Application of Network Analysis to Protein-Protein Interaction Network

Yuanda Dong

November 25, 2020

1 Introduction

CDK4 is an essential protein involved in the cell cycle and is required for the initiation and progression of various malignancies. [4] Since the growth of tumors is driven by complex proteins networks, it is essential to integrate biochemistry and mathematics to identify possible avenues to limit proliferation of cancers. SACCHAROMYCES CEREVISIAE (yeast) possesses 23% homologous genes to humans, and is considered as a useful model for studying gene function.[5] Because of the accessibility of Baker's yeast, readily available PPI datasets on yeast, our project focused at the CDK4 homolog, CDC28 (systemic name: YBR160W) in yeast. Here, i will report on the mathematical approaches that we have considered/tried and results obtained on the PPI network. This will be followed by a discussion of of the strengths and limitations of our approaches; the mathematical analysis will also be put into context of the overall group project and current body of researches.

2 Approaches and results

At a high level, our approach is to use community detection algorithms to uncover the community structure of the yeast network. Then the central nodes in each of the communities naturally form a list of potential candidates. Biochemical analysis (mainly literature search) reveals their relevance to cancer and their suitability for experimentation. BCMB students then used these proteins for designing experiments to study the target disease. Further, we have performed path analysis between CDC28 and nodes deemed by BCMB students as important (green nodes). The aim of path analysis is to identify potential proteins along the path connecting CDC28 and green nodes. These proteins are generally less central than the green nodes and this makes them suitable targets for knock-down / knock out experiments. However, due to time constraint of the project, the results of path analysis were not conveyed to the BCMB students until the very end, and therefore not used for designing experiments.

This section details on the different methods that we have tried for community detection and path analysis, with focus on mathematical theory, numerical algorithms and computational implementation. There will also be sections explaining how we pruned the network to uncover hidden structures; and how we determined nodes as being central.

2.1 Community detection

Before doing community detection, the network is per-processed to remove edges with confidence level less than 700, and we restrict to the largest component. Researches have shown that most of the methods applied to analyzing biological networks directly or indirectly maximizes modularity, and for Yeast and Human PPI networks, Louvain method is likely the best method to find communities in terms of detecting known core pathways in reasonable time.[6] We applied the Louvain method in our project. However, we also looked at other approaches to community detection including spectral methods involving graph Laplacian or random walk matrix and the maximal clique containing CDC28.

2.1.1 Modularity optimization

In all modularity optimization methods, a modularity metric Q need to be defined. Networks with high modularity will have few connections between different partitions, and dense connections within the partitions. The classic definition of Q is given by the following

$$Q = \frac{1}{2} \sum_{i,j \text{ in the same group}} A_{i,j} - \frac{1}{2} \sum_{i,j \text{in the same group}} k_i \frac{k_j}{2m}$$

where $A_{i,j} - k_i \frac{k_j}{2m}$ measures the difference between the actual number edges between node i, j and the expected number of edges. Maximizing this difference (can be done through spectral methods as shown in the lecture) achieves exactly the goal of optimizing modularity. However, spectral methods can only be used for bisections, and doing successive bisections does not typically give the maximal modularity. Hence we have to look at alternative methods such as Combo, Conclude, Fast Greedy and Louvain etc. The Louvain method is best approach for PPI network which has multiple communities as explained in [6]. The intuition of the Louvain method is to

Algorithm 1 Louvain method

```
1. first assign each nodes to a different community.
2. Then form small communities by optimizing modularity measure $\Delta Q$ (defined differently to $Q$) locally
3. These small communities will then form new nodes in a graph where the edges were defined by the sum of edge weights between nodes in the corresponding two communities.
4. Perform step 2-3 again until reaching the maximum modularity
```

Refer to [6] method sections for more details.

Louvain method has theoretical running time of $\mathcal{O}(nlog(n))$, where n is the number of nodes in the network. The method is implemented in Python module networkx and it can find the partitions in less than 1 minute on Ryzen5 3600 cpu with 16 GB of RAM. Applying the default implementation on networkx gives roughly about 15 communities, however we wanted some more granularity for our purpose of identifying interesting nodes. By changing the *parameter resolution = 0.5, random_ state*

= *1*, we were able to find 28 communities and each time we get the same partitioning, results in appendix section. There were two detected communities of size less than 10 nodes, and another one of size 33, all other ones are reasonably sized.

