Project Report for Comparative Analysis on Community Detection

Rohan Khopkar, Yiru Wang, Yachen Qin

1. Introduction

Community detection is a well-defined problem for understanding and extracting meaning from complex networks. We hope to subdivide network data into non-overlapping communities and extract meaning that could be applied to areas such as recommendation systems. In our project, *email-core data, dblp-data(with ground truth); facebook data and biosnap data(without ground truth)* are analyzed with several algorithms to reveal structures and gain new insights from these data.

We mainly chose 4 algorithms to build our models.

- The **Girvan-Newman algorithm** was chosen because it is a fundamental approach that has been well-defined and documented. It introduces the methodology of edge- removal which several, more novel approaches have adopted. The limitation of this algorithm is that it becomes very time-consuming on large-scale networks. We will analyze these limitations and how other algorithms overcome them in the later.
- The **Clauset-Newman Moore** algorithm operates on different principles from that of Girvan and Newman (GN), but gives qualitatively similar results and runs far more quickly, and it can deal with super large dataset.
- The **Louvain Modularity** algorithm is a modularity optimization algorithm. Modularity is by far the most used and best-known quality function. It represents one of the first attempts to achieve a first principle understanding of the clustering problem. From the paper we read, this method owns fast speed and relatively high outcome quality, which has been mainly confirmed in our project.
- The **Walktrap** algorithm was chosen because it has a clear advantage in term of quality of the computed partition and presents good quality for complex networks. We experienced this advantage in our project, especially in the email dataset.

2. The existing methods and related work

Community detection has become a core problem in describing the topologies of many kinds of systems from biological to social networks, and many different algorithms have been proposed to

address it. Many community detection algorithms have been developed to uncover the mesoscopic properties of complex networks. However how good an algorithm is, in terms of accuracy and computing time, remains still open. Testing algorithms on real-world network has certain restrictions which made their insights potentially biased: the networks are usually small, and the underlying communities are not defined objectively.

Up to now, there are a variety of algorithms, which could be generally classified into 5 kinds---- Link removal methods, Agglomerative methods, Maximizing modularity methods, Spectral Analysis methods and other methods.

- Link Removal methods remove some(or all) links of a network until it is no longer connected or meets certain requirements to extract the partition into communities, and Girvan Newman method is just one of the representative.
- Agglomerative Methods is to join or agglomerate network in larger groups from bottom to up.
- Maximizing modularity methods introduce the concept of modularity to achieve better partitioning, including Greedy Algorithm, Extremal Optimization, Simulated Annealing Methods, Information Theoretic Approach, and Louvain Algorithm is also in this list.
- Spectral Analysis methods are inherently global methods based on the eigenvectors of matrix representations of network structure.
- There are also some other typical methods addressing the problem from other perspectives, such as Clustering and Curvature, Random Walk based methods and Q-potts model.

3. Our methods

We use 4 methods to do community detection, including Girvan Neumann algorithm, The Clauset-Newman Moore method, the Louvain algorithm and Walktrap algorithm.

3.1. Girvan Neumann algorithm

A fundamental method for community detection via edge removal is the Girvan Neumann algorithm. This algorithm generalizes the "betweenness centrality" metric for a graph proposed by Freeman(1977) to a metric for edges of a network. The "betweenness" of an edge is essentially the number of shortest paths between vertices that pass through a given edge. where σ_{st} is the total

The edge betweenness for edge e:

$$\sum_{s,t \neq v} \frac{\sigma_{st}(e)}{\sigma_{st}},$$

number of shortest paths from node s to node t and σ_{st} (e) is the number of those paths that pass through edge e.

Then, by iteratively removing the edge with the highest betweenness and recalculating the edge betweenness for the remaining edges, we can remove every single edge of the network. From this edge removal, community structures can be identified because the shortest paths between vertices in two different communities will all contain certain edges connecting the communities. Thus, edges connecting different communities will likely have a high betweenness. Removing these edges reveals distinct community structures.

This betweenness calculation only ranks edges and does not tell the user which edges to remove for precise community detection. To find the optimal number of edge removals, a metric - modularity is introduced.

Modularity quantifies the quality of a community assignment by measuring how dense the connections are compared to what they would be in a particular type of random network.

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

There are two mathematical definitions of modularity. Here is the one used in the original paper on the Louvain method.

Here, A is the usual adjacency matrix, $k_i = \sum^{j} A_{ij}$ is the total link weight penetrating node i, and m= $\frac{1}{2}\sum^{ij} A_{ij}$ is the total link weight in the network overall. The Kronecker delta $\delta(ci,cj)$ is 1 when nodes i and j are assigned to the same community and 0 otherwise. Consider one of the terms in the sum. Remember that ki is the total link weight penetrating node i, and note that $\frac{k_i}{2m}$ is the average fraction of this weight that would be assigned to node j, if node i assigned its link weight randomly to other nodes in proportion to their own link weights. Then, $Aij - \frac{k_i k_j}{2m}$ measures how strongly nodes i and j are in the real network, compared to how strongly connected we would expect them to be in a random network of the type described above.

