles such hubs that bridge clusters and outsiders that are marginally connected to clusters. Identifying vertex roles is useful for applications such as viral marketing and epidemiology. For example, hubs are responsible for spreading ideas or disease. In contrast, outsiders have little or no influence, and may be isolated as noise in the data. An empirical evaluation of the algorithm using both synthetic and real datasets demonstrates superior performance over other methods such as modularity-based algorithms.

1. INTRODUCTION

Networks are ubiquitous. Common networks include social networks, the World Wide Web, and computer networks. A network consists of a set of vertices interconnected by edges. A vertex represents some real entities such as a person, website, or piece of networking hardware. An edge connects two vertices if they have some relationship such as a friendship, hypertext link, or wired connection. In such networks, each vertex plays a role. Some vertices are members of clusters; a group of peers in a social network or a group of related websites in the WWW are examples. Some vertices are hubs that bridge many clusters but don't belong strongly to any one cluster; for example politicians tend to play such a role in social networks, and websites like wikipedia.org are clearing-houses for all kinds of information. Some vertices represent outsiders that have only weak associations with any cluster; for example a loner in a social network, or a parked domain on the internet.

To illustrate these points further, consider the network in Figure 1. If might confidently consider the vertices {0,1,2,3,4,5} and {7,8,9,10,11,12} to be clusters of peers. Vertex 6 is difficult to classify. It could arguably belong to either cluster or to none. It is an example of a hub. Likewise, vertex 13 is weakly connected to a cluster. It is an example of an outsider.

Network clustering (or graph partitioning) is the detection of structures like those in Figure 1, and it is drawing increased attention and application in computer science [1][2], physics [7], and bioinformatics [3]. Various methods have been developed. They tend to partition on the principle that clusters should be sparsely connected, but the vertices within each cluster should be densely connected. Modularity-based algorithms [3][7][8] and normalized cut [1][2] are successful examples. However, they do not distinguish the roles of vertices.

The modularity-based algorithm [8] will cluster the network in Figure 1 into two clusters: one consisting of vertices 0 to 6 and the other consisting of vertices 7 to 13. It doesn't isolate vertex 6 or vertex 13.

The identification of hubs gives valuable information. For example, hubs in the WWW are deemed authoritative information sources among web pages [4], and hubs in social networks are believed to play a crucial role in vial marketing [5] and epidemiology [6].

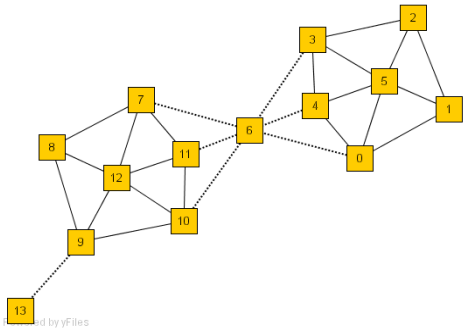

Figure 1. A Network with 2 Clusters, a Hub and an Outsider.

In this paper, we propose a new method for network clustering. The goal of our method is to find clusters, hubs, and outsiders in large networks. To achieve this goal, we use the neighborhood of the vertices as clustering criteria instead of only their direct connections. Vertices are clustered by how they share neighbors. Doing so makes sense when you consider the detection of communities in large social networks. Two people who share many friends should be clustered in the same community.

Referring to Figure 1, consider vertices 0 and 5, which are connected by an edge. Their neighborhoods are the vertex sets {0, 1, 4, 5, 6} and {0, 1, 2, 3, 4, 5}, respectively. They share many neighbors and thus are reasonably clustered together. In contrast, consider the neighborhoods of vertex 13 and vertex 9. These two vertices are connected, but share only few common neighbors, i.e. {9, 13}. Therefore, it is doubtful that they should be grouped together. The situation for vertex 6 is a little different. It has many neighbors, but they are sparsely interconnected.

Our algorithm identifies two clusters, {0, 1, 2, 3, 4, 5} and {7, 8, 9, 10, 11, 12}, and isolates vertex 13 as an outsider and vertex 6 as a hub.

Our algorithm has the following features:

- It detects clusters, hubs, and outsiders by using the structure and the connectivity of the vertices as clustering criteria.
- It is efficient. It clusters the given network by visiting each vertex exactly once.

We will demonstrate that our algorithm finds meaningful clusters and identifies hubs and outsiders in very large networks. With respect of efficiency, our algorithm's running time on a network with n vertices and m edges is $O(m)$.

