

Comp790-166: Computational Biology

Lecture 4

January 25, 2022

Welcome Back

Today...

- Finish priming graph partitioning, graph clustering, community detection
- Modularity Based (e.g. Using a null model)
 - Louvain
 - Leiden
 - Identifying Disease-Relevant Gene and Protein Interactions
- Probabilistic
 - Stochastic block model
 - Affiliation model
- Clique counting and start graph embedding

Community Detection Illustrated

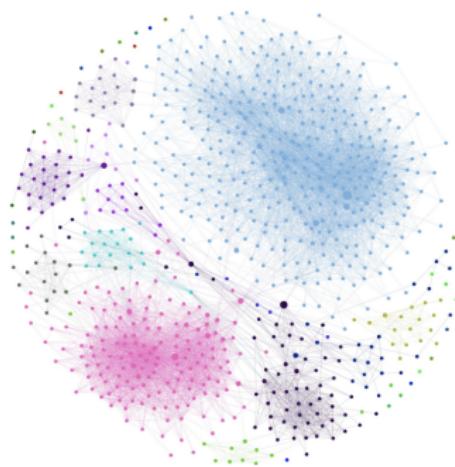


Figure: The high-level goal is to partition nodes into coherent node-subsets (communities), such that nodes within a community have on average more within group edges than between-group edges.

Disclaimer : Multiple Ways of Referring to Elements of The Partition

Depending on which literature you read, you may hear the following. Also, I will use these words interchangeably,

- cluster
- partition
- module
- community

Community Detection (Graph Partitioning) is just Clustering

- Instead of your standard clustering problem in a matrix of N objects with P features, our input here is an adjacency matrix, \mathbf{A} , where we want to cluster nodes based on similarities in their connectivity patterns.
- Most community detection optimization problems seek a hard partition (each node assigned to a single community)
- There are some variants that learn a soft partition, or a propensity or probability that each node is assigned to each community.
- Optimization for this problem can come in many flavors

Optimization Approaches

- Quality Function with a Null Model + Heuristic for Optimization ←
 - A null model describes a graph with no structure, for example, nodes connected randomly.
- Probabilistic and Likelihood Optimization ←
- Spectral Clustering Methods (Partition based on graph Laplacian)
- Most recently: graph embedding + clustering on embedding
 - We will touch on this briefly Thursday with node2vec

Why Might a Person Waste Time Partitioning a Graph?

Build a graph between cells based on marker expression and partition into cell-populations. Members of a cell-population should be phenotypically similar and therefore express the same markers.

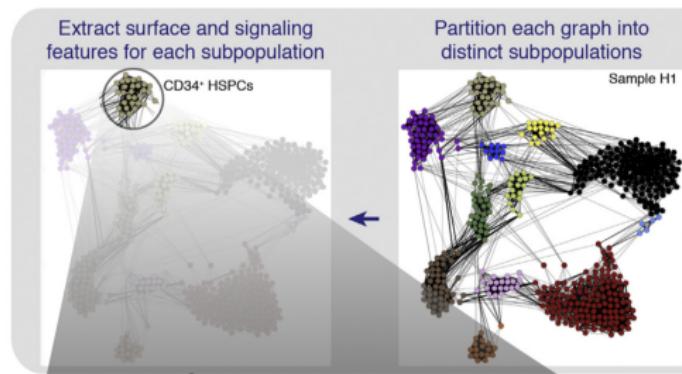


Figure: from Levine *et al.* Cell. 2015. This is the PhenoGraph algorithm.

PhenoGraph Uses Modularity Based Maximization

- **Intuition:** We first specify a null model of a network with no structure about the probability of two nodes being connected. Then we find the partition that is maximally different from this null model.
- **A Simple Null Model:** An easy way to think about the probability of two edges being connected is based on some function of their degree.
 - Consider the null model, $\frac{k_i k_j}{2M}$
 - k_i and k_j give the degrees of nodes i and j
 - M (as always in our notation) is the total number of edges, or the sum of all edge weights.
- **Configuration Model:** This is known as the configuration model, or fixing the degree sequence and connecting nodes at random. ¹

¹ see

https://sites.santafe.edu/~aaronc/courses/3352/csci3352_2019_L3.pdf for combinatorics exercise for null model

Modularity Defined

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

- A_{ij} is the adjacency matrix entry for node pair (i, j) (can be weighted and not just binary)
- $\delta(c_i, c_j)$ is an indicator function for whether or not nodes i and j were assigned to the same community.
- We need an algorithm to help us determine node-to-community assignments for all nodes i , such that Q is as large as possible.
- γ is a resolution parameter controlling the size of communities

Louvain: A Simple Algorithm that Makes a lot of Sense

Merge (if modularity increases), agglomerate, repeat until modularity doesn't increase anymore.