2.1.2 Spectral methods

We computed the eigenvalues of graph Laplacian $L = D - A$, and the random walk matrix $L = AD^{-1}$ of the PPI network. It can be shown (as in the lecture) by considering a perturbation on the ideal case (where each community of the graph is all-to-all connected, but there is no connection among communities), that the number of eigenvalues before the occurrence of the first spectral gap gives the number of communities k in a network. Then, we can run k-means algorithm on the eigenvector matrix $v_1, v_2, v_3, ..., v_k$ to find out the actual allocation of nodes to communities. However in PPI networks, the value k is not well-defined. We plotted the first 300 (ordered by magnitude) eigenvalues of the graph Laplacian and the random walk matrix.

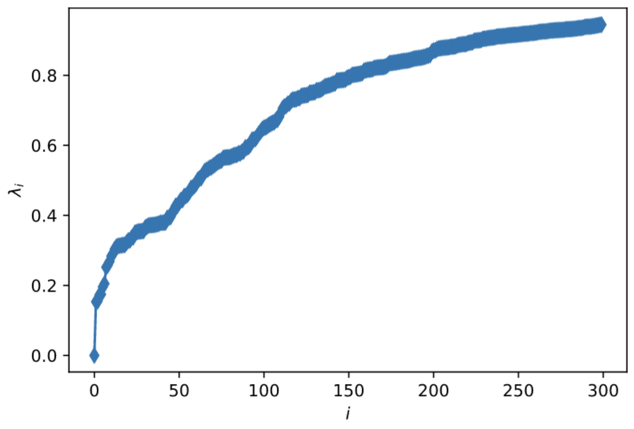


Figure 1: First 300 eigenvalues of the graph Laplacian of PPI network

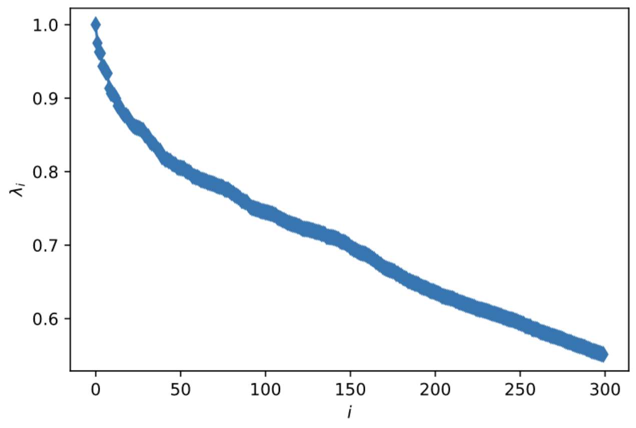


Figure 2: First 300 eigenvalues of the random walk matrix of PPI network

The largest spectral gap for the graph Laplacian and random walk matrix both happened at λ_2, although the spectral gap for the random walk matrix is small. The differences in λ_k and λ_{k+1} in both graphs are generally very small, which indicates strong connections across the network (i.e. less community strucutre). This makes the spectral method unsuitable for community detection in PPI networks.

2.1.3 Maximal clique

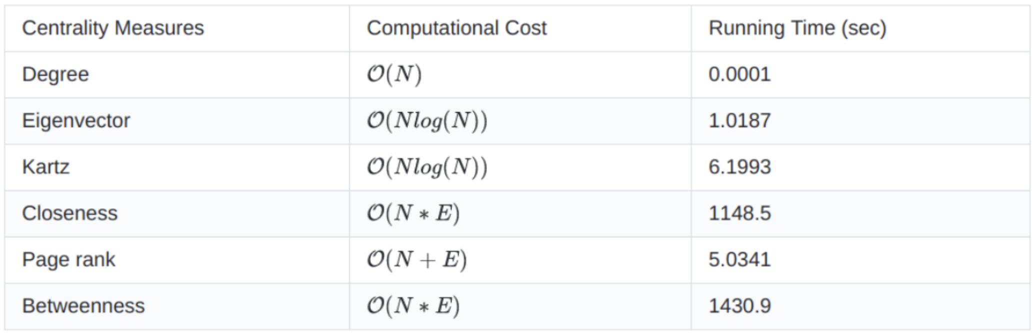
The general clique problem is to find sets of vertices which are all-to-all connected in a graph, and is proved to be NP-hard. (i.e. does not admit polynomial time solutions) Here, we consider a variation of the clique problem, where we want to find the all cliques involving a target node, and then find the maximal clique involving the target. This formulation is still NP-hard, the proof is trivial, and can be done by contradiction.

