Fast Spectral Clustering in Large Scale Networks

CS 249 Current Topics in Data Structures – Course Project Report

Yunqi Guo
304776755
guoyunqi@gmail.com

Teghveer Sahni
204866389
tsahni001@ucla.edu

Chidambaram Muthappan
704774938
cmuthapp@cs.ucla.edu

ABSTRACT

Graph patitioning/clustering is an important problem that has a wide range of applications in various areas. Spectral clustering, one of the various partitioning approaches, has been shown to be a very effective clustering method which can be applied to common data mining problems such as community detection. In this project, we will apply the spectral clustering method to large scale networks for graph partitioning. More importantly, we will show how to greatly improve the overall runtime efficiency of the spectral method while upholding its high clustering quality by introducing a new framework. To evaluate the experiment, we compare our results with other state-of-the-art clustering methods and we find that our framework can achieve more accurate clusters in most cases.

KEYWORDS

Spectral Clustering, Social Network Analysis, Unsupervised Learning

ACM Reference format:
Yunqi Guo, Teghveer Sahni, and Chidambaram Muthappan. 2017. Fast Spectral Clustering in Large Scale Networks. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 5 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In recent years there has been a growing public fascination with the complex "connectedness" of modern society. This connectedness is found in many incarnations: in the rapid growth of the Internet and the Web, in the ease with which global communication now takes place, and in the ability of news and information as well as epidemics and financial crises to spread around the world with surprising speed and intensity. These are phenomena that involve networks, incentives, and the aggregate behavior of groups of people; they are based on the links that connect us and the ways in which each of our decisions can have subtle consequences for the outcomes of everyone else. In the most basic sense, a network is any collection of objects in which some pairs of these objects are connected by links. The typical size of large networks such as social network services, mobile phone networks or the web now counts in millions when not billions of nodes and these scales demand new methods to retrieve comprehensive information from their structure. Many analytical techniques have been proposed which have strived to help create a better understanding of information networks and their properties. Clustering groups of objects based on a certain proximity measure so that similar objects are in the same cluster, whereas dissimilar ones are in different clusters is one such technique. Graph partitioning/clustering is an important problem that has extensive applications in many areas.

The graph partitioning problem is NP-complete. However, many algorithms have been developed that find a reasonably good partition. Spectral partitioning methods are known to produce good partitions for a wide class of problems, and they are used quite extensively. However, these methods are very expensive since they require the computation of the eigenvector corresponding to the second smallest eigenvalue (Fiedler vector). In this project, we will be exploring methods that can effectively reduce the time and space complexity of spectral clustering on large networks without decreasing the accuracy. Our experiment results show that our framework can generate more accurate clusters than the state-of-the-art clustering methods like METIS.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 MOTIVATION

The goal of spectral clustering is to cluster data that is connected but not necessarily compact or clustered, within convex boundaries. The basic steps are as follows:

(1) project your data into R^n
(2) define an Affinity matrix A, using a Gaussian Kernel K or say just an Adjacency matrix (i.e. $A_{i,j} = \delta_{i,j}$)
(3) construct the Graph Laplacian from A(i.e. decide on a normalization)
(4) solve an Eigenvalue problem , such as $Lv = \lambda v$ (or a Generalized Eigenvalue problem $Lv = \lambda Dv$)
(5) select k eigenvectors $\{v_i, i = 1, ..., k\}$ corresponding to the k lowest (or highest) eigenvalues $\{\lambda_i, i = 1, ..., k\}$, to define a k-dimensional subspace $P^t LP$
(6) form clusters in this subspace using an algorithm such as k-means

Now solving an eigenvalue problem has a computational complexity of $O(n^3)$. In large networks where the number of nodes range anywhere between 25,000 to 2,000,000, this runtime is close to impossible. Also, the space complexity of Spectral Clustering is $O(n^2)$. Our main motivation is to bring this computational complexity to reasonable bounds.

