Force-Directed Layout Community Detection

Yi Song and Stéphane Bressan

School of Computing,
National University of Singapore
{songyi,steph}@nus.edu.sg

Abstract. We propose a graph-layout based method for detecting communities in networks. We first project the graph onto a Euclidean space using Fruchterman-Reingold algorithm, a force-based graph drawing algorithm. We then cluster the vertices according to Euclidean distance. The idea is a form of dimension reduction. The graph drawing in two or more dimensions provides a heuristic decision as whether vertices are connected by a short path approximated by their Euclidean distance. We study community detection for both disjoint and overlapping communities. For the case of disjoint communities, we use k-means clustering. For the case of overlapping communities, we use fuzzy-c means algorithm. We evaluate the performance of our different algorithms for varying parameters and number of iterations. We compare the results to several state of the art community detection algorithms, each of which clusters the graph directly or indirectly according to geodesic distance. We show that, for non-trivially small graphs, our method is both effective and efficient. We measure effectiveness using modularity when the communities are not known in advance and precision when the communities are known in advance. We measure efficiency with running time. The running time of our algorithms can be controlled by the number of iterations of the Fruchterman-Reingold algorithm.

1 Introduction

Communities detection is instrumental in fields of study such as sociology [9], biology [8], and marketing [21]. Communities exist when nodes in the network form a group in which they are better connected to each other than to the rest of the network. In this paper, we propose methods for finding communities.

We model a network as a simple graph $G(V, E)$, which is undirected, unweighted and without self-loop. V is a set of vertices. E is a set of edges. Our main idea is to obtain a representation of the graph in a Euclidean space and then cluster the vertices based on the Euclidean distance. This is different from what common graph clustering algorithms in that most of them cluster the graph and detect communities directly or indirectly according to geodesic distance. We use Fruchterman-Reingold's force-directed algorithm (FR) (see [11].) This graph layout approach transforms the connections among vertices based on attractive forces and repulsive forces pulling vertices together and pushing them apart, respectively, into proximity in a Euclidean space. In this way, vertices with more

H. Decker et al. (Eds.): DEXA 2013, Part I, LNCS 8055, pp. 419–427, 2013.

connections are closer while vertices without or with less connections are relatively further from each in the Euclidean space of possibly lower dimension that is intrinsic of the graph. Such a dimension reduction is a good opportunity enables the use of the graph layout and Euclidean distance to heuristically detect communities. While the original *FR* algorithm is presented for two dimensions, we consider versions for one, two, three and more dimensions and three dimensions. We evaluate the role of the number of dimensions as a parameter of our methods in terms of its impact on effectiveness and efficiency. For disjoint community detection, the data clustering technique we use is *k-means clustering (KM)*, while for overlapping community, we use *Fuzzy C-mean clustering (FCM)*, which can indicate the strength between each vertex and communities, and thus does not restrict each vertex to belonging to one group only. All these algorithms' complexities are not high and neither is *FR*'s. Our method building on these techniques is thus efficient for large social networks.

We evaluate effectiveness by measuring modularity. Modularity is defined based on this idea that edges between nodes in the same community are dense, and are sparse between different communities. To find communities with natural division, modularity is defined as the number of edges falling within groups minus the expected number in an equivalent (same number of edges and the same degree distribution) graph with edges placed at random [19]. For graphs with known community structures, we measure the precision as well by comparing memberships to communities that our approach discovers with those to the known communities.

The rest of the paper is organized as follows. Section 2 introduces the related works on graph clustering and community detection. Section 3 briefly reviews the background and presents our approach to discovery community structures in networks. Section 4 describes the data sets that we use, and present and analyze the results of our experiments. Finally we conclude in Section 5.

2 Related Work

Community detection is a form of graph clustering. *Graph clustering* methods can be categorized into partition clustering, hierarchical clustering, divisive global clustering, and agglomerative global clustering [23]. Large amount of specific methods are proposed such as Star clustering [3], Repeated Random Walks [17], and Markov Clustering [24].

Some methods are specifically proposed for disjoint community detection. Girvan and Newman, in their pioneering works on community detection ([13,18]), propose a divisive method to identify community. The edges with highest betweenness are removed iteratively, which splits the graph into communities. Clauset et al. in [6] propose a greedy hierarchical agglomerative algorithm which starts from each vertex being a community and then joins two communities at each iteration. Rosvall and Bergstrom in [22] use information theoretic approach to detect community in weighted and directed network.