Using this modularity metric, an optimal edge removal can be accomplished and communities can be identified from a complex network.

The removal of each edge requires a calculation of the edge betweenness for each node of the network. This results in a time complexity of O(N) for a single calculation for all nodes, and O(N^2) for all edge removals. This quadratic run-time works well for networks with a low number of vertices and edges, but we expect it to become less optimal in performance for large, non-sparse networks. This algorithm will serve as a fundamental baseline for our study on community detection networks, and we will employ several other methods to improve the efficiency and robustness of the Girvan-Neumann algorithm.

3.2. Clauset-Newman Moore algorithm

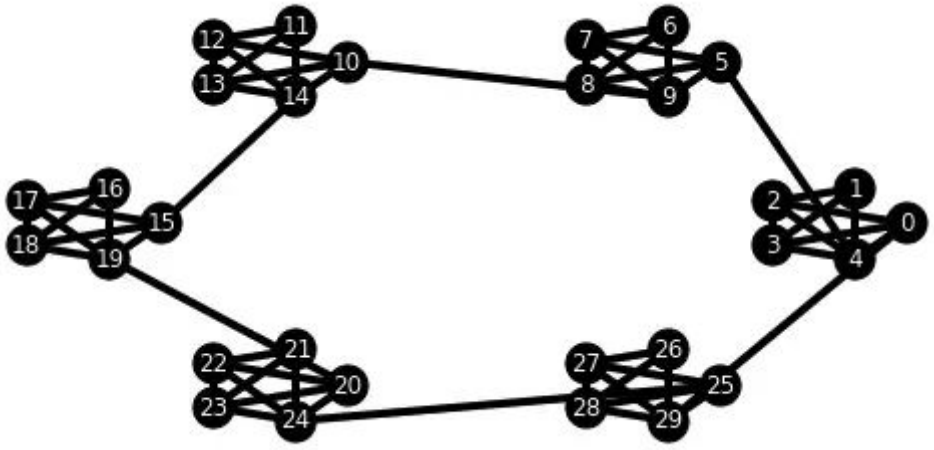
The Clauset Newman Moore algorithm addresses the large time complexity of the Girvan Newman algorithm by proposing an algorithm based on the greedy optimization of modularity of the network.

The algorithm looks at two separate communities and defines delta Q as the change in modularity of the network if the communities were merged. If delta Q > 0 then merging the two communities will improve the modularity of the network. The algorithm starts with every node in its own community and calculates delta Q for every pair of communities. Whichever two communities have the largest delta Q are merged and this is repeated until all the delta Q are negative.

3.3. Louvain Modularity algorithm

One popular class of community detection algorithms seek to optimize the so-called modularity of the community assignment. Following the definition of modularity explained in Girvan-Neumann algorithm, now I am going to provide a more detailed explanation about Louvain Method.

Louvain Method works by applying it to a "connected caveman graph." This is a network where you begin with *N* fully-connected cliques of *M* nodes each. Next, you arrange these cliques in a circle. Then, you take one random link from each clique and rewire it so that the clique is connected to its nearest clockwise neighbor.



All links in this initial network have unit weight.

In the first stage of the Louvain Method, we iterate through each of the nodes in the network. For each node, we consider the change in modularity if we remove the node from its current community and place it in the community of one of its neighbors. We compute the modularity change for each of the node's neighbors.

The next stage in the Louvain Method is to use the communities that were discovered in the community reassignment stage to define a new, coarse-grained network. In the simply connected caveman network that we're studying, there's only one, the unit-weight link connecting neighboring communities, so the links between the coarse-grained communities also have unit weight. If there were two or more links between communities, the new coarse-grained link would have the weight equal to the sum of all the lower-level links.

The rest of the Louvain Method consists of the repeated application of stages 1 and 2. By applying stage 1 (the community reassignment phase) to the coarse-grained graph, you find the second tier of communities of communities of nodes. Then, in the next application of stage 2, you define a new coarse-grained graph at this higher level of the hierarchy. You keep going like this until an application of stage 1 yields no reassignments. At that point, repeated application of stages 1 and 2 will never yield any more modularity-optimization changes, so the process is complete.

3.4. Walktrap Algorithm

This approach is based on the intuition that random walks on a graph tend to get 'trapped' into densely connected parts corresponding to communities.

After the construction of transition matrix P, we introduce a distance r between the vertices that captures the community structure of the graph and r_C1C2 between 2 communities. Let C_1 , C_2 be two communities. We define the distance r_{C1C2} between these 2 communities by:

$$r_{C_1C_2} = \left\| D^{-\frac{1}{2}} P^t_{C_1\bullet} - D^{-\frac{1}{2}} P^t_{C_2\bullet} \right\| = \sqrt{\sum_{k=1}^{n} \frac{(P^t_{C_1k} - P^t_{C_2k})^2}{d(k)}}$$

Next, the problem of finding communities is now a hierarchical clustering problem. An agglomerative approach based on Ward's method will be applied here.