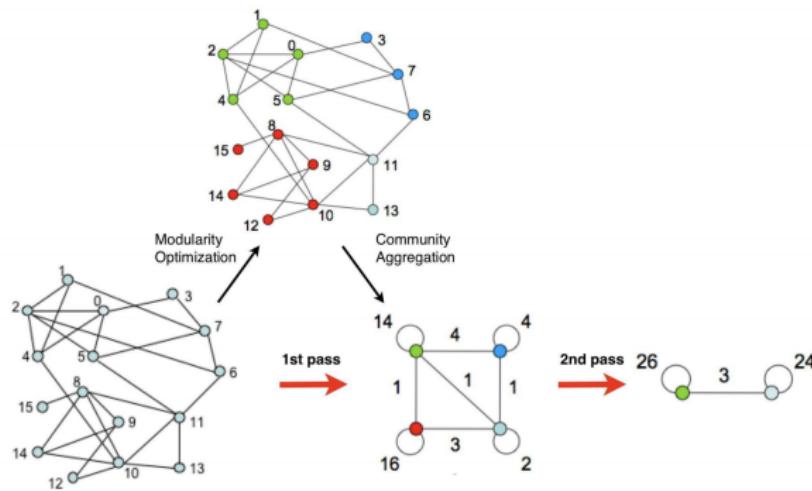


Figure: from Blondel *et al.* Journal of Statistical Mechanics. 2008.

Louvain is Fast Because Potential Merges are Easy to Compute

They show in Blondel *et al.* 2008 that the change in modularity by moving a node into a community, C , can be computed in closed form as,

$$\Delta Q = \left[\frac{\sum_{in} + k_{i,in}}{2M} - \left(\frac{\sum_{tot} + k_i}{2M} \right)^2 \right] - \left[\frac{\sum_{in}}{2M} - \left(\frac{\sum_{tot}}{2M} \right)^2 - \left(\frac{k_i}{2M} \right)^2 \right] \quad (2)$$

- \sum_{in} the number of edges (or sum of the weights) of links inside of community C
- \sum_{tot} is the number (or sum of the weights) of the edges connected to the nodes in C .
- k_i is the degree of node i
- $k_{i,in}$ is the sum of the edges (or edge weights) from nodes i to nodes in C

Practical Louvain Details

- Very fast, scalable, method. Works for most things if your graph is relatively sparse and or structured.
- Code: <https://pypi.org/project/louvain/>
- A very good bet to get the job done quickly....
- You do not need to specify the number of communities. The default resolution parameter is $\gamma = 1$.
- A limitation is that it only allows for a hard partition of nodes.

All was Fine and Good Until they Realized a little Quirk

Louvain can produce communities that are internally disconnected. That is, the shortest path between a pair of nodes in the same community may require actually leaving the community.

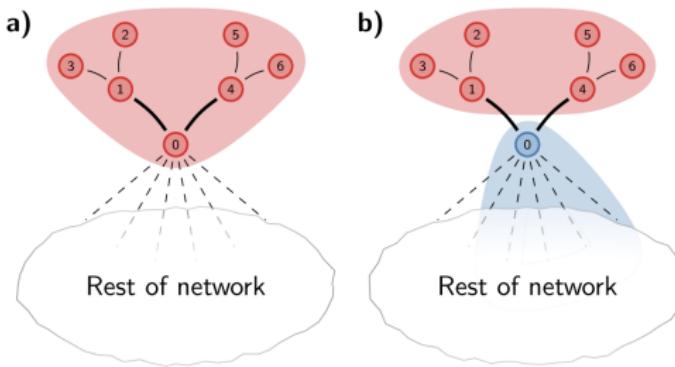


Figure: from Traag *et al.* Scientific Reports. 2018.

Overview of Leiden

Leiden makes a few modifications to guarantee well-connected communities.

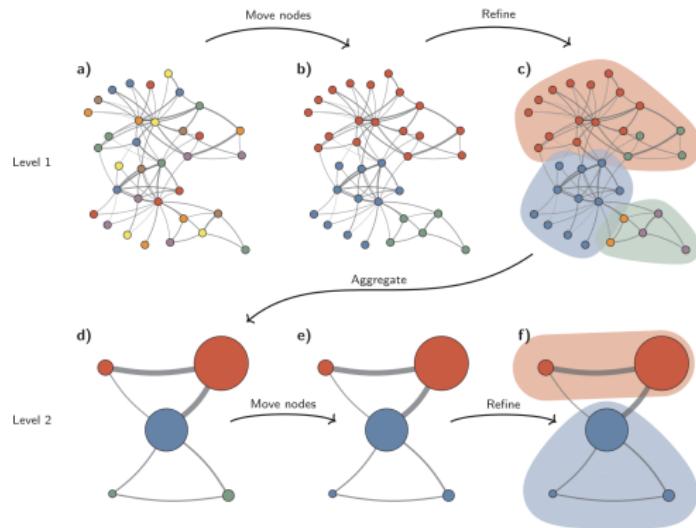


Figure: from Traag et al. Scientific Reports. 2018.