For overlapping community detection, Du et al. in [7] use maximal cliques for community detecting. They propose an algorithm that enumerates all maximal

cliques, finds clustering kernel in each group of the overlapping maximal cliques. Palla et al. in [12] propose the clique percolation method which finds all cliques of size k and communities are detected by finding connected union of k-cliques. It 's based on an assumption that the network must have a large quantity of similar cliques. Chen et al. in [5] detect overlapping communities utilizing concept from game theory. Lancichinetti et al. in [15], considering various networks features, present a method called Order Statistical Local Optimization Method. It can be applied to weighted, directed graphs besides simple graphs. Jierui and Boleslaw in [25] propose the Speaker-listener Label Propagation Algorithm for overlapping community detection in large-scale networks.

3 Algorithm

3.1 Background

Force-Directed Algorithms. The idea of force-directed algorithms is to achieve a "aesthetically pleasing" graph layout by simulating the whole graph as a physical system. Edges in the graph are seen as springs binding vertices. Vertices are virtually pulled closer together or pushed further apart according to physical forces. The positions of the vertices are adjusted and this procedure continues until the the system comes to an equilibrium. In addition, Fruchterman and Reingold's force-directed algorithm [11] aims at even vertex distribution. The authors define the attractive force and the repulsive force as $f_a(d) = d^2/k$ and $f_r(d) = -k^2/d$, where $k = C\sqrt{\frac{area}{number\_of\_vertices}}$, and d is the distance between every pair of vertices. $area$ is the windows size for display the graph.

***k*-Clustering.** K-means clustering [16] partitions objects to k clustering, assigns each object the cluster with the nearest mean and adjusts their membership untill an optimum is reached. As a soft version of k-means, Fuzzy C-means clustering (FCM) [4] assigns each object a fuzzy degree of membership to each cluster. Instead of belonging to only one cluster, objects classified via this algorithm can belong to several clusters with different strengths. As a general version of k-means, Expectation-maximization algorithm (EM) [2] models clusters using statistic distributions. The reason we adopt k-means, rather than EM is that k-means is effective enough for this problem and k-means is more efficient. We experimentally show this in Section 4.

3.2 Algorithm

We propose an algorithm that can systematically enumerate all possible number of clusters and find the configuration with the highest modularity. Therefore, the algorithm iterates by changing the value of k from 1 to $|V|$ which is the number of vertices in the network. We show the changes of modularity with the change of k value. If the number of clusters is prior knowledge, we can set the number of iterations to be 1 to this numeber.

Our method starts from the *FR* algorithm. The inputs for the algorithm are the edges of the graph only. Output is the coordinates of vertices in Euclidean Space. Then we sort the degrees of the vertices and initialize the centers of the clusters for the clustering by the vertices with highest degrees. The idea is that the vertices with high degree have higher chance of being the community centers. The centers may change during the clustering. We refine the clusters after the data clustering in Euclidean space. If there's any vertex that doesn't have any connection with other vertices in the same cluster, or it has less connections inside its cluster than outside its cluster, then it will be grouped to the cluster where it has the maximum number of connections. In other words, this vertex will be grouped to the cluster that has most immediate neighbors. The refinement process may change the number of clusters, which is actually good for those who only roughly know the number of clusters. They can input the maximum number of clusters they believe and let our method find out the exact number of clusters in the network without trying all the value of k from 1 to $|V|$.

Algorithm 1. Force-directed Layout Community Detection Algorithm

Input: graph G with n vertices, the number of trials t, $t \leq n$.;
Result: Clusters C_i, $i \in (1, 2, ..., k')$

```
1 v = Fruchterman_Reingold(G), v ∈ R^{n*2}, v =[v1;v2;...;vn];
2 Sort_degree(G);
3 k ← 1;
4 for each k ≤ t do
5 |   C'_i = K-means(v);
6 |   C_i = Refinement(C'_i);
7 |   Calculate modularity and record the maximum;
8 Return C_i,i ∈ (1, 2, ..., k') with the maximum modularity;
```

Algorithm 2. Refinement

Input: Clusters C_i, $i \in (1, 2, ..., k)$;
Result: Clusters C'_i, $i \in (1, 2, ..., k')$;

```
1 for i from 1 to k do
2 |   for v ∈ C_i do
3 |   |   find the cluster C_j where v has the maximum number of immediate
  |   |   neighbors;
4 |   |   if i ≠ j then
5 |   |   |   Cluster v into C_j;
6 Return C'_i,i ∈ (1, 2, ..., k');
```

We call the above algorithm *FR-KM* for the experiments. The other two versions of the algorithm are similar to *FR-KM* but depend on different clustering methods. We name the one using expectation-maximization algorithm *FR-EM* and the one using fuzzy c-means algorithm *FR-FCM*. For *FR-FCM*, there's no

refinement of the memberships for the vertices, since we intend to deal with overlapping community.