Proof. If we can find the cliques involving a target node in polynomial time using an algorithm. Then we can used the same algorithm to find all cliques in the graph, and this multiplies the time complexity only by a constant factor N. However, we know the clique problem does not admit polynomial time algorithm. ↯ □

In practice this exponential time is not so bad for our PPI network. Using networkx function, cliques_containing_node(G[, nodes, cliques]), we were able to compute the the largest clique involving CDC28 in a short amount of time, and is given in the appendix. This method gives a strongly connected local community of the target node. However, this defeats our aim of finding interesting nodes in the network, since we are restricted to the neighbors of the target.

2.2 Centrality measures

Once we have identified the community structure, then the central nodes in each of the communities naturally form a list of potential candidates for biochemical analysis. We define the central nodes through centrality measures. Commonly used centrality measures include degree, eigenvector, Kartz, closeness, page rank and between-ness centrality, and they all can be computed in polynomial time.



Centrality Measures	Computational Cost	Running Time (sec)
Degree	$\mathcal{O}(N)$	0.0001
Eigenvector	$\mathcal{O}(Nlog(N))$	1.0187
Kartz	$\mathcal{O}(Nlog(N))$	6.1993
Closeness	$\mathcal{O}(N * E)$	1148.5
Page rank	$\mathcal{O}(N + E)$	5.0341
Betweenness	$\mathcal{O}(N * E)$	1430.9

Figure 3: [1]Running time is measured in CPU time using Ryzen 5 3600 CPU, and tested on the yeast protein network from String Database using Networkx packages.

For our aim of identifying interesting nodes from each community, i chose to use 3 centrality measures and the idea is:

- degree centrality measures the number of connections of a node,
- eigenvector centrality for a node is high if it's connected to important nodes or is connected to lots of nodes
- page rank is similar to eigenvector centrality but it dilutes the importance score gained by being connected other nodes by the degree, the hope is to dilutes the importance gained from biologically essential nodes.

The results are in the appendix section. I found these three centrality measures returns almost the same results for most communities except for some. Specifically, our target node YBR160W is ranked 2_{nd} in page rank centrality, but ranked 8_{th} in degree centrality, and has rank > 10 in eigenvector centrality. Nevertheless, we passed the list of central nodes of each communities to the BCMB students. From this, they were able to find a list of interesting secondary nodes (green nodes). This is also given in the appendix section.

In addition, we have looked at local bridges and cut vertices in each communities (but it's not used for biochemical analysis). I sorted the local bridges by their span, which is defined as the shortest path length between the two end-points of the local bridge, if the local bridge is removed. I noted that almost half of the local bridges have infinite distance, and the others range from 3 to 7. We also analyzed the impact of cut vertices. A vertex is said to have a high impact, if the removal of which disconnected the graph into connected components of reasonably size. The results are in the appendix section.

2.3 Pruning

Removing essential, yet irrelevant to the target disease, nodes helps to uncover the hidden nodes that might be of interest to biochemical analysis. We used the obvious pruning method - list pruning in the project. Here, i also propose an alternative method to pruning that uses modularity measures.

2.3.1 List pruning

In list pruning, we identified a list of top 100 nodes (by degree) of the original network, and passed this list to BCMB students. They go through a manual process of searching each of the node online, and decide whether the node is important for the target disease. We pruned the nodes that are deemed as unimportant.

2.3.2 Prune using Modularity

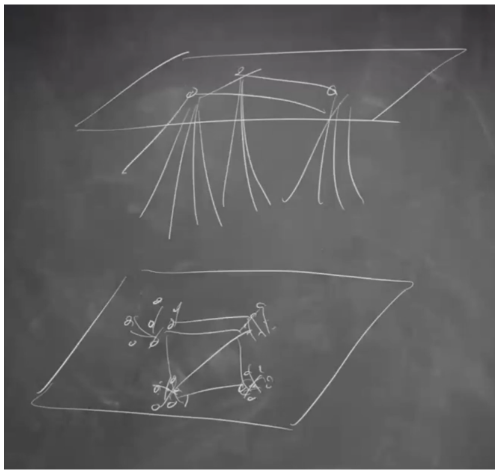


Figure 4: Layered diagram of network, top layer: essential nodes, bottom layer: community structure