We start from a partition of the graph into n communities reduced to a single vertex. Firstly, compute the distances between all adjacent vertices. Then this partition evolves by repeating the following operations. At each step k:

1) Choose 2 communities C1 and C2 in P_k according to a criterion based on the distance between the communities.

At each step k, we merge the 2 communities that minimize the mean of the squared distances between each vertex and its community.

$$\sigma_k = \frac{1}{n} \sum_{C \in \mathcal{P}_k} \sum_{i \in C} r_{iC}^2$$

2) Merge these 2 communities into a new community $C_3 = C_1 \cup C_2$ and create the new partition: $\mathcal{P}_{k+1} = (\mathcal{P}_k \setminus \{C_1, C_2\}) \cup \{C_3\}$.

3) Update the distances between communities.

After n-1 steps, the algorithm finished and we obtain $\mathcal{P}_n = \{V\}$. Each step defines a partition Pk of the graph into communities, which gives a hierarchical structure of communities called dendrogram.

4. Experiment setup and the data used

4.1. The Data Used

In our project, *email-core data, dblp-data(with ground truth); facebook data and biosnap data(without ground truth)* are analyzed with several algorithms to reveal structures and gain new insights from these data.

4.2. Evaluation Methods

We use 2 different ways to evaluate our algorithms, external way for dataset with ground truth and internal way for dataset without ground truth.

A. Internal evaluation:
   a. Partition quality function (Modularity)
   b. Execution time and Complexity

B. External Evaluation:
   a. Ground-truth testing(F1-Measure)

For internal evaluation in our methods, we calculate the modularity

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

After we implement our methods on the datasets, the algorithm with larger modularity means it has a better result. We also evaluate the time complexity and true execution time of each algorithm. A good algorithm should both has good results and low time consuming.

For external evaluation in our methods, after compare the results with the ground truth data. we have three parameters to calculate:

- P(Precision) -- % of nodes in community x belonging to the ground truth community y.
- R(Recall) -- % of nodes of the ground truth community y covered by x
- F1(Community quality) -- $F1 = 2\frac{PR}{P+R}$

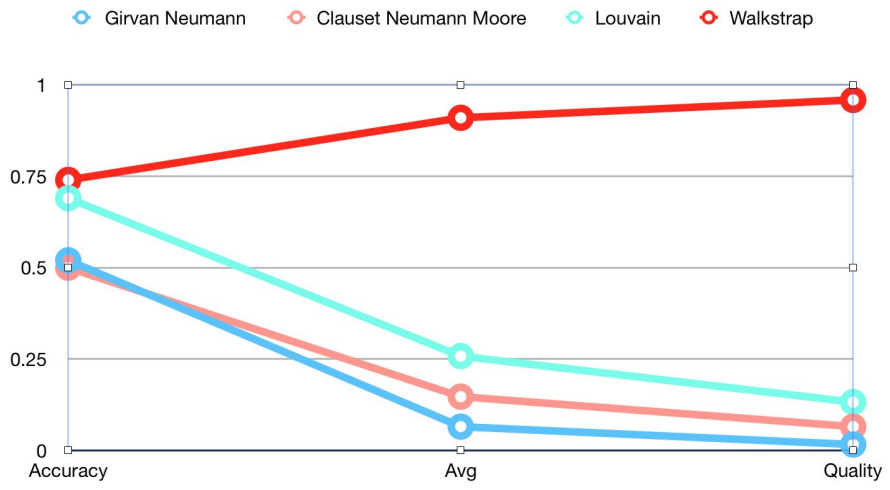
This computational complexity of this method is only O(|n|)(n is the size of the community set). It is also easy to interpret and not affected by overlap. The results with higher P and lower R means our methods may lead to under estimate. The results with higher R and lower P means our methods may lead to over estimate. The results with both higher R and higher P will have high F1 which means our methods is perfect match to the ground truth data..

5. Comparison Results

For email dataset with ground truth, we use external method to do evaluation and comparison. After thorough consideration, F1 measure is chosen because of its low computational complexity, easy interpretation and not affected by overlap. The Walktrap algorithm shows a better partition than other methods while Louvain algorithm owns the shortest running time.

5.1. Results on data with ground truth

- **email-core dataset:**

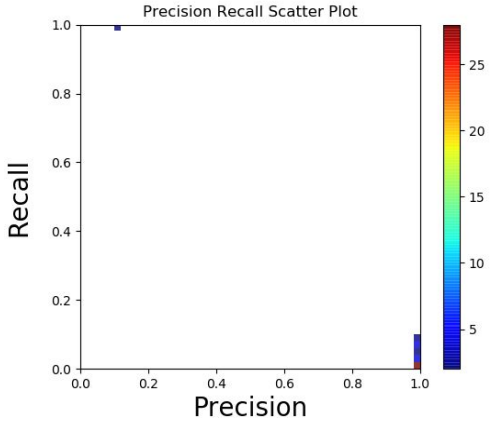


Accuracy -- The percent of ground-truth data mathed.