3 RELATED WORK

Many algorithms that find community structures/clusters in graphs already exist. Clustering is another way to summarize information network and discover the underlying structures in data. This

class of algorithms partitions the objects of an information network into subsets (clusters) so that the objects in each subset share some common trait. In clustering, proximity between objects is often defined for the purpose of grouping "similar" objects into one cluster, while partitioning dissimilar ones far apart. Spectral graph clustering [14, 15] is a state-of-the-art method that does clustering very well on homogeneous networks. In the machine learning community, spectral clustering has been made popular by the works of [9, 11, 14, 16]. The spectral relaxation of standard Euclidean k-means objective function was discussed in [17]. The formulation was based on unweighted k-means and no connection was made to graph cuts. A recent paper [13] also discusses a connection between k-means and spectral clustering in the unweighted case; however, normalized cuts were not amenable to the analysis in [13]. The latter paper also discussed the issue of enforcing positive definiteness for clustering.

The concept of multilevel clustering is explored in many papers like [1, 6]. METIS is often cited as one of the best examples of a multi level graph clustering algorithm. However, METIS and Chaco [5] are both restricted to the Kernighan-Lin heuristic and force nearly equally sized clusters. Some recent work has considered hierarchical spectral methods [2] to improve efficiency, though this work is limited to stochastic matrices (and thus applies only to the normalized cut objective); as future work, it may be possible to compare with and extend/incorporate the methods discussed in this domain.

4 PROBLEM DEFINITION

In this paper, we are focusing on the undirected and unweighted graph $G(V, E)$, which is usually used to represent many different graph datasets such as the Facebook friend network and the co-author relation. In this graph, the number of Vertexes $|V| = n$. The goal of this model is to partition these n nodes to t clusters, so that the clustering results can reasonably fit the ground truth labels of the vertices.

5 METHODOLOGY

As illustrated in Figure 1, the fast spectral graph clustering framework consists of four major steps of operations:

(1) compressing the n nodes of the original graph G to k groups.
(2) building a new undirected weighted graph G' with these k groups of nodes.
(3) applying spectral clustering algorithm to the new graph G' which partitions G' to t different communities.
(4) decompressing the labeled k groups back to n nodes.

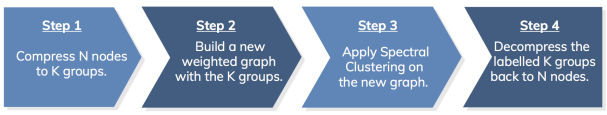

Figure 1: Fast Spectral Graph Clustering Framework

In the following parts of this section, we will introduce each step in detail.

5.1 Graph Compressing

One of the key factors of this fast spectral clustering method is the choice of the preprocessor in step one. Any of the various computationally cheaper graph partitioning algorithms that exist could be used in this step to group the vertices together into k communities. To compare the efficiency and accuracy of different compressing algorithms, we chose five well-known graph community detection methods, Fast Greedy[3], Info Map[7], Leading Eigenvector[10], Spin Glass algorithm[12], and Label Propagation algorithm[8]. The reason why we considered these algorithms is because the time complexities of these methods are much smaller than the basic spectral algorithm which has a runtime of $O(n^3)$.

As for the community number k, there are two factors that we need to take into consideration - the performance of the compressor and the influence it will have on the later steps. On one hand, each community detection algorithm has its 'best' choice of the number of clusters k to detect, which depends on the community modularity and runtime. On the other hand, the group number k will influence various performance metrics, including the accuracy and efficiency of the spectral method. If k is chosen to be too large, the spectral clustering method with $O(k^3)$ time complexity will take an extremely long time to composite the Laplacian matrix. If k is chosen to be too small, it will be hard for this model to show the advantage of the spectral clustering method. In summary, we need to find a trade-off between the runtime and performance when it comes to choosing different values for ks. In the experiment section, we will vary the parameter k for an algorithm that we choose as our preprocessor, and compare the performance using purity as our main metric.