We didn't use this method for the project. Referring to Figure 4, We assume the top layer of the diagram are biologically essential nodes of high degree, yet they are irrelevant to the target disease, whereas the bottom layer is the community structure. Then intuitively, removing a central node of a community from the bottom layer will decrease the number of communities, since the members of that community will join other communities. However, removing an essential node will increase the number of communities, since the members of the community can form new communities. This observation gives rise to the following algorithm:

Algorithm 2 Prune using modularity

```
1. Sort the vertices by degree
2. Compute the community allocation and the modularity measure $Q$ of the graph
3. Create a copy of the graph, remove the highest degree node that hasn't been removed before
4. Compute the community allocation and the modularity measure $Q$ of the copy, if $Q_{copy}>$ Q,
   then remove the vertex from the original graph
5. Repeat step 2-4
```

In practice, finding the optimal Q is NP-hard [3], so neither Girvan-Newman (which will take about 100 days)[6], nor Louvain method will find the optimal Q. Nevertheless, i used implemented this using Louvain method and it removed roughly 50 nodes from the top 100 nodes. The result is in appendix.

2.4 Path analysis

Given the node CDC28, and a list of important nodes, the goal is to find interesting nodes along the paths.

2.4.1 Steiner tree

The minimum spanning tree (MST) problem involves finding a subset of edges going through all vertices of a graph that have the minimum weight. It can be solved using Kruskal's algorithm / Prim's algorithm in $O(mlogn)$ time. However, we are interested in finding the minimum spanning tree that go through a particular set of nodes. This is called the Steiner tree problem, and is a generalization of the MST problem where the set of nodes is the all vertices. We assign each edge of the network graph weight 1. Since we have a list of green nodes [nodes that deemed by BCMC students as important], we hope to construct the Steiner tree involving these green nodes. Then the nodes in the Steiner tree will be suitable targets for biochemical analysis.

It is known in computer science that the Steiner tree problem is NP hard. However, there exists many approximation algorithms. In particular, one can find the metric closure (the complete graph G^*, that has edge weight between two nodes defined as the shortest path length), and find the MST of G^*. This method has an approximation ratio of $2 - \frac{2}{t}$, where t is the number of terminal nodes and the running time is dominated by computing pair-wise shortest path length which is $O(n^3)$ using Floyd-Warshall algorithm[2]. After an hour of running the Steiner tree implementation in networkx, on the green nodes ['4932.YGR232W', '4932.YOR335C', '4932.YOL058W',

’4932.YHL050C’, ’4932.YLR466W’], it returned the Steiner tree. Please see the appendix section for more details on the green nodes and the resulting Steiner tree.

2.4.2 Shortest k-paths

The shortest path between CDC28 and one of the green nodes is not guaranteed to have biological significance. We needed a method that’s able to find more candidate nodes than the shortest paths. To this end, we consider the shortest k-paths between CDC28 and one of the green nodes, where $k > k_{shortest}$, $k_{shortest}$ is the length of the shortest path. The value k is determined experimentally, although one can define some heuristic (for example, to keep the total number of nodes that lie on shortest k-paths under 500). One can form a sub-graph consisting of the nodes that lie on the shortest k-paths. Then, the central nodes (we looked at top 6 by degree) of the sub-graph forms a list of candidates for biochemical analysis. We can perform this analysis for each of the green nodes. Interestingly enough, that the 3 nodes on the Steiner tree (excluding the terminals) coincide with the interesting nodes we found through shortest k-path analysis.

Furthermore, one can take the intersection of all sub-graphs (ideally the corresponding green nodes need to be somehow related), the vertices on the intersection may also be interesting. Credit to Nicholas, the code and results are in the appendix section.

3 Discussion

In terms of the basis of our mathematical methods, we have been using fairly standard algorithms (Louvain method) for community detection and common centrality measures. Although they are not ground-breaking by any means, they are widely used and known to return good results. For example, researches have shown the Louvain method is likely the best method to find communities for human and yeast PPI networks.[6] In our case, it is fast $\in O(nlogn)$, and readily implemented in networkx. The running time for centrality measures are also fast (refer to figure 3), and readily implemented in networkx.

The mathematical analysis also has its limitations, which mainly comes from the compromise between computational costs and optimality of the solution. Take the same example, the Louvain method is only an approximation algorithm to the NP-hard problem of modularity optimization, and there is no theoretical proof on the approximation ratio.