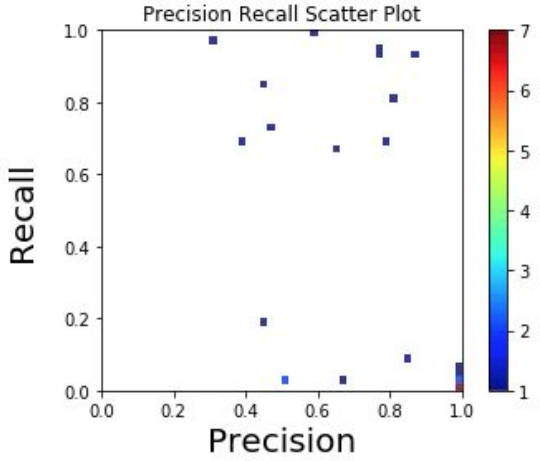
Avg -- The average matching rate of each communities in ground truth data

Quality -- The F1 quality of the final result

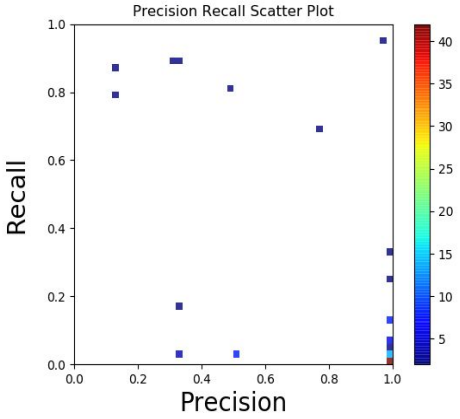
These are four outputs for the F1 measure result. The x-axis is precision and y-axis is recall. From above we could see for the percent of ground-truth data mathed, the louvain method and walktrap has similar accuracy. The GM and CNM is a little low. But when it comes to the average matching rate of each community, the louvain method is much worse than the walktrap algorithm. The walktrap algorithm has the best performance with the highest accuracy and highest average matching rate. So when we test deeper through F1 measure, the Walktrap algorithm provide a higher F1 quality.



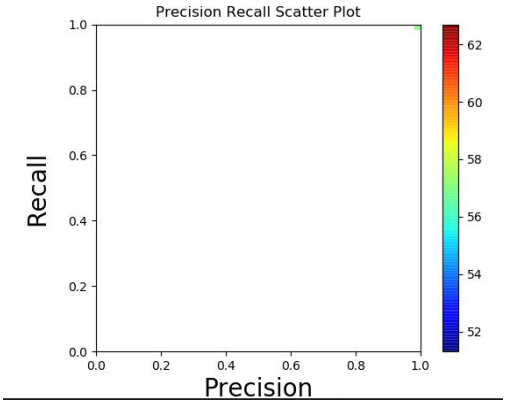
Girvan Neumann



GNM



Louvain Algorithm



Walktrap Algorithm

With high precision and low recall means we maybe underestimate the results. With high recall and low precision means we maybe overestimate the results. From the figure above we could see that the most results GN produced is underestimate with little results overestimate. The results from GNM we could see most communities are underestimate but some communities are close to the perfect math. For louvain algorithm, the results are sparse. Most communities are underestimate, some are overestimate, some are not good with both precision and recall, one community is close to perfect match. The walktrap algorithm has a really good results. With both high precision and high

recall means that is perfect match. And all points are perfect match to the ground-truth results.

Both GN and louvain algorithm is working better on undirected graph than directed graph, but walktrap algorithm working on both graph type. Email data is actually an directed data. When we perform our algorithm on email dataset, we just consider it as undirected, and ignored the direction information contained in the input data. The loss of information is also the reason why GN and louvain has low F1 quality and most of them underestimate.

To assess the quality of community structure discovery algorithms, we rely on the F1 measure derived from normalized mutual information. However this measure, which is commonly used in the context of classical, clustering to compare two different partitions, does not take into account the topological properties of the compared community structures. A natural extension of this work consists of investigating this complementary aspect. In the future, we plan to use community-oriented topological measures to compare the community structure revealed by competing algorithms. The interest of such an approach is to shed additional light on the present evaluation.

- **Some modification on Louvain Algorithm**

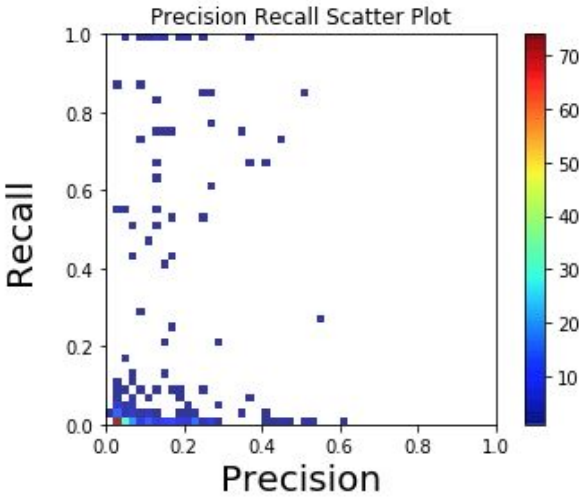
Louvain Algorithm is a greedy optimization method and it is only based on the calculation of modularity. From the Modularity calculation equation:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