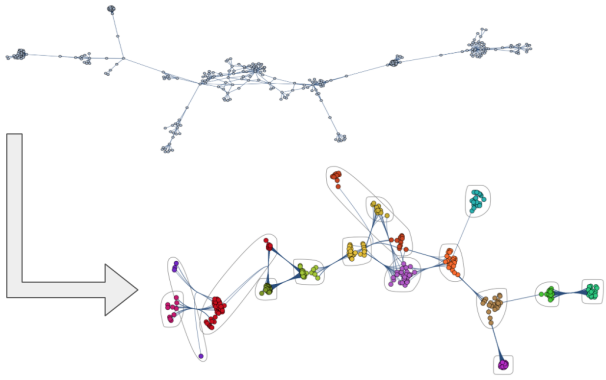

Figure 2: Step 1: Compress n nodes to k groups

After this step, the community structure we obtain will be $\{C_i | i = 1, \ldots, k\}$.

5.2 Constructing graph G'

The second step is to build a new weighted graph G' based on the k groups we find in step one. With our approach, we utilize the set of nodes in each group as new vertices. Hence, the new graph G' will be composed of the k vertices we found in step 1. As for the edges, the approach requires a weight function to quantify the distance between each pair of groups.

Distances between each pair of nodes in the original graph G is required for determining the weights between each pair of groups. The shortest path is the most wide used approach to representing this distance. We define D as the shortest path matrix, which is calculated by Dijkstra's Algorithm[4]. The time complexity of generating matrix D is $O(n^2)$.

We use the average to represent the group distance. Specifically, the group distance function is defined as

$$\Lambda\left(C_i, C_j\right) = \frac{\sum_{v_p \in C_i} \sum_{v_q \in C_j} D_{pq}}{|C_i| \times |C_j|}$$

We want pair of nodes with shorter distances to have higher weights assigned to them, which is why we take the reciprocal of the distance and make that the weight between a pair of vertices. The weight matrix is defined as:

$$W_{ij} = 1/\Lambda\left(C_i, C_j\right)$$

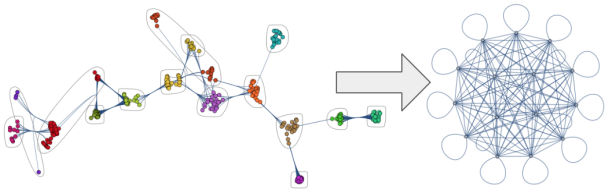

Figure 3: Step 2: Construct new weighted Graph

In some cases, zeros can appear in the distance matrix Λ. To avoid the division by zero problem, we add one to all the elements in matrix Λ:

$$W_{ij} = 1/(\Lambda\left(C_i, C_j\right) + 1)$$

After calculating all the edge weights between all pairs of k nodes in graph G', G' becomes a complete weighted graph as showed in Figure 3.

5.3 Spectral Clustering on G'

After the second step, the compressed graph we get is a complete weighted graph G' represented by the weighted adjacency matrix W. The goal of this step is to separate k vertexes to t communities using spectral clustering method. The algorithm is as showed in Algorithm 1.

The time complexity of this algorithm is the same as the time to compute the eigenvectors, which is $O(k^3)$.

5.4 Reverse mapping

The third step results in the k nodes of graph G' to be assigned a label that associates it with one of the t different communities. The decompressing part consists of using the community identification in G' to label all the nodes in the corresponding group in graph G. The runtime of this procedure is $O(n)$.

The overall time complexity of all four steps in our novel procedure includes three parts: the time complexity of the first step, $O(n^2)$ and $O(k^3)$.

Algorithm 1: Spectral Clustering on Graph G'

Input : weighted adjacency matrix W, community number t
Output: t-way partition of the input data

(1) Compute the similarity matrix A with elements:

$$a_{i,j} = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}}$$

(2) Compute the diagonal matrix D with elements:

$$d_i = \sum_{j=1}^{n} a_{i.j}$$

(3) Compute the normalized Laplacian matrix $\mathcal{L}$ of A:

$$\mathcal{L} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

(4) Select k eigenvectors $\{v_i, i = 1, ..., k\}$ corresponding to the k lowest (or highest) eigenvalues $\{\lambda_i, i = 1, ..., k\}$, to define a k-dimensional subspace $P^t LP$