Perhaps due to the current pandemic, the back-and-forth exchange of ideas and results between Maths and Biochemistry was not extensive as i would liked. Many mathematical analysis that we have done required more inputs from BCMB students, and vice versa. Nevertheless, the main approach (thoroughly discussed among BCMB and MATH students) is to consider the central nodes in each communities. This approach has its strength and limitations.

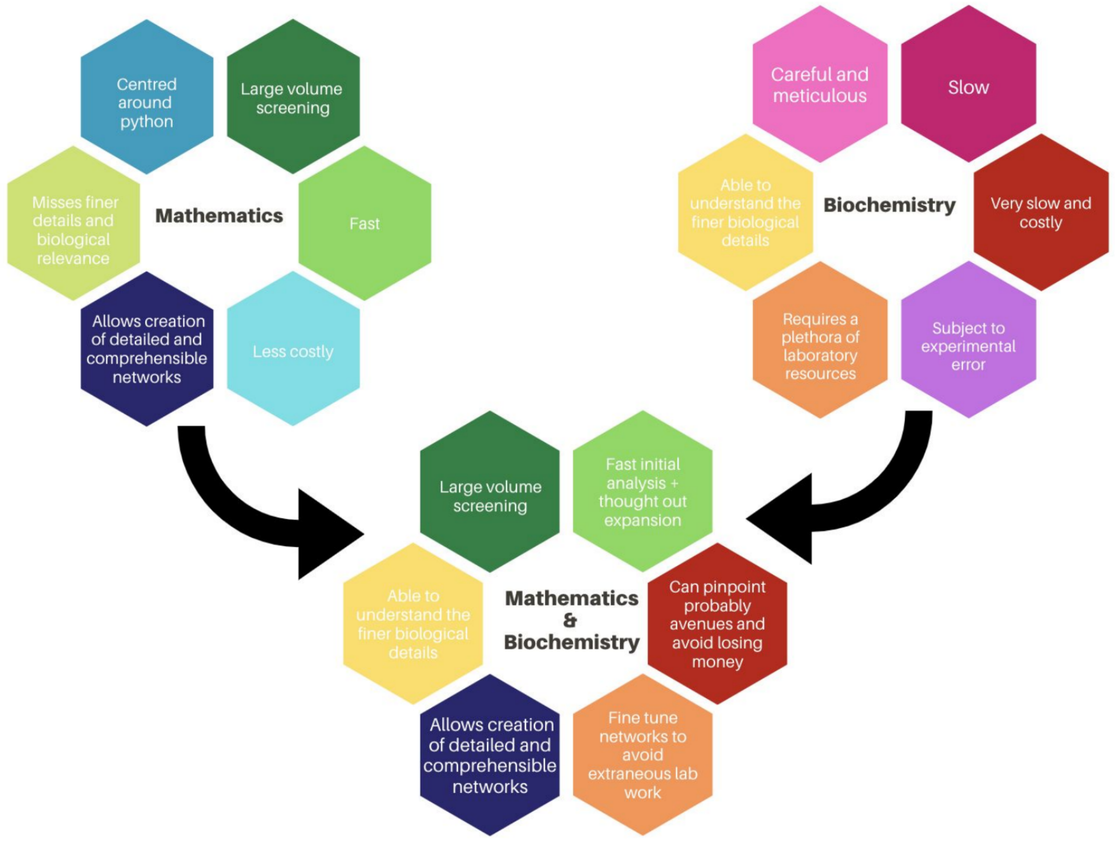

Centred around python
Large volume screening
Mathematics
Misses finer details and biological relevance
Fast
Allows creation of detailed and comprehensible networks
Less costly
Careful and meticulous
Slow
Able to understand the finer biological details
Biochemistry
Very slow and costly
Requires a plethora of laboratory resources
Subject to experimental error
Large volume screening
Fast initial analysis + thought out expansion
Able to understand the finer biological details
Mathematics & Biochemistry
Can pinpoint probably avenues and avoid losing money
Allows creation of detailed and comprehensible networks
Fine tune networks to avoid extraneous lab work

Figure 5: Schematic of integration of two disciplines, created by project group I

Referring to figure 5, mathematical methods can be realized in Python with simple lines of code in short amount of time, while the PPI datasets involving > 6000 proteins, and > 900000 edges are readily available. However, we will miss the finer details and biological relevance. On the other hand, biochemical analysis are meticulous in the sense that every biological details can be considered, but it's costly and slow. Therefore, the integration of Maths and Biochemistry in this project gives rise to its strength such as, large volume of protein screening; fast initial analysis and later expansion on results; ability to pinpoint possible research avenues etc. In addition, referring to the second paragraph, careful biochemical analysis on lists of candidates nodes found using approximation algorithms, helps to alleviate some of the imprecision of approximation algorithms. The limitation here is really the effectiveness of communication between MATH and BCMB students.

