CS-E4600 - Algorithmic Methods of Data Mining project report

Julius Hietala, Jarkko Tulensalo

December 2019

1 Introduction

Graphs can be used to model many different types of relations or processes. Examples include social networks, computer networks, biological networks, power lines, and citation networks [1]. Graph partitioning is a related and important task, where the goal is to find similar communities within such graphs using different algorithmic methods.

In this work, we explored different graph partitioning methods by partitioning five different graph data sets from the Stanford Network Analysis Project (SNAP, http://snap.stanford.edu/data/index.html) as a part of the CS-E4600 Fall 2019 course (Aalto University) programming project. To evaluate performance and further motivate exploring different methods, there was also a competition taking place between students where the highest ranked teams received extra points. Further motivation for the problem, the formal problem definition, and the definitions for the graph data sets will be introduced in the next section.

2 Background

Graph partitioning is motivated by many real world applications, where the goal is to find communities inside graphs where the members of the found communities have similar characteristics. An important example of such a problem would be for example the task of partitioning a graph of social media users into different communities, and then utilizing such information in advertisement targeting, personalizing shown content, or making suggestions of new connections within the platform.

In this work, the given graphs to be partitioned were the following: *ca-GrQc, Oregon-1, roadNet-CA, soc-Epinions1, and web-NotreDame.* The nature of the graphs vary, and more information about them can be found from the Stanford Network Analysis Project website (http://snap.stanford.edu/data/index.html).

The graphs are represented as undirected graphs $G = (V, E)$, where V is the set of vertices and E is the set of edges in G. The goal of the project is to

partition the set of vertices V into k communities, given some positive integer k, where $\bigcup_{i=1}^{k} V_i = V$ and $V_i \cap V_j = \emptyset$ for all $i \neq j$. Also the graphs should be partitioned such that the objective $\phi(V_1, ..., V_k) = \sum_{i=1}^{k} \frac{|E_*(V_i, \overline{V_i}|}{|V_i|}$ is as small as possible, where $E_*(S,T) = \{(u,v) \in E, |u \in S, v \in T\}$, $S, T \subseteq V$, $S \cap T = \emptyset$, and $\overline{V_i} = V \setminus V_i$.

The objective is also called *ratio cut*. In essence minimizing ratio cut tries to achieve minimal amount of edges between the partitions, while making the partitions balanced in size compared to each other.

The original graphs were not necessarily connected i.e. that there is a path between any two given vertices in the graph, but the CS-E4600 course staff provided the largest connected components instead to be partitioned. Statistics of the graphs are provided in Figure 1. Figure 2 shows a visualisation of the graph ca-GrQc. Based on the visualisation it is easy to see that no trivial solutions are available just by examining the graph due to the graph size, calling for more advanced partitioning methods.

Figure 1: Datasets

Graphs			
graph name	#vertices (n)	#edges	value of k
ca-GrQc	4 158	13 428	2
Oregon-1	10 670	22 002	5
soc-Epinions1	75 877	405 739	10
web-NotreDame	325 729	1 117 563	20
roadNet-CA	1 957 027	2 760 388	50

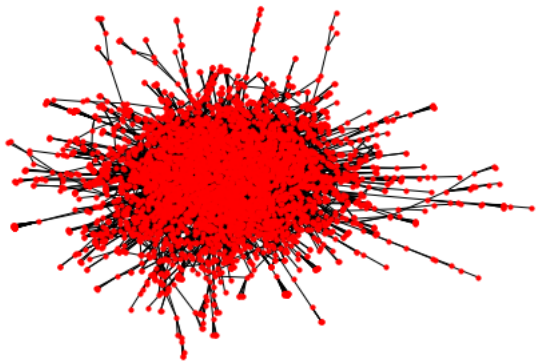

Figure 2: An example of a given graph: ca-GrQc.

3 Methods and algorithms

The method through which we decided to explore the graph partitioning problem and that was also suggested by the course staff is spectral graph theory. In general, spectral graph theory involves calculating and using the eigenvectors of matrices, that are related to the structure of the graph. A common starting point in acquiring such a matrix is to form the adjacency matrix A, where the matrix entries correspond to the edges of the graph. Given a graph with n vertices, the adjacency matrix would be of size n x n, and the individual rows and columns can be seen as representing the vertices [4].