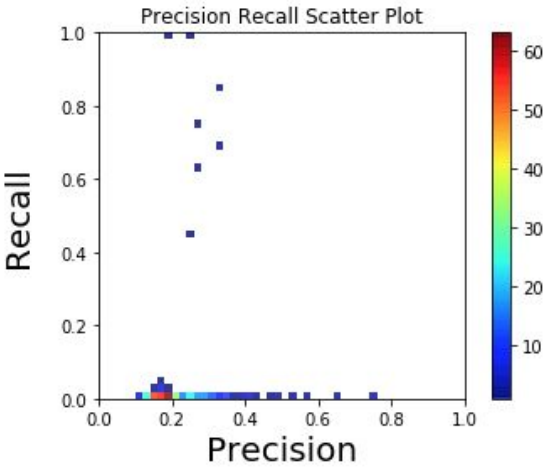
This algorithm produce 273 communities for DBLP data with modularity 0.82 but the ground truth data has 13,477 communities. And produce 261 communitie for Amazon data with modularity 0.93.

From the internal evaluation side, this algorithm provide a really good result with high modularity. But when we compare the result with ground truth data. It generate a really bad results.

DATA TYPE	Accuracy	Average matching rate	F1-Quality
DBLP data	0.013	0.059	0.003
Amazon Data	0.01	0.014	0.001



Plot for DBLP data



Plot for Amazon data

We could easily see that for larger networks, the Louvain method doesn't stop with the "intuitive" communities. Instead, there's a second pass through the community modification and coarse-graining stages, in which several of the intuitive communities are merged together. This is a general problem with modularity optimization algorithms; they have trouble detecting small communities in large networks. It's a virtue of the Louvain method that something close to the intuitive community structure is available as an intermediate step in the process.

Try to solve this problem, we propose to use what we call the extended modularity, which is a generalization of the classic modularity that introduces a new parameter α. The parameter α allows us to tune the granularity of the communities found by the optimization algorithm. When α is 1 the extended modularity is equivalent to the classic modularity.

$$Q = \sum_{vw} \left[\frac{A_{vw}}{2m} - \alpha \times \frac{k_v * k_w}{(2m)(2m)} \right] \delta(c_v, c_w)$$

The modularity measure has two components: one can be interpreted as the probability that a randomly chosen edge will connect nodes from the same partition, and the second–subtracted–component is the same expected probability for the graph after if it were randomly rewired. The second component is important because without it, the optimization algorithm would find the trivial solution of creating a big cluster with all nodes in it. By multiplying the second factor by a parameter α, we can control the granularity of the clustering solution found by the optimization algorithm. Networks tend to have communities and communities inside communities. We can select the level of sub-communities that we want to find by changing the value of α. Small values of α will tend to give big communities, while bigger values of α will tend to find smaller communities.

After our modification on the Louvain algorithm, we get more communities for both dblp data and amazon data.

DBLP Data

α size	Modularity	Communities	Accuracy	Average matching rate	F1-Quality
1	0.82	273	0.01	0.014	0.001
800	0.60	13336	0.382	0.096	0.014

Amazon Data

α size	Modularity	Communities	Accuracy	Average matching rate	F1-Quality
1	0.93	261	0.01	0.014	0.001
1000	0.88	14628	0.077	0.054	0.002
10000	0.3	54712	0.281	0.059	0.006

With the ground truth data, we use them to find the optimal α, and we use this value for the testing data. In real world, when we built different communities in large network, modularity is not the only thing we considered. We should combined different information together to get the final results.

5.2. Results on Data without Ground Truth:

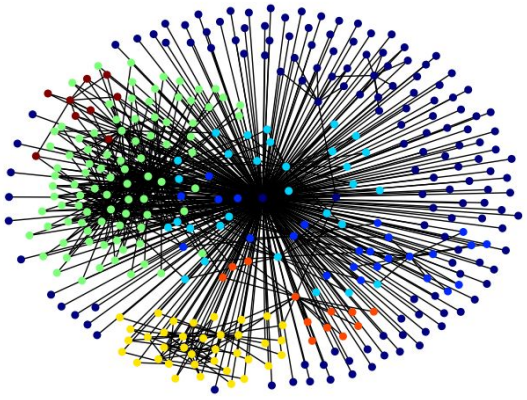
For facebook dataset and biosnap dataset without ground truth, we mainly use modularity and space/time complexity to do evaluations.