The modularity we use is the same as [18], defined as $\frac{1}{2m}\Sigma_{i,j\in V}(A_{ij} - \frac{k_i k_j}{2m})\delta(c_i, c_j)$, where $A_{ij} = 1$ if i and j are connected, otherwise $A_{ij} = 0$, and $\delta(c_i, c_j) = 1$ if i and j belong to the same cluster, otherwise $\delta(c_i, c_j) = 0$.

4 Experiment

We conduct experiments on both synthetic and real world graphs including two benchmark graphs for community detection algorithm. The experiments ran on an Inter Core, 2 Quad CPU, 2.83GHz, 2GB machine running Windows 8 OS. The algorithms are implemented in C.

We use a batch of benchmark graphs [14] to evaluate the effectiveness of our method. The real-world benchmark graphs we use are Zachary's Karate Club data and American College Football data. We also test on the Email-URV data set, Wikipedia data set, and Facebook data set. They represent large online social network data. See [1] for detailed description of the data sets ,and results and analysis of overlapping community detection.

4.1 Analysis of Non-overlapping Community Detection

We compare our method to the algorithms of Girvan and Newman(*GN*) ([13,18]), one of the state-of-the-art algorithms in community detection. Modularity is first proposed in this algorithm. We also compare our method with *Walktrap* algorithm ([20]) and *InfoMap* algorithm ([22]), which has been shown to perform quite well for community detection (see [10]).

Table 1. Performance Comparison between *FR-EM*,*FR-KM* and *GN*

	KarateClub		AmericanFootball		EmailURV	
	modularity	running time	modularity	running time	modularity	running time
GN	0.4013	0.016	0.5976	1.014	0.5323	3193.532
Walktrap	0.3944	0.0000001	0.6015	0.015	0.5250	0.92
InfoMap	0.402038	0.015	0.599176	0.047000	0.521420	5.912000
FR-KM	0.417406	0.020000	0.601731	2.179000	0.542659	15.388000

Table 1 shows the performance of the algorithms. In this comparison, we use the normal two dimension *FR* algorithm with its iteration equal 400 for KarateClub and AmericanFootball data and 1000 for EmailURV data. The number of trials is set to 30. For all three graphs, our method produces partitions with highest modularity among the four algorithms. Although *Walktrap* and *InfoMap* are faster than our method and *GN* is faster than our method for smaller graphs, the running time of our method is still tolerable. As the size of graph becomes larger, our method becomes faster. If the number of clusters is known in advance, then the number of trial is 1 instead of 30 that we set. If so, our method takes much less time. *GN* is much slower for larger graphs. For the other two

real-world data sets, Wiki-Vote and Facebook, we are unable to make the comparison due to *GN*'s scalability, but we will show the running time of clustering these two graphs by our method.

Figure 1 shows the performance comparison between multiple dimension *FR-KM* and *GN*. We extend the normal two dimension FR algorithm to one dimension and three, four, five dimensions. We set the number of trials 30. For karate club data, the number of trial is equal to its number of vertices. We run each *FR-KM* with the number of iterations of *FR* changing from 100 to 2000 with interval 100. We find that the larger the number of iterations of *FR-KM*, the longer time it takes. However, the number of iterations of *FR* doesn't have decisive influence on the modularity. This suggests that there is no need to increase the number of iterations to get higher modularity. In terms of dimension, we find that for small graphs, projecting them to one dimension or three dimension may get higher modularity sometimes, but for large graphs, the two dimension *FR-KM* performs best. It's faster and clusters graph with higher modularity. That is why we shall adopt the normal two dimension *FR* in our algorithm when it comes to large graphs. *FR-KM* outperforms in both effectiveness and efficiency with large graphs compared with *GN*.