The most common matrix that is based on the adjacency matrix and that we will also be using is the *Laplacian matrix*. The Laplacian matrix can be denoted by $L = D - A$, where A is the adjacency matrix described above, and D is the degree matrix of the graph. The degree matrix is an n x n diagonal matrix where the diagonal entries correspond to the degree of a particular vertex. The degree of a vertex is the number of other vertices that the vertex is connected to. Another popular version of the Laplacian matrix is the *Normalised laplacian matrix* $L = I - D^{-\frac{1}{2}}AD^{\frac{1}{2}}$, where I is the identity matrix. In literature names for these matrices are used loosely for different versions of the Laplacian, so the reader should always check the exact definition [5].

A basic scheme in spectral partitioning is to use a number of the eigenvectors corresponding to the smallest real eigenvalues of the different matrices related to the graph, that will hopefully give a meaningful embedding of the graph. This embedding can then be partitioned into communities with some well known partitioning algorithms such as k-means. In the ideal scenario (also in terms of our objective ratio cut), the graph would consist of k disconnected components. In that case, the eigenvalue 0 of the Laplacian matrix would have multiplicity of k and the eigenspace would be spanned by indicator vectors of the graph components, thus the partiotions for vertices y_i could just be "read" based on the indication value i.e. by checking which of the k eigenvectors has a value of 1 in the ith entry (only one will). In reality, graphs usually do not have such clearly defined components. Even if the graph does not have completely separated communities, the approach still works approximately, and the problem can be considered as clustering euclidean points defined by the eigenvectors. Also, the amount of "more" clearly defined communities in the graph might be different than the amount of clusterings desired (such as in this project), calling for the use of more or less eigenvector components in clustering than just an amount of desired k of them (we will use m to denote this quantity). Even though such approaches have proven empirically very useful, many practical algorithms still lack proofs that they will guarantee a meaningful partitionings [2]. Therefore we will be applying grid search to find the approximately best value for m.

During this project, we explored three different spectral graph partitioning methods. The first and the most simple one utilizes the *unnormalised Laplacian* matrix L described earlier. We then go on to find m eigenvectors of the matrix that correspond to the m smallest eigenvalues. We then stack these eigenvectors as columns of a new n x m matrix, and treat each row of that matrix as data

points y_i that are representations of the vertices (the ith vertex in the adjacency matrix) to be clustered, where $i = 0, ..., n$. We then use the k-means algorithm [7] to cluster these data points into k clusters (as given per graph in Figure 1).

The second algorithm we explored is similar in nature to the first one, but instead of the finding the eigenvectors of the laplacian matrix L, we find the eigenvectors of the *generalised eigenproblem* $Lu = \lambda Du$ and again stacking the m eigenvectors corresponding to the m smallest real valued eigenvalues as the columns of a new n x m matrix. Again, this matrix can be partitioned into k parts using k-means [3].

The third and last algorithm we explored for the graph partitioning problem is also very similar to the two previous ones, but with a key difference that now the matrix for which we find eigenvectors is the *normalised laplacian matrix* $L = I - D^{-\frac{1}{2}}AD^{\frac{1}{2}}$. After again creating the n x m matrix where the columns are the m eigenvectors that correspond to the m smallest eigenvalues, we now normalize each row to have a norm 1 before proceeding to the partitioning step with k-means [2].

In practice, all of these algorithms were implemented in the *python* programming language. Many readily available software solutions would have been available directly for the graph partitioning problem, but the idea was to implement our own. On top of regular python code, the main libraries that we used (and also verified with the teaching assistants that they were appropriate to use) were *scipy* (https://www.scipy.org/), *networkx* (https://networkx.github.io/), and *skicit-learn* (https://scikit-learn.org/stable/). Networkx was mainly used for abstracting the graphs into an easy to use form, whereas scipy was utilized for calculating the eigenvectors of the different Laplacian matrices. Finally, skicit-learn was used for the k-means clustering. We ran k-means with the default settings of skicit-learn k-means and utilizing k-means++ initialization [6]. Also, as a practical consideration, we computed the eigenvectors corresponding to 100 smallest eigenvalues for each graph and algorithm type and saved them as files. This way it was easier to run new partition simulations without the need for new eigenvector computations. Code for the project has been attached in the project submission. The precomputed eigenvectors are not included due to the large file size, but code for producing them is included. Code can also be found from https://github.com/hietalajulius/amdm-project-2019.

The biggest challenge implementation wise was that some of the graphs were very large (in particular web-NotreDame and roadNet-CA), thus making the calculation of the eigenvectors challenging. On the other hand, those graphs were very sparse so we were able to use the *scipy.sparse* (https://docs.scipy.org/doc/scipy/reference/sparse.html) package for finding the eigenvectors of those large matrices. The format in which *scipy.sparse* handles the matrices is called *CSR* (or *Compressed sparse row*) [8], where the matrix is stored as three one dimensional arrays. This format allows for quick matrix-vector multiplications, thus making finding the eigenvectors feasible.