	Girvan Newman	Clauset Newman Moore	Louvain	Walktrap
modularity for email data	0.471	0.438	0.41	0.3781061
modularity for facebook data	0.782	0.812	0.89	0.8226436

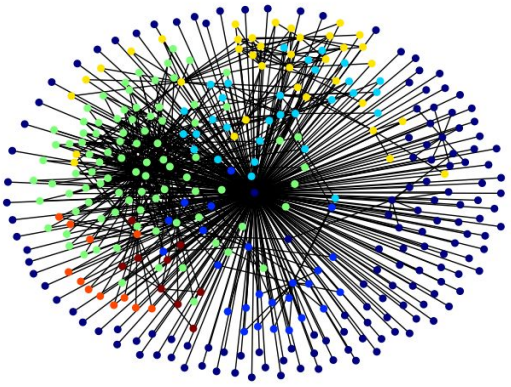
modularity for biosnap data	0.538	0.642	0.75	0.6756927
time complexity	$(O(m^{2n}))$	O(n log2 n)	O(nlogn)	$O(mn^2)$ in worst case; $O(n^2 logn)$ in most cases
space complexity	$O(n^2)$	$O(n^2)$	O(n)	$O(n^2)$

- **Facebook Data:**

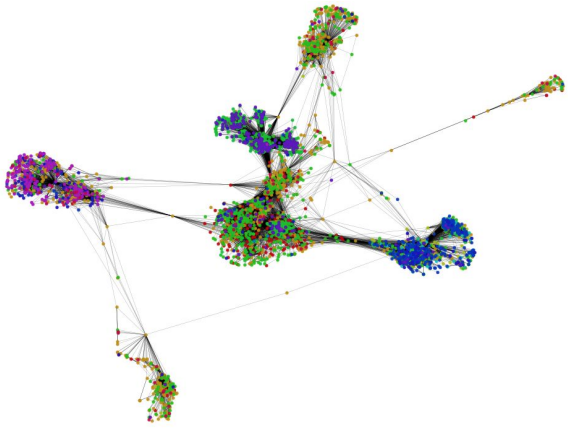
  Community detection visualization:



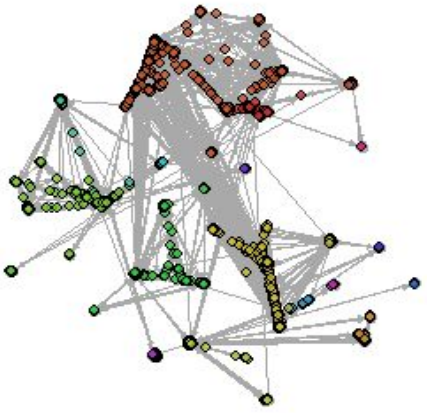
Girvan Newman



Clauset Newman Moore



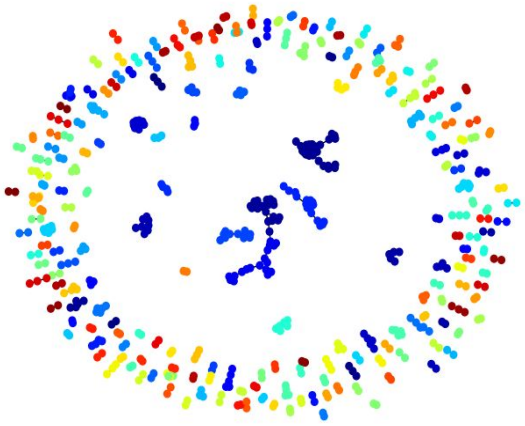
Louvain



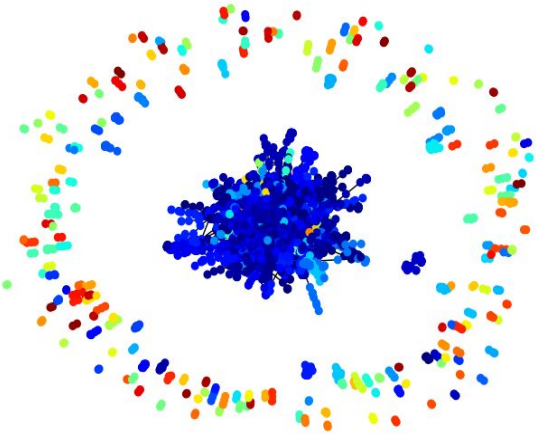
Walktrap

Using modularity comparison criteria, we rank our 4 methods as Louvain > Walktrap > CNM > Girvan Neumann, which is similar with what we have read in essays.

- **biosnap data:**

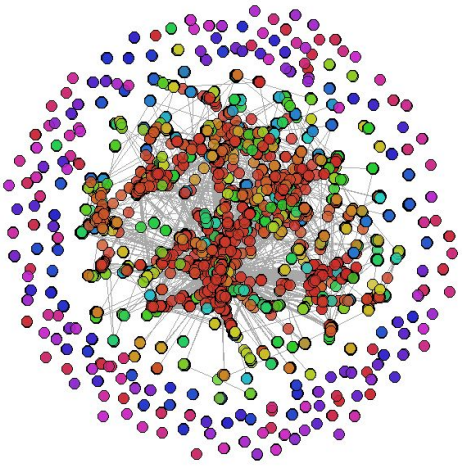


Girvan Newman (subset)