2. EVALUATION

In this section we evaluate our algorithm using both synthetic and real datasets. The performance of the algorithm is compared with fast modularity-based network clustering algorithm proposed by Clauset *et al* in [8], which is faster than many competing algorithms: its running time on a network with n vertices and m edges is $O(md \log n)$ where d is the depth of the dendrogram describing the hierarchical cluster structure. We implemented our algorithm in C++. We used the original source code of fast modularity algorithm by Clauset *et al* [11]. All the experiments were conducted on a PC with a 2.0 GHz Pentium 4 processor and 1 GB of RAM.

2.1 Efficiency

To evaluate the computational efficiency of the proposed algorithm we generate ten networks with the number of vertices ranging from 1,000 to 1,000,000 and the number of edges ranging from 2,182 to 2,000,190. We adapted the construction as used in [7] as follows: first we generate clusters such that each vertex connects to vertices within the same cluster with a probability P_i, and connects to vertices outside its cluster with a probability $P_o<P_i$. Next we add a number of hubs and outsiders. An example of a generated network is presented in Figure 2.

The running time for fast modularity and our algorithm on the synthetic networks are plotted in Figure 3 and Figure 4, respectively. Running time is plotted both as a function of the number of nodes and the number of edges. Figure 4 shows that our algorithm's running time is in fact linear w.r.t. to the number of vertices and the number of edges, while fast modularity's running time is basically quadratic and scales poorly for large networks. Note the difference in scale for the y-axis between the two figures.

2.2 Effectiveness

To evaluate the effectiveness of network clustering, we use real datasets whose clusters are known a priori. These real datasets include American College Football and Books about US politics. We use adjusted Rand index (ARI) [10] as a measure of effectiveness of network clustering algorithms in addition to visually comparing the generated clusters to the actual. The ARI is defined as follows:

$$\frac{\sum_{i,j}\binom{n_{ij}}{2} - \left[\sum_i \binom{n_{i\cdot}}{2}\sum_j \binom{n_{\cdot j}}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_{i\cdot}}{2} + \sum_j \binom{n_{\cdot j}}{2}\right] - \left[\sum_i \binom{n_{i\cdot}}{2}\sum_j \binom{n_{\cdot j}}{2}\right] / \binom{n}{2}}$$

Where $n_{i,j}$ is the number of vertices in both cluster x_i and y_j; and $n_{i,\cdot}$ and $n_{\cdot,j}$ is the number of vertices in cluster x_i and y_j respectively. The ARI lies between 0 and 1. When the two clustering agree perfectly, the ARI is 1.

2.2.1 College Football

The first real dataset we examine is the 2006 NCAA Football Bowl Subdivision (formerly Division 1-A) football schedule. This example is inspired by the set studied by Newman and Girvan [7], who consider contests between Div. 1-A teams in 2000. Our set is more complex, considering all contests of the Bowl Subdivision schools including those against schools in lower divisions.