(5) Form t clusters in this subspace using, say, k-means

6 EXPERIMENT

6.1 Dataset

First, we will start our discussion of the experimentation aspect of our project by describing the dataset we used in our experiments. We used the DBLP dataset which is a co-authorship network that is available to the public and can be found online. Lets call the graph expressed by this dataset G(V, E). The vertices V in the graph represent the ID of an author of an academic paper. 2 vertices have an edge between them if the 2 authors represented by the connecting vertices published at least 1 academic paper together. The data came with a list of communities and the nodes that are in each community. We decided to use the 5 largest communities as our ground truth. In total, the 5 largest communities that we worked with contained 89,063 edges and 24,493 nodes. As you can see, this is a very large and highly connected graph which would be a challenge for various graph clustering algorithms to work with.

6.2 Procedure

The first step in our experimentation phase was to process the data. The dataset did not require any cleaning. We did need to go through the dataset and determine the largest 5 communities. Once we found the communities, we went through all of the edges and removed an edge if both of the vertices connected by the edge did not exist in one of the 5 communities we had chosen to work with.

Once the data had been processed and in the format we wanted, we were ready to start doing some clustering. We first wanted to run spectral clustering alone on the data so that we could get an idea of the time our method would need to beat. When we ran the algorithm, we were not able to get any results. We let the algorithm run for about 10 hours until we decided to quit. The program at one point was taking 16GB of memory. This led us to believe that spectral clustering is not very efficient at finding clusters in this dataset which opened the door for our novel method to come in and improve the runtime.

We were not able to get any results from spectral clustering but still wanted to have some benchmark to which we could compare

our method to. After some research, we found METIS which is a software library that performs graph partitioning. The algorithms that are part of this package are built on multilevel recursive-bisection, multilevel k-way, and multi-constraint partitioning schemes that were developed by the creators of this package. One of the reasons why we used this software was because the developers of METIS claimed that it can achieve better accuracy and purity than spectral clustering. Hence, we decided that if METIS performs better than spectral clustering, and our method performs better than METIS, then we may be able to conclude that our framework is better than spectral clustering (in terms of purity).

When we ran METIS on the dataset(approximately 25,000 nodes), we were able to achieve a purity of .568. This became the number that we wanted to beat.

After this step, we incorporated METIS into the first step of the algorithm that we had developed. As a reminder, the first step of our method is to use a clustering algorithm to compress the N starting nodes into K groups/communities. METIS allowed us to specify the number of communities beforehand so we ran it a number of times with varying values for community. Below, we have a graph of our results (Figure 4.

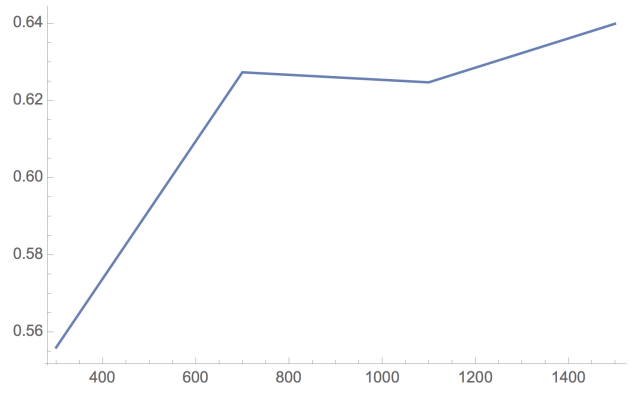

Figure 4: Purity with different k (x-axis: k, y-axis: Purity)

The X axis represents the value for "number of communities" that was passed as an argument to METIS, and the Y axis represents the purity that was achieved by our method when METIS was used at the first step with the corresponding argument value for community. The maximum purity that was obtained was .640 when the number of communities was set to 1500. As you can see, METIS performed better when incorporated into our framework as opposed to when it ran alone on the dataset (.640 purity compared to .568 purity). Since METIS performs better than spectral clustering according to the developers of METIS, the purity achieved by our method is theoretically better than the purity spectral clustering would have achieved if it had ran to completion. In addition, we have significantly improved the runtime of spectral clustering. Spectral clustering when applied to the dataset directly did not produce any results even after 10 hours, but when used with our framework at the 3rd step, it ran to completion in the matter of seconds. This worked because the number of nodes being fed into Spectral Clustering was decreased to k nodes by METIS, where k can be less than or equal to n which is the number of nodes we started with.