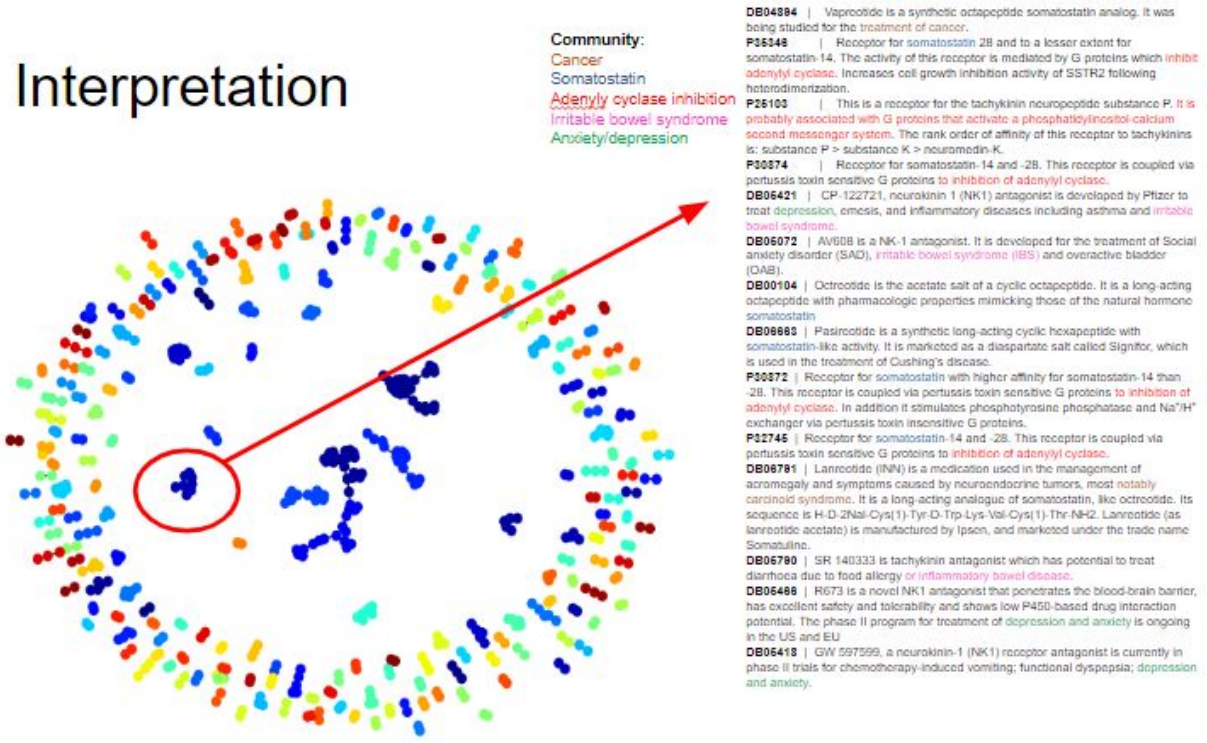
Clauset Newman Moore

Louvain



Walktrap

Because the biosnap dataset had labeled nodes we were able to perform preliminary natural language processing on the clustered data. The dataset consisted of gene and drug interaction pairs. First, we took a random cluster and examined the most frequent phrases in the cluster to validate our