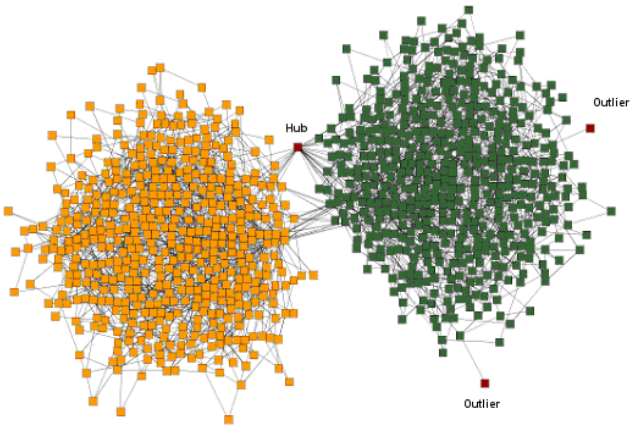

Figure 2. A Synthetic Network with 1,000 Vertices

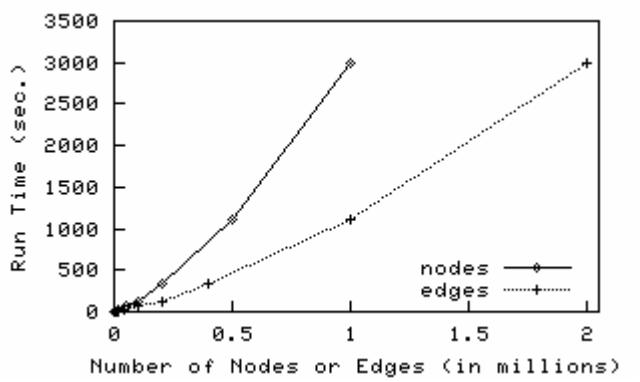

Figure 3. Running Time for fast modularity algorithm

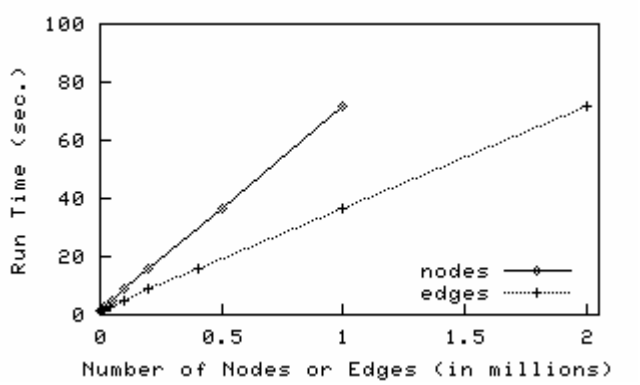

Figure 4. Running Time for our algorithm

The challenge is to discover the underlying structure of this network – the college conference system. The National Collegiate Athletic Association (NCAA) divides 115 schools into eleven conferences. In addition there are four independent schools at this top level: Army, Navy, Temple, and Notre Dame. Each Bowl Subdivision school plays against schools within their own conference, against schools in other conferences, and against lower division schools. The network contains 180 vertices (119 Bowl Subdivision schools and 61 lower division schools)

interconnected by 787 edges. Figure 5 shows this network with schools in the same conference identified by color.

This example illustrates kinds of structures that our method seeks to address. Schools in the same conference are clusters. The four independent schools play teams in many conferences but belong to none; they are hubs. The lower division schools are only weakly connected to the clusters in the network; they are outsiders.

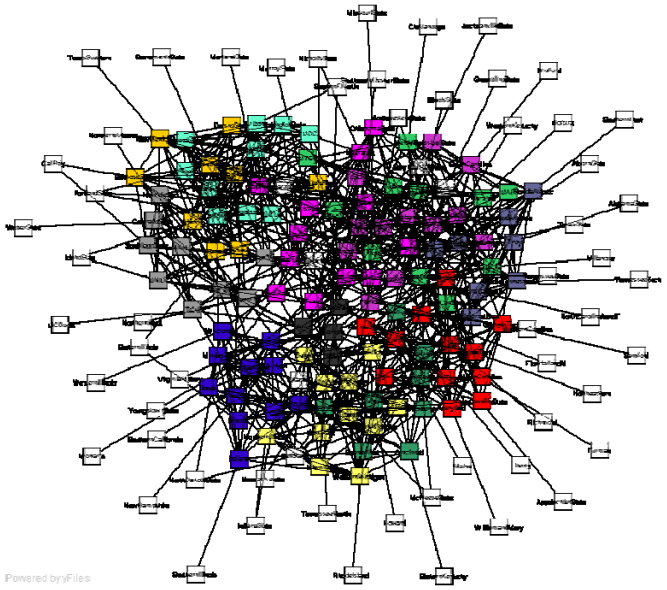

Figure 5. NCAA Football Bowl Subdivision schedule as a network, showing the 12 conferences in color, independent schools in black, and lower division schools in white.

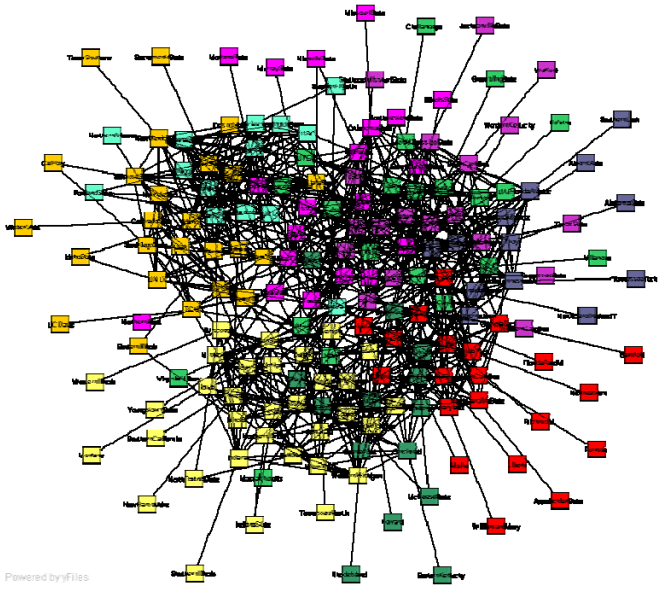

Figure 6. NCAA Football Bowl Subdivision schedule as clustered by fast modularity algorithm.