After this, we wanted to see what results we would obtain if we were to vary the algorithm used at the first step of our method. We did research on and chose 5 relatively cheap graph clustering algorithms that can group nodes into communities at the first step. We needed to make sure that these algorithms do not exceed $O(n^3)$ in runtime, because if the runtime of the algorithm exceeds $O(n^3)$, the total runtime of our novel method would exceed n^3, which would make it slower than spectral clustering. The 5 algorithms we chose are Fast Greedy Modularity Optimization Algorithm , InfoMAP Community Detection Algorithm, Leading Eigenvector Method, Spinglass, and Label Propagation. All of these algorithms were used on our DBLP dataset and produced communities of varying sizes which were then fed to spectral clustering which gave us our 5 clusters. Table 1 are the results of our experiments.

Table 1: Experiment result comparing our method with the other clustering algorithms. Purity 1 is the accuracy achieve by our clustering algorithm, Purity 2 is from the original clustering algorithm.

Algorithm	k	Runtime(s)	Purity 1	Purity 2
Fast Greedy	289	5.82	0.44	0.48
Info Map	1829	182.88	0.51	0.47
Leading Eigenvector	43	45.05	0.38	0.40
Spinglass	25	794.08	0.38	0.37
Label Propagation	2235	36.095	0.57	0.54

The 2nd column in this chart shows the number of communities that were produced by the clustering algorithm. The 3rd column shows the time taken by our method from step one where we apply the algorithm all the way to step 4 where we decompress the labeled k groups back to n nodes. All of these times are great compared to the 10+ hours Spectral Clustering on its own would take.

The 4th column shows the purity achieved by our method. The last column shows the purity achieved when the algorithm was used on its own, separate from our framework. The InfoMAP and Label Propagation algorithm performed the closest to the technique where METIS was used at the first step. Here we have another chart (Figure 5) that compares the purity achieved by our technique to the purity achieved if we were to use the algorithm on its own.

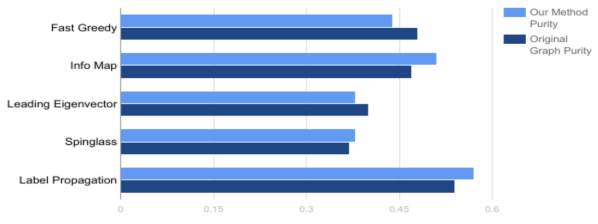

Figure 5: Purity comparison

One point we noticed from these experiments and results was that the purity achieved by some of the clustering methods was

higher when incorporated into our framework as opposed to when the method was used on its own. InfoMap, Spinglass, and Label Propagation show this in addition to METIS where we saw a maximum of .072 gain in purity. Overall, our experimentation process required a fair amount of research and implementation. In the end, we were able to produce meaningful results.

7 CONCLUSION

In conclusion, we were able to achieve the goals that we set out to achieve in this project. Our goal was to improve the time and space complexity of spectral clustering, while maintaining or even improving the accuracy of the algorithm on large scale networks. The time complexity was decreased from $O(N^3)$ to $O(n^3)$ meaning that we were able to create a method where the initial number of nodes that were being fed into the spectral clustering algorithm was decreased. In addition, we saw an improvement in space complexity due to the fact that spectral clustering is working with less nodes.