Putting our project in the context of current field of cancer research, it's by no means groundbreaking. However, it provides some insights into selection of target proteins for biological treatments. For instance, CDK4 is known to to be related to tumors, but inhibition of which can cause adverse effects on the cell cycle. Our project is all about finding secondary proteins that are less important than CDK4, but still can be targeted to treat cancer. The process of performing network analysis and then biochemical experiments on yeast require collaboration between MATH and Biochemistry knowledge, but it can be scaled and be used as a model for studying cancer in human.

4 Conclusions

In conclusion, the mathematical analysis of the complex PPI networks have centered around community detection and path analysis of proteins. The mathematical methods are realized in Python, however the implementations have its theoretical and practical limitations. The results from mathematical methods are used for biochemical analysis. The integration of Biochemistry and Mathematics allows biochemists to identify target proteins for designing experiments to study cancer.

References

[1] Oracle reference algorithms. https://docs.oracle.com/cd/E56133_01/2.4.1/reference/algorithms/. Accessed: 2020-11-24.

[2] Steiner tree problem. https://www.wikiwand.com/en/Steiner_tree_problem. Accessed: 2020-11-25.

[3] U. Brandes, D. Delling, M. Gaertler, R. Goerke, Martin Hoefer, Z. Nikoloski, and D. Wagner. Maximizing modularity is hard, 09 2006.

[4] Shom Goel, Molly J. DeCristo, April C. Watt, Haley BrinJones, Jaclyn Sceneay, Ben B. Li, Naveed Khan, Jessalyn M. Ubellacker, Shaozhen Xie, Otto Metzger-Filho, Jeremy Hoog, Matthew J. Ellis, Cynthia X. Ma, Susanne Ramm, Ian E. Krop, Eric P. Winer, Thomas M. Roberts, Hye-Jung Kim, Sandra S. McAllister, and Jean J. Zhao. Cdk4/6 inhibition triggers anti-tumour immunity. *Nature*, 548(7668):471–475, 2017.

[5] Wei Liu, Li Li, Hua Ye, Haiwei Chen, Weibiao Shen, Yuexian Zhong, Tian Tian, and Huaqin He. From saccharomyces cerevisiae to human: The important gene co-expression modules. *Biomedical reports*, 7:153–158, Aug 2017.

[6] Sara Rahiminejad, Mano R. Maurya, and Shankar Subramaniam. Topological and functional comparison of community detection algorithms in biological networks. *BMC Bioinformatics*, 20(1):212, 2019.

5 Appendices

Louvain method results

Link: https://drive.google.com/file/d/1DEkjhXDrmVoqPiyL-uNjflA57nuT6lcT/view?usp=sharing

Maximal clique involving CDC28

['4932.YBR160W', '4932.YPR119W', '4932.YPR120C', '4932.YGR109C', '4932.YLR210W', '4932.YDL155W', '4932.YLR167W', '4932.YLL039C', '4932.YKR094C', '4932.YJL194W', '4932.YGL003C', '4932.YBR135W', '4932.YFR004W', '4932.YJL001W', '4932.YDL147W', '4932.YHR027C', '4932.YHR200W', '4932.YFR050C', '4932.YPR108W', '4932.YPR103W', '4932.YDL097C', '4932.YBL041W', '4932.YFR052W', '4932.YOL038W', '4932.YGR232W', '4932.YDR394W', '4932.YOR157C', '4932.YER021W', '4932.YKL145W', '4932.YOR362C', '4932.YGL048C', '4932.YML092C', '4932.YMR314W', '4932.YER012W', '4932.YGL011C', '4932.YGR253C', '4932.YIL007C', '4932.YDR427W', '4932.YGR135W', '4932.YOR261C', '4932.YOR259C', '4932.YER094C', '4932.YIL075C', '4932.YDL007W', '4932.YOR117W', '4932.YDL132W', '4932.YDR328C']