4 Experiments and Results

We ran experiments on each of the three variants of the spectral graph partitioning algorithms for four smaller graphs by using different number eigenvector components ranging between $m = 1$ and $m = 100$, where k is the number desired partitions. We ran the experiments multiple times in order to improve the competition results. It was computationally expensive to run the experiments especially on the larger graphs which is why we focused on trying to reduce the total simulation time in order to run more experiments.

We found a few factors where we could reduce the total simulation time. Firstly, using sparse matrices and loading precalculated eigenvectors from a file for each graph reduced the simulation time significantly, as explained in previous section.

Secondly, we narrowed down to experimenting with a smaller range of eigenvector components $m = k$ and $m = k + 50$, where k is the number desired partitions. We noticed that the ratio cut objective did not improve after about $k+50$ simulations. However, we only ran the experiment for RoadNet-CA graph between $m = k$ and $m = k + 15$ since a single run took 1-2 hours which meant a total running time of 2-3 days with tests on using 15 different amounts of eigenvector components alone.

The proposed spectral partitioning algorithms performed relatively well in the task of graph partitioning for all five different graph data sets. Figure 8 shows the smallest ratio cut achieved by us for each graph. The table also shows which version algorithm (*unnormalised Laplacian*, *normalised Laplacian*, *generalised eigenproblem*) found the solution and the number of optimal eigenvector components. The (*) sign shows where multiple lowest scores were found. Below we describe the result per graph. Below, we also present graphs which show the ratio cut with three spectral partitioning algorithms for each graph between $m = k$ and $m = k + 50$ number of eigenvector components. For RoadNet-CA, we present the graph with $m = k + 5$ and $m = k + 15$ number of eigenvector components.

4.1 ca-GrQc

The goal was to partition the ca-GrQc graph into $k = 2$ partitions while minimizing the ratio cut. The algorithms based on the generalised eigenproblem and the normalised Laplacian both performed best for the ca-GrQc graph. The algorithm based on the generalised eigenproblem achieved a score of 0.0627 when using $m = 4$ eigenvector components. The algorithm based in the unnormalised Laplacian achieved the same score with (multiple, smallest) $m = 9$ eigenvector components. Running k-means clustering for the eigenvector embedding with identical settings produced identical results, leaving us speculating about a global optimum. Figure 3 shows the performance of all three different variants of the algorithm with different amounts of eigenvector components.

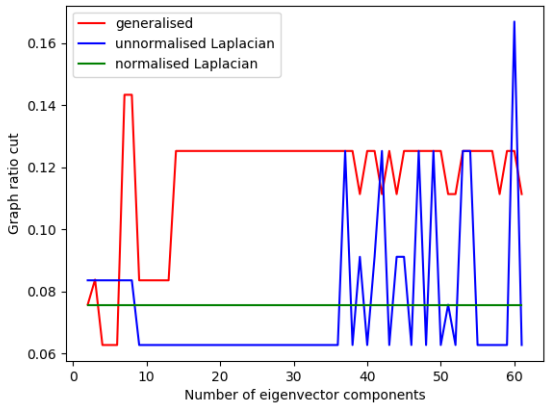

Figure 3: Ratio cut for ca-GrQc graph with different number of eigenvector components and different algorithms

4.2 Oregon-1

With the Oregon-1 graph, the task was to partition it into $k = 5$ partitions while minimizing the ratio cut. As with the ca-GrQc graph, the algorithms based on the generalised eigenproblem and the normalised Laplacian both performed best for the Oregon-1 graph. The algorithm based on the generalised eigenproblem achieved a score of 0.6004 when using $m = 22$ eigenvector components. The algorithm based on the unnormalised Laplacian achieved the same score with (multiple, smallest) $m = 35$ eigenvector components. Running k-means clustering for the eigenvector embedding with identical settings again produced identical results. Figure 4 shows the performance of all three different variants of the algorithm with different amounts of eigenvector components.

4.3 soc-Epinions1

The soc-Epinions1 was already a bit larger than the first two graphs, and the goal was to partition it into $k = 10$ partitions while again minimizing the ratio cut. The algorithm utilizing the unnormalised Laplacian found the smallest ratio cut of 0.5475 for this graph, using 64 eigenvector components. Running k-means clustering for the eigenvector embedding with identical settings did not produce identical results, leaving some room for improvement regarding k-means initialization. Figure 5 shows the performance of the variants of the algorithms based on the unnormalized Laplacian and the generalized eigenproblem with different amounts of eigenvector components. The performance plot of the algorithm based on the normalised Laplacian is left out since the scores were worse by a factor compared to the other algorithms.