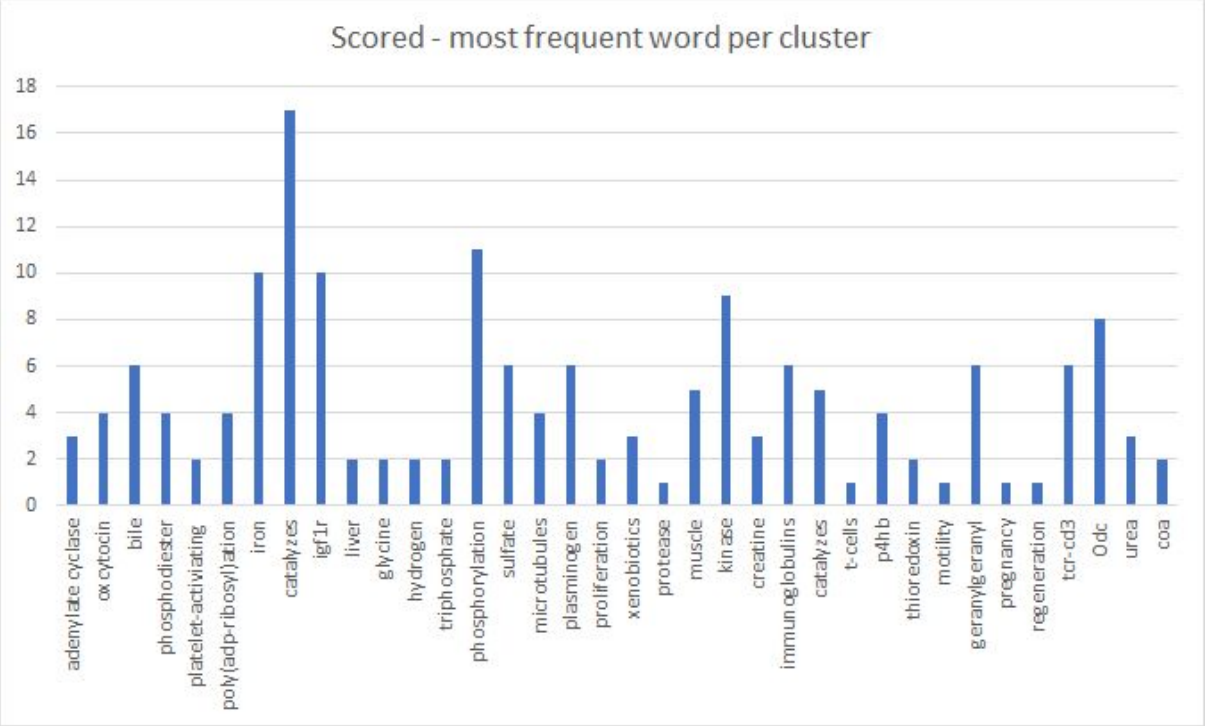


clustering.

The figure shows that for a given cluster there are several key phrases that appear frequently within the gene/drug functions that correspond to certain pathological conditions or biological processes. Because of the high incidence of these keywords and their underlying relationship, we can see how the clusters can give insights to in pathology and signal transduction pathways connections in medicine and biology. If several conditions or diseases and symptoms previously thought as unrelated were clustered together within a complex interaction network there is possibility for new insights and discovery.

To further automate the discovery process for large datasets we developed a toolchain to translate the gene IDs in the clusters to string blurbs about the function of each gene and then apply text mining techniques to characterize the clusters. Using REST API provided by the Uniprot gene database we were able to develop a script to query all of the gene members of a given cluster and translate them into text files containing the gene function for each of the nodes. Then, using text mining libraries in R, we extracted the most frequent words for each cluster and scored them based on their relative frequency compared to the next most frequent word. This was a rough metric for the importance of the word in identifying cluster characteristics. We can see how certain clusters are heavily categorized with one keyword, while others remain more ambiguous. Some clusters even pinpoint the specific functional gene for the cluster, and with a high score (igf1r or tcr-cd3) we can be more confident the members of this cluster are correlated. Further work would be to implement

tf-ifd calculations on the corpus of gene function clusters, and examine each cluster to see the most common words per cluster.



6. Conclusion and beyond

6.1. Comparison Summary

From our implementations on more than 3 datasets, we mainly drive to such a brief comparison between our 4 algorithms:

Girvan Neumann	pros	classic way, many novel approaches have adopted it
	cons	very time-consuming on large-scale networks
CNM	pros	can be used in very large datasets and owns fast speed.
	cons	relatively bad results
Louvain	pros	relatively good tradeoff between the quality and running time, one of the most popular and best method now.
	cons	in our experiments, the quality is lower than the walktrap method.
Walktrap	pros	better quality compared to other methods
	cons	spend a lot of memory and time, very time-cosuming for large dataset

6.2. What we learned from this project

Our contributions are as follow. First, we introduced a modification to the NMI measure to evaluate the performance of our algorithms, in order to make the embeddedness distribution more realistic in the generated networks. Second, we studied some networks data in terms of community-centered properties. This complements some previous analyses focusing on network-centered properties. Third, we applied several community detection algorithms on ground-truth networks and characterized their results relatively to the same community-centered properties.

Our work can be extended in various ways. Applying several algorithms relying on the same definition of the community concept would allow to compare their properties and maybe associate a certain type of community structure to a certain family of algorithms. It could additionally be interesting to use other performance measures than the NMI to assess their relevance with the studied topological properties.

However, based on our results, some existing community detection algorithms still need to be improved to better uncover the ground truth of networks.

6.3. Future work – what can be done after your project

There are many aspects of the project that can be extended and improved upon. First and foremost we want to extend our network community clustering to larger datasets. Many of the algorithms used in this study could not handle the large sizes of the SNAP validation datasets, and thus we were limited to the email validation dataset. This extension could include implementing more novel algorithms that can better handle large networks, or running the algorithms in this study on a server or cloud with increased memory and parallelization.

Additionally, we want to work on improving the accuracy of our algorithms. By improving the compatibility of our algorithms with large networks we can test the algorithms on more datasets and obtain a more robust metric of accuracy. Additionally, by employing more novel algorithms or fine tuning the parameters of our implementations we can work to achieve a greater performance for our community detection.

With respect to our cluster definition extraction, we hope to continue work to develop more advanced text mining algorithms such as tf-ifd to better understand the content of our gene-drug clusters. While we defined a toolchain to take clusters from a community detection algorithm and translate them to a corpus of gene definitions per cluster, we hope to use these more robust text mining techniques to better validate our community detection algorithms by comparing how well clustered certain genes and drugs are, while also finding a better way to represent the content or summarize sampled clusters.

6.4. Credit distribution

Name	Contirbution
Rohan Khopkar	Girvan-Newman algorithm Clauset-Newman Moore algorithm
Yiru Wang	Walktrap algorithm
Yachen Qin	Louvain Modularity algorithm
We collaborated on all other things.	