Centrality results for all communities

Link: https://drive.google.com/file/d/1nSS0n_TOz_sLPHT_0b2pQQ_0FgTf3ED6/view?usp=sharing

Green nodes

Link: `https://drive.google.com/file/d/1fwf92852NWl4iOJNzl1vIwb-2zRSxbiV/view?usp=sharing`

Local bridges and cut vertices

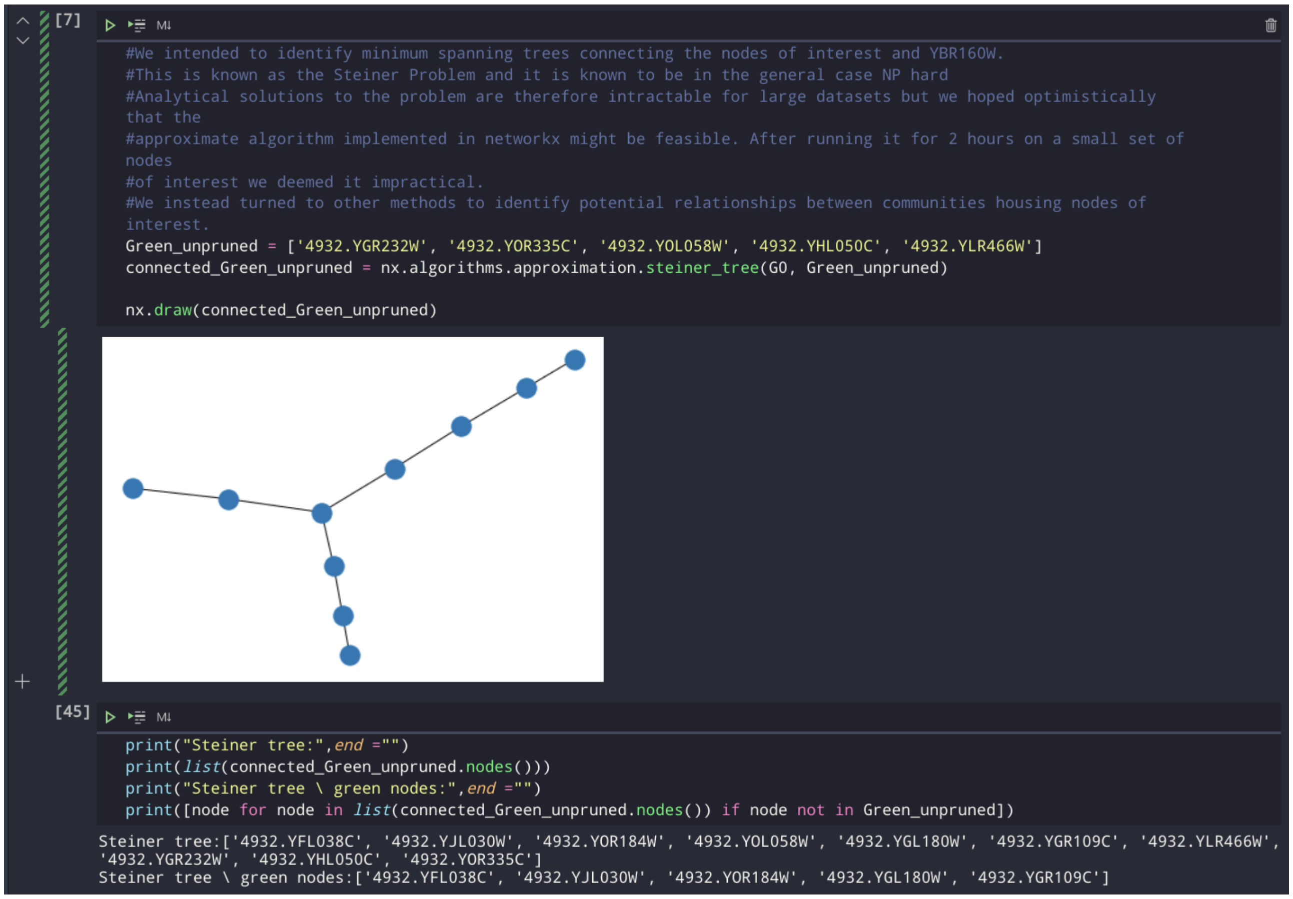
Link: `https://drive.google.com/file/d/15jvYQ4X-ICikgaQ8c40F1eAirrMWL7OA/view?usp=sharing`

Prune using modularity

Link: `https://drive.google.com/file/d/1URy9Hzw7NCXwGvHkuzqqWV7wGH0UqVjs/view?usp=sharing`

Steiner tree

Actually, during the project, we asked the BCMB students to identify important nodes from pruned and unpruned netowrk, since it's the effect of pruning was unknown at the moment. The green nodes used to generate the Steiner tree in this case is the Green nodes identified from the unpruned network.