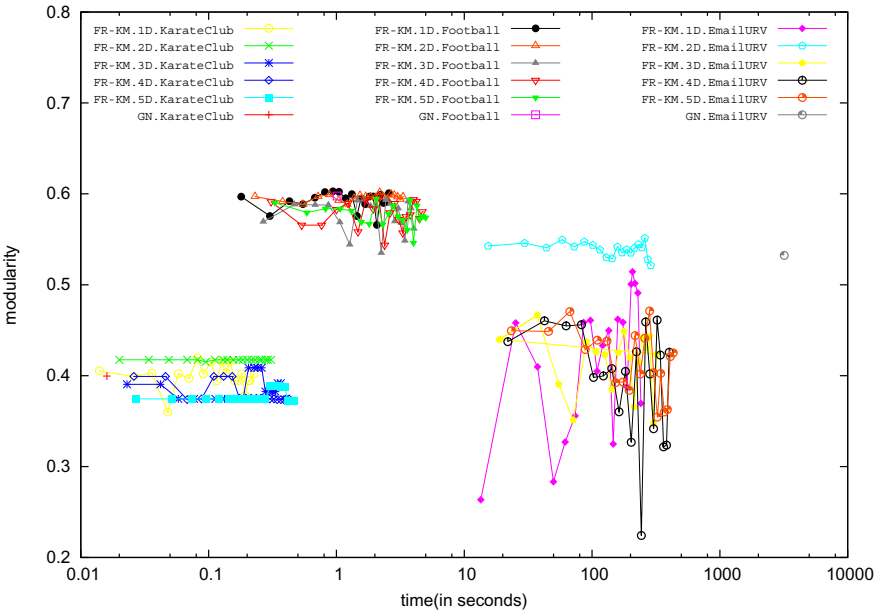
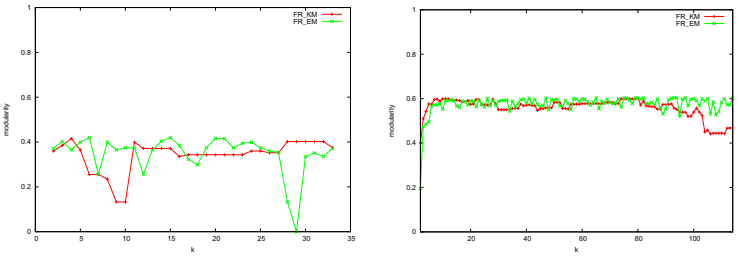


Fig. 1. Performance Comparison between multiple dimension *FR-KM* and *GN*

Figure 2 shows the modularity when the initial input number of clusters, k varies. The final number of clusters may be different from the values of k on X-axis here. Our method changes the number of clusters during cluster refinements,

which produces local optimum number of clusters. Therefore, we can see from the result that the trend of the line is horizontal in general. This suggests that even without knowing the number of clusters beforehand, we can find a local optimum around initial k value. This local maximum is probably the global optimum or close to the global optimum.



(a) Karate Club data (b) Football data

Fig. 2. Modularity for varying number of clusters

Figure 3 shows the running time for varying number of clusters for Email-URV data, Wiki-Vote data and Facebook data. For each data set, the time for projecting the graph onto Euclidean space is the same, but the clustering time differs. *KM* running time keeps the same in general as the initial number of clusters increases while *EM*'s running time linearly increases as the initial number of clusters increases. Compared with *KM*, *EM* takes much more time. The trends are similar among results for the three data sets.

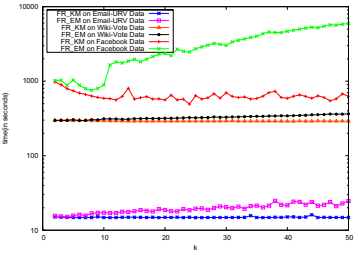


Fig. 3. Running time for varying number of clusters for Email-URV, Wiki-Vote, and Facebook data set

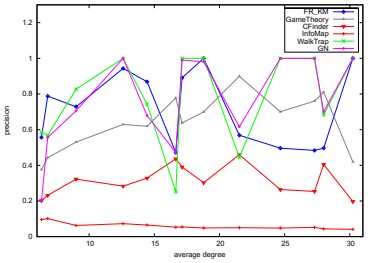


Fig. 4. Precision for varying average degree of synthetic graphs

We compare our method with *GN*, *InfoMap* and *WalkTrap* algorithm, and two other community detection algorithms, CFinder ([12]) and GameTheory algorithm ([5].) Figure 4 shows the precision achieved by the algorithms on the generated graphs with different average degrees. Since the community structures

are known, precision is obtained by counting the number of correctly clustered vertices. The results show that our method outperforms the CFinder, *GN* and *InfoMap*, and produces results comparable with GameTheory algorithm and *WalkTrap*. The reason for CFinder having the low precision may be that not every vertex in the graph are clustered. The clusters consists of 3-cliques only in our experiment. The reason for *InfoMap* having the low precision may be that the number of community this method detects is large and most of the communities are of small size. Many communities are of size of two vertices only.