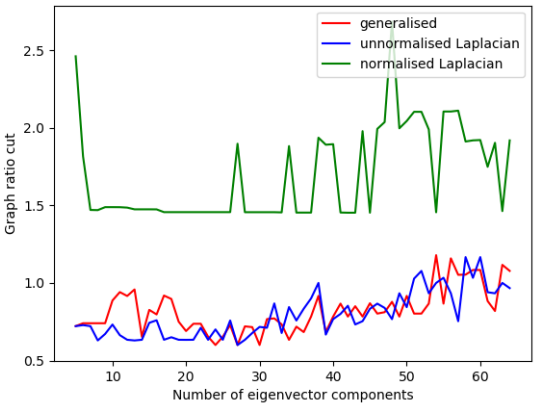

Figure 4: Ratio cut for Oregon-1 graph with different number of eigenvector components and different algorithms

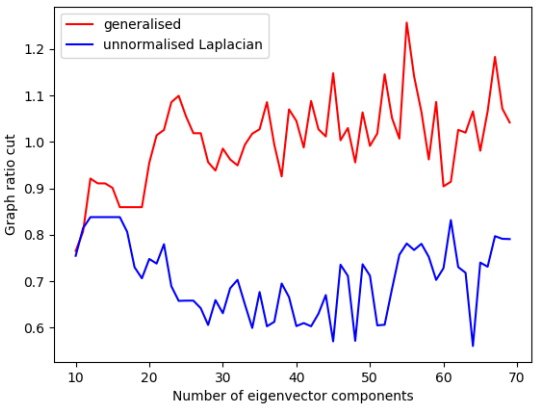

Figure 5: Ratio cut for soc-Epinions1 graph with different number of eigenvector components and different algorithms. The ratio cut with normalised Laplacian was higher by a factor why it was left out the figure.

4.4 web-NotreDame

The web-NotreDame was again a factor larger than the soc-Epinions1 graph, and the task was to find $k = 20$ partitions while minimizing the ratio cut. As with the previous graph, the algorithm utilizing the unnormalised Laplacian found the smallest ratio cut of 0.0249 for the graph, with $m = 22$ eigenvector components. As with the previous graph, running k-means clustering for the

eigenvector embedding with identical settings did not produce identical results, leaving room for improvement regarding k-means initialization. Figure 6 shows the performance of all three different variants of the algorithm with different amounts of eigenvector components.

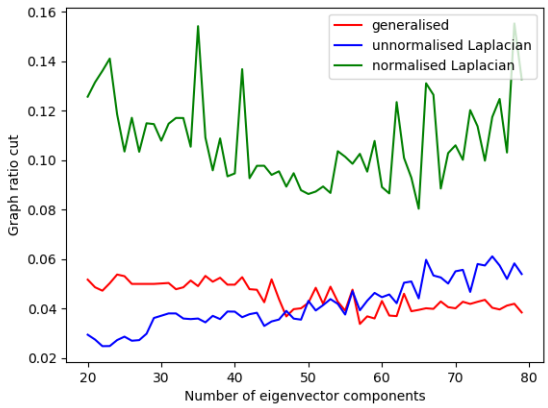

Figure 6: Ratio cut for web-NotreDame graph with different number of eigenvector components and different algorithms

4.5 roadNet-CA

roadNet-CA was the largest of all the graphs and the goal was to split it into $k = 50$ partitions, again also minimizing the ratio cut. As with the other two larger graphs, the algorithm utilizing the unnormalised Laplacian found the smallest ratio cut of 0.2492 for the graph using $m = 60$ eigenvector components. As with the two previous larger graphs, running k-means clustering for the eigenvector embedding with identical settings did not produce identical results from simulation to simulation, meaning that even better results could have been available only by running the experiments more times. Figure 7 shows the performance of all three different variants of the algorithm with different amounts of eigenvector components.

5 Conclusion

In this report, we studied how spectral graph partitioning algorithms worked for five different graphs, where ratio cut was the objective function. We found that the partitioning algorithm based on the generalised eigenproblem of the Laplacian performed best for small graphs and the algorithm based on the unnormalised Laplacian performed best for the larger graphs.