First we cluster this network by using the fast modularity algorithm. The results, for which the modularity is 0.599 is shown in Figure 6. Maximizing Newman's modularity gives a satisfying network clustering, identifying nine clusters. All schools in the same conference are clustered together. However, two of the conferences are merged (the Western Athletic and Mountain West conferences and the Mid-American and Big Ten conferences), the four independent schools are classified into various conferences despite their hub-like properties. All lower division teams are assigned to clusters.

Next we cluster the network using our algorithm, using the parameters (ε= 0.5, μ = 2). This clustering succeeds in capturing all the features of the network. Eleven clusters are identified, corresponding exactly to the eleven conferences. All schools in the same conference are clustered together. The independent schools and the lower division schools are unclassified – they stand apart from the clusters. The four independent schools show strong properties as hubs; they have inactive edges that connect them to a large number of clusters – at minimum five. In contrast the lower division schools have only week connections to clusters – one or two, and in a single case three. They are true outsiders. This clustering matches perfectly the underlying structure shown in Figure 5.

2.2.2 Books about US politics

The second example is the classification of books about US politics. We use the dataset of Books about US politics compiled by Valdis Krebs [12]. The vertices represent books about US politics sold by the online bookseller Amazon.com. The edges represent frequent co-purchasing of books by the same buyers, as indicated by the "customers who bought this book also bought these other books" feature on Amazon. The vertices have been given values "l", "n", or "c" to indicate whether they are "liberal", "neutral", or "conservative". These alignments were assigned separately by Mark Newman [13] based on a reading of the descriptions and reviews of the books posted on Amazon. The political books network is illustrated in Figure 7. The "conservative", "neutral" and "liberal" books are represented by red, gray and blue respectively.

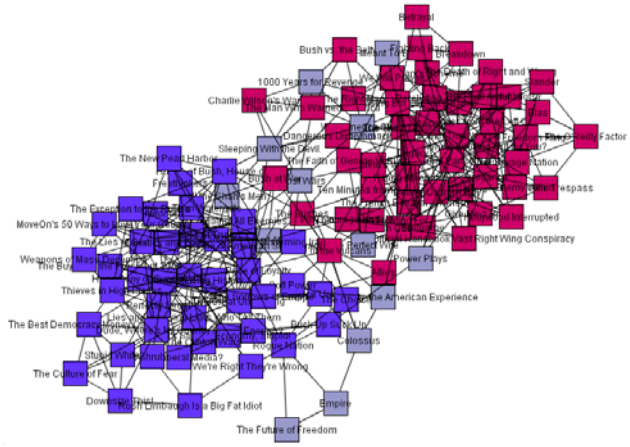

Figure 7. Political Book Network.

First we apply our algorithm to the political books network, using the parameters (ε= 0.35, μ = 2). Our goal is to find clusters that represent the different political orientations of the books. The result is presented in Figure 8. Our algorithm successfully find three clusters representing "conservative", "neutral" and "liberal" books respectively. The obtained clusters are illustrated using three different shapes: squares for "conservative" books and

triangles for "neutral" books and circles for "liberal" books. Additionally, each vertex is labeled with the book title.

The result for the fast modularity algorithm is presented in Figure 9. The fast modularity algorithm found 4 clusters, presented using circles, triangles, squares, and hexagons. Although two dominant clusters, represented by circles and squares, align well with the "conservative" and "liberal" classes, the "neutral" class is mostly misclassified. This demonstrates again that fast modularity algorithm can not handle vertices that bridge clusters.