One of the main reasons our framework achieved the results that it did was due to the nature of the algorithm which was to reduce the number of nodes before applying spectral clustering. The first step we perform is that we take the N nodes and apply a cheap clustering algorithm that outputs some number of communities. By cheap, we mean that the algorithm used should not exceed $O(N^3)$ which is the runtime of spectral clustering. If the runtime of a clustering algorithm we use in the first step exceeds this runtime, this would end up making the overall runtime of our method more than $O(N^3)$. Hence, our goal of improving the runtime of spectral clustering would not be possible. Once the clustering algorithm in step 1 has finished we make a new graph with these nodes and run spectral clustering on it. This first step is one of the main reasons why our method achieved good results.

Another result that we noticed during the course of the project was that our novel method can be used to improve other various clustering algorithms. A clustering algorithm that finds communities in graphs can be easily incorporated into our framework(specifically, the algorithm will be used at the first step). Doing so may achieve better results than if the algorithm was just used alone. As talked about in our experimentation section, we tested many different community finding algorithms such as Fast Greedy and Label Propagation by using them at the first step in our methodology. For 4 out of the 6 algorithms we tested, we achieved a higher purity when the algorithm was used with our framework as opposed to being used alone. One possible reason why a higher purity was not achieved for the some of the algorithms was because of the nature of the clustering algorithm itself. Maybe only algorithms that operate in a certain way can work well with the method we developed. There are many other possible reasons why a higher purity was not achieved for all of the algorithms. We recognize that the results we achieved may not be convincing enough to show that our method really does improve the purity of the communities for various graph clustering algorithms. Further experiments and research needs to be done to find out what adjustments can be made to our framework so that many other clustering algorithms besides spectral clustering can be improved.

We plan to take our project further beyond this course. One step we want to take is to improve our system and apply it on other datasets. This will allow us to further explore the power of our framework and check whether or not it performs better on certain types of datasets. Overall, this project enabled us to expand our knowledge in data mining and was a great learning experience.

TASK DISTRIBUTION FORM

Table 2: Task distribution

Task	People
Collecting and processing data	Teghveer
Implementing Part 1	Chidambaram
Implementing Part 2	Yunqi
Evaluating and comparing algorithms	All
Writing report	All

REFERENCES

[1] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[2] Chakra Chennubhotla and Allan D Jepson. 2005. Hierarchical eigensolver for transition matrices in spectral methods. In *Advances in Neural Information Processing Systems*. 273–280.

[3] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical review E* 70, 6 (2004), 066111.

[4] Edsger W Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* 1, 1 (1959), 269–271.

[5] Bruce Hendrickson and Robert W Leland. 1995. A Multi-Level Algorithm For Partitioning Graphs. *SC* 95, 28 (1995).

[6] George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing* 20, 1 (1998), 359–392.

[7] Andrea Lancichinetti and Santo Fortunato. 2009. Community detection algorithms: a comparative analysis. *Physical review E* 80, 5 (2009), 056117.

[8] Xin Liu and Tsuyoshi Murata. 2010. Advanced modularity-specialized label propagation algorithm for detecting communities in networks. *Physica A: Statistical Mechanics and its Applications* 389, 7 (2010), 1493–1500.

[9] Marina Meila and Jianbo Shi. 2001. A random walks view of spectral segmentation. (2001).

[10] Mark EJ Newman. 2006. Modularity and community structure in networks. *Proceedings of the national academy of sciences* 103, 23 (2006), 8577–8582.

[11] Andrew Y Ng, Michael I Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*. 849–856.

[12] Jörg Reichardt and Stefan Bornholdt. 2006. Statistical mechanics of community detection. *Physical Review E* 74, 1 (2006), 016110.

[13] Volker Roth, Julian Laub, Motoaki Kawanabe, and Joachim M Buhmann. 2003. Optimal cluster preserving embedding of nonmetric proximity data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25, 12 (2003), 1540–1551.

[14] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence* 22, 8 (2000), 888–905.

[15] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17, 4 (2007), 395–416.

[16] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17, 4 (2007), 395–416.

[17] Hongyuan Zha, Xiaofeng He, Chris Ding, Ming Gu, and Horst D Simon. 2002. Spectral relaxation for k-means clustering. In *Advances in neural information processing systems*. 1057–1064.