5 Conclusions

In this paper, we propose a graph-layout based community detection algorithm. We use Fruchterman-Reingold algorithm to project the graph onto a Euclidean space and we cluster the vertices according to their Euclidean distance. Then we refer to the original graph information to refine the communities detected. We evaluate the effectiveness and efficiency on both real-world data and synthetic data. For disjoint community detection, the results show that *FR-KM* is more effective on both small graphs and large graphs than *GN*, and is much more efficient than *GN* on large networks. *FR-KM* is also more effective than *Walktrap* and *InfoMap* algorithms in terms of modularity. Compared with *GN*, *CFinder*, *InfoMap*, *WalkTrap* and Gametheory algorithms on the synthetic graphs with known communities in advance, our method is more effective than *GN* and *CFinder* and has good performance comparable with *WalkTrap* and Game Theory algorithm according to the results of precision. For overlapping detection, the result for Karate Club data shows that *FR-FCM* is reasonably effective.

References

1. https://dl.comp.nus.edu.sg/dspace/handle/1900.100/4144
2. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical society, Series B (1977)
3. Aslam, J.A., Pelekhov, E., Rus, D.: The star clustering algorithm for static and dynamic information organization. J. Graph Algorithms Appl. 8, 95–129 (2004)
4. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Kluwer Academic Publishers, Norwell (1981)
5. Chen, W., Liu, Z., Sun, X., Wang, Y.: A game-theoretic framework to identify overlapping communities in social networks. Data Min. Knowl. Discov. (2010)
6. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Phys. Rev. E 70, 066111 (2004)
7. Du, N., Wu, B., Pei, X., Wang, B., Xu, L.: Community detection in large-scale social networks. In: WebKDD/SNA-KDD, ACM (2007)
8. Eisen, M.B., Spellman, P.T., Brown, P.O., Botstein, D.: Cluster analysis and display of genome-wide expression patterns. Proc. Natl. Acad. Sci. U.S.A. (1998)
9. Etling, B., Kelly, J., Faris, R., John, P.: Mapping the arabic blogosphere: Politics, culture, and dissent. Berkman Center Research Publication (2006-06) (2009)
10. Fortunato, S., Castellano, C.: Community structure in graphs. In: Encyclopedia of Complexity and Systems Science, pp. 1141–1163 (2009)

11. Fruchterman, T.M.J., Reingold, E.M.: Graph drawing by force-directed placement. Softw. Pract. Exper. 21(11), 1129–1164 (1991)
12. Gergely Palla, I.F., Derenyi, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature (2005)
13. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proceedings of the National Academy of Sciences (2002)
14. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Physical Review E 78 (2008)
15. Lancichinetti, A., Radicchi, F., Ramasco, J.J., Fortunato, S.: Finding statistically significant communities in networks. CoRR (2010)
16. Lloyd, S.P.: Least squares quantization in pcm. IEEE Transactions on Information Theory 28, 129–137 (1982)
17. Macropol, K., Can, T., Singh, A.K.: Rrw: repeated random walks on genome-scale protein networks for local cluster discovery. BMC Bioinformatics (2009)
18. Newman, M., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E 69, 026113 (2004)
19. Newman, M.E.J.: Modularity and community structure in networks. Proceedings of the National Academy of Sciences 103(23), 8577–8582 (2006)
20. Pons, P., Latapy, M.: Computing communities in large networks using random walks. In: Yolum, p., Güngör, T., Gürgen, F., Özturan, C. (eds.) ISCIS 2005. LNCS, vol. 3733, pp. 284–293. Springer, Heidelberg (2005)
21. Reddy, P.K., Kitsuregawa, M., Sreekanth, P., Rao, S.S.: A graph based approach to extract a neighborhood customer community for collaborative filtering. In: Bhalla, S. (ed.) DNIS 2002. LNCS, vol. 2544, pp. 188–200. Springer, Heidelberg (2002)
22. Rosvall, M., Bergstrom, C.T.: Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences of the United States of America 105 (2008)
23. Schaeffer, S.E.: Graph clustering. Computer Science Review 1(1), 27–64 (2007)
24. van Dongen, S.: Graph Clustering by Flow Simulation. PhD thesis (2000)
25. Xie, J., Szymanski, B.K.: Towards linear time overlapping community detection in social networks. In: Tan, P.-N., Chawla, S., Ho, C.K., Bailey, J. (eds.) PAKDD 2012, Part II. LNCS, vol. 7302, pp. 25–36. Springer, Heidelberg (2012)