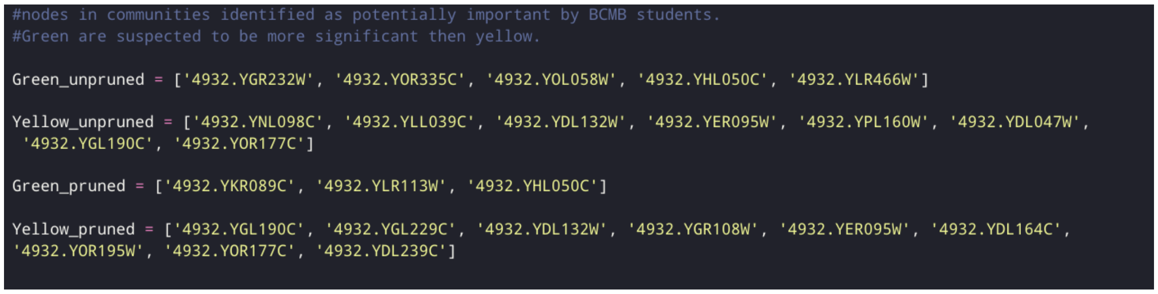
```
[7]
#We intended to identify minimum spanning trees connecting the nodes of interest and YBR160W.
#This is known as the Steiner Problem and it is known to be in the general case NP hard
#Analytical solutions to the problem are therefore intractable for large datasets but we hoped optimistically
that the
#approximate algorithm implemented in networkx might be feasible. After running it for 2 hours on a small set of
nodes
#of interest we deemed it impractical.
#We instead turned to other methods to identify potential relationships between communities housing nodes of
interest.
Green_unpruned = ['4932.YGR232W', '4932.YOR335C', '4932.YOL058W', '4932.YHL050C', '4932.YLR466W']
connected_Green_unpruned = nx.algorithms.approximation.steiner_tree(G0, Green_unpruned)

nx.draw(connected_Green_unpruned)
```

```
[45]
print("Steiner tree:",end ="")
print(list(connected_Green_unpruned.nodes()))
print("Steiner tree \ green nodes:",end ="")
print([node for node in list(connected_Green_unpruned.nodes()) if node not in Green_unpruned])
```

```
Steiner tree:['4932.YFL038C', '4932.YJL030W', '4932.YOR184W', '4932.YOL058W', '4932.YGL180W', '4932.YGR109C', '4932.YLR466W',
'4932.YGR232W', '4932.YHL050C', '4932.YOR335C']
Steiner tree \ green nodes:['4932.YFL038C', '4932.YJL030W', '4932.YOR184W', '4932.YGL180W', '4932.YGR109C']
```

Figure 6: Steiner tree involving the green nodes

Here is the list of green nodes on the pruned/unpruned network and also yellow nodes (less important than green ones).



```python
#nodes in communities identified as potentially important by BCMB students.
#Green are suspected to be more significant then yellow.

Green_unpruned = ['4932.YGR232W', '4932.YOR335C', '4932.YOL058W', '4932.YHL050C', '4932.YLR466W']

Yellow_unpruned = ['4932.YNL098C', '4932.YLL039C', '4932.YDL132W', '4932.YER095W', '4932.YPL160W', '4932.YDL047W',
 '4932.YGL190C', '4932.YOR177C']

Green_pruned = ['4932.YKR089C', '4932.YLR113W', '4932.YHL050C']

Yellow_pruned = ['4932.YGL190C', '4932.YGL229C', '4932.YDL132W', '4932.YGR108W', '4932.YER095W', '4932.YDL164C',
'4932.YOR195W', '4932.YOR177C', '4932.YDL239C']
```

Figure 7: List of green and yellow nodes

Shortest k-paths results

Credit to Nicholas Giannoulis,

Link: `https://drive.google.com/file/d/15K60F6Xk4Km-7MA7t3JpvRejfT0sabhz/view?usp=sharing`