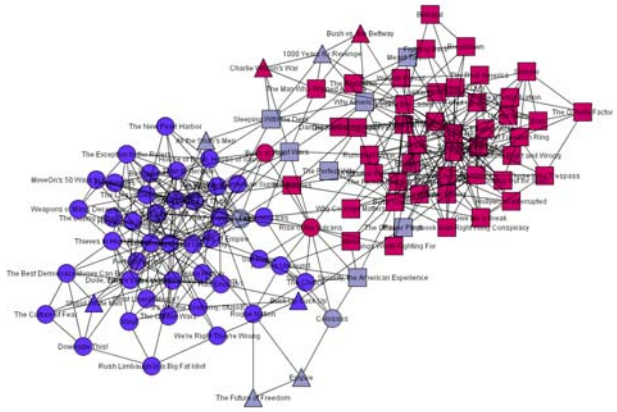

Figure 8. The Result of our algorithm on Political Book Network.

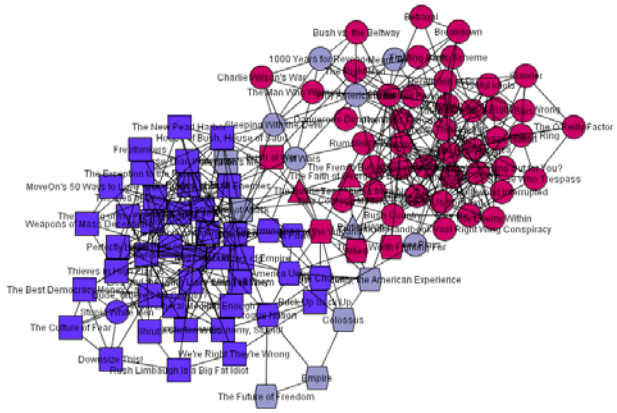

Figure 9. The Result of fast modularity algorithm on Political Book Network.

2.2.3 Adjusted Rand Index Comparison

The Adjust Rand Index is an effective measure of the similarity of a clustering result to the true clustering. The results for College Football, Political Books, CG1 and CG2 are presented in Table 1.

Table 1. Adjust Rand Index Comparison.

	Our algorithm	Fast modularity algorithm
College football	**1**	0.24
Political books	**0.71**	0.64

The ARI results clearly demonstrate that the proposed algorithm outperforms fast modularity algorithm at producing clustering that resemble the true clustering for the real world networks in our study.

3. CONCLUSIONS

In this paper, we proposed a new algorithm to detect clusters, hubs and outsiders in networks. It clusters vertices based on their common neighbors. Two vertices are assigned to cluster according to how they share neighbors. This makes sense when you consider social communities. People who share many friends create a community, and the more friends they have in common, the more intimate the community. But in social networks there are different kinds of actors. There are also people who are outsider (like hermits), and there are people who are friendly with many communities but belong to none (like politicians). The latter play a special role in small-world networks [9].

We applied our algorithm to some real world networks. The empirical evaluation demonstrates superior performance over the modularity-based network clustering algorithm.

4. REFERENCES

[1] C. Ding, X. He, H. Zha, M. Gu, and H. Simon, "A min-max cut algorithm for graph partitioning and data clustering", Proc. of ICDM 2001.

[2] J. Shi and J. Malik, "Normalized cuts and image segmentation", IEEE Trans. On Pattern Analysis and Machine Intelligence, Vol 22, No. 8, 2000.

[3] R. Guimera and L. A. N. Amaral, "Functional cartography of complex metabolic networks." Nature 433, 895–900 (2005).

[4] J. Kleinberg. "Authoritative sources in a hyperlinked environment." Proc. 9th ACM-SIAM Symposium on Discrete Algorithms, 1998.

[5] P. Domingos and M. Richardson, "Mining the Network Value of Customers", Proc. 7th ACM SIGKDD, pp. 57 – 66, 2001.

[6] Y. Wang, D. Chakrabarti, C. Wang and C. Faloutsos, "Epidemic Spreading in Real Networks: An Eigenvalue Viewpoint", SRDS 2003 (pages 25-34), Florence, Italy

[7] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks", Phys. Rev. E 69, 026113 (2004).

[8] A. Clauset, M. Newman, and C. Moore, "Finding community in very large networks", 2004.

[9] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," Nature, 393:440-442 (1998)

[10] L. Hubert and P. Arabie, "Comparing Partitions". *Journal of Classification*, 193–218, 1985.

[11] http://cs.unm.edu/~aaron/research/fastmodularity.htm.

[12] http://www.orgnet.com/.

[13] http://www-personal.umich.edu/~mejn/netdata/.