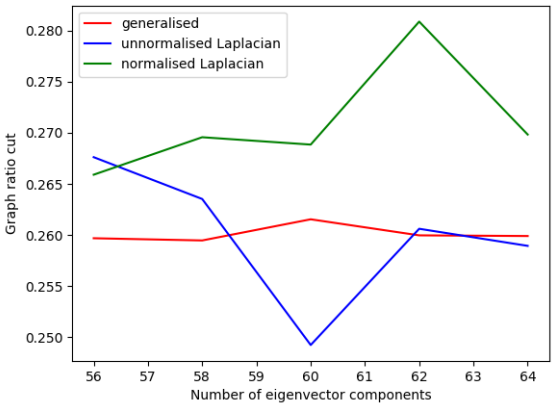

Figure 7: Ratio cut for roadNet-CA graph with different number of eigenvector components and different algorithms. Due the significantly larger graph size, we were The figure shows the ratio cut using the eigenvectors between 56 and 64 with every second value.

Figure 8: Smallest ratio cut for each graph. There were multiple optimal solutions for the first two graphs.

Graphs				
graph name	algorithm	#clusters (k)	#eigenvector components (m)	ratio cut
ca-GrQc	generalised/unnormalised	2	4/9*	0.0627
Oregon-1	generalised/unnormalised	5	22/35*	0.6004
soc-Epinions1	unnormalised	10	64	0.5475
web-NotreDame	unnormalised	20	22	0.0249
roadNet-CA	unnormalised	50	60	0.2492

We experimented with different number of eigenvector components with each algorithm which showed that changing the number of components had a clear effect on finding a smaller ratio cut of a graph.

A competition was also organized among students in finding the lowest ratio cuts of the graphs. Our results performed very well in the competition where we placed in shared first place in two of the smallest graph competition and within top 10 in the three larger graph competitions.

We used k-means algorithm to cluster the spectral eigenvector components into graph partitions. We found out that running the same algorithm twice did not always lead to same ratio cut as k-means clustering is not guaranteed on finding the global optimum. In this study, we did not experiment with different k-means initialisation algorithms which could improve the results even more.

One of the graphs, RoadNet-CA was exponentially more computationally expensive compared to the other graphs. In the given time, we were limited in the amount of experiments we could do with that graph. With more time, we could have run some more extensive tests on the amount of eigenvector components.

6 Competition results

We finished within the top 10 in all five competitions. We placed in shared first place in the ca-GrQc and Oregon-1 competition with a dozen other groups. We placed seventh in soc-Epinions1, fourth in web-Notredame and eight in roadNet-CA. We could see that working with the larger graphs, especially roadNet-CA was highly computationally demanding, where more usage of more powerful CPU or with more time we could run more experiments to try find lower ratio cuts. However, in the given time we are quite happy with our performance.

Figure 9: Performance in the competition. Parenthesis shows the position on the shared first place.

Graphs			
Competition	reported ratio cut	#placement	#1 ratio cut
ca-GrQc	0.0627	1 (6)	0.0627
Oregon-1	0.6004	1 (9)	0.6004
soc-Epinions1	0.5475	7	0.5197
web-NotreDame	0.0249	4	0.0236
roadNet-CA	0.2492	8	0.2076

7 Team member contributions, information, and collaboration with others

We contributed to the project with equal share. Collaboration was fluent and the group was happy with the process and results.

7.1 Julius Hietala (317447, julius.hietala@aalto.fi)

- Researching algorithm options
- Researching implementation details, especially sparse matrix solutions for the larger graphs
- Building the initial versions of the three different algorithms
- Computing 100 eigenvectors for each graph and algorithm type and providing them as files
- Report: Sections 1-3

7.2 Jarkko Tulensalo (84071T, jarkko.tulensalo@aalto.fi)

- Initial testing on basic spectral graph and graph manipulation
- Implementation, improvement and running experiments on different algorithms and number of eigenvector components
- Running specific experiments to improve our competition score
- Experiment and result visualisation
- Report: Sections 4-6

7.3 Collaboration with other teams

We also had some informal discussions with the team *TBD* about how many eigenvectors us and them had used for achieving competitive results.

References

[1] Athanassios Kintsakis. Mcl clustering of sparse graphs, 2018. [Online; accessed 6-December-2019].

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

[3] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Departmental Papers (CIS)*, page 107, 2000.

[4] Daniel Spielman. Spectral graph theory. *Lecture Notes, Yale University*, pages 740–0776, 2009.

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

[6] Wikipedia contributors. K-means++ — Wikipedia, the free encyclopedia, 2019. [Online; accessed 6-December-2019].

[7] Wikipedia contributors. K-means clustering — Wikipedia, the free encyclopedia, 2019. [Online; accessed 5-December-2019].

[8] Wikipedia contributors. Sparse matrix — Wikipedia, the free encyclopedia, 2019. [Online; accessed 6-December-2019].