Methods of Clean-Up (1)

- **Smart Local Move:** Regular Louvain update of a merge, such that objective function cannot be increased
 - Before aggregating, consider the possibility of splitting each newly formed community
 - Gives the opportunity to split a subset of the newly formed community

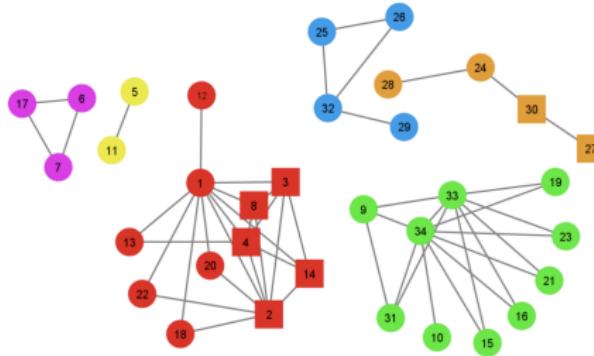


Figure: from <https://arxiv.org/pdf/1308.6604.pdf>

Methods of Clean-Up (2)

- **Random Moving:** For a particular node chosen at random, look at its neighbor in a different community and see if there is a gain in modularity if moving to that community
 - Intuition : The best community to join is the one where most of your neighbors are.

Punchline: Leiden will find higher quality communities (e.g. better connected) in less time than Louvain.

Code: <https://github.com/vtraag/leidenalg>

scipy.tl.leiden %

```
scipy.tl.leiden(adata, resolution=1, *, restrict_to=None, random_state=0, key_added='leiden', adjacency=None, directed=True, use_weights=True, n_iterations=1, partition_type=None, neighbors_key=None, obsp=None, copy=False, **partition_kwarg)
```

Cluster cells into subgroups [Traag18].

Cluster cells using the Leiden algorithm [Traag18], an improved version of the Louvain algorithm [Blondel08]. It has been proposed for single-cell analysis by [Levine15].

This requires having ran `neighbors()` or `bbknn()` first.

A Word of Caution for Single-Cell People

It is incorrect to say 'I clustered by single-cell data with scanpy' or to say 'I clustered by single-cell data with Seurat.' Look at which graph partitioning approach they were using!

Indeed, there are less disconnected communities under leiden...

Depending on your application, this is important. At some point though if your graph gets large enough, what do we think?

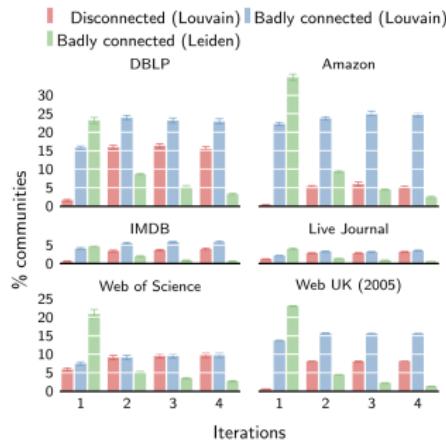


Figure: from Traag et al. Scientific Reports. 2018.

Badly connected → two nodes in the same community can be connected by walking outside of the community.

A Question

Can anyone think of a situation where disconnected communities might still be valid to consider?

Leiden Also Increases the Maximum Observed Modularity

	Nodes	Degree	Max. modularity	
			Louvain	Leiden
DBLP	317,080	6.6	0.8262	0.8387
Amazon	334,863	5.6	0.9301	0.9341
IMDB	374,511	80.2	0.7062	0.7069
Live Journal	3,997,962	17.4	0.7653	0.7739
Web of Science	9,811,130	21.2	0.7911	0.7951
Web UK	39,252,879	39.8	0.9796	0.9801

Figure: from Traag *et al.* Scientific Reports. 2018.

Modularity is more of a big picture score. I think whether you care about modularity or disconnectedness depends on your application.

Free Project Idea

A free idea for a course project: The authors define here a very particular notion of quality community. Are there other quality definitions. How does this observation change with different kinds of networks?

- You can find many of the standard networks people using for benchmarking and more on SNAP
<https://snap.stanford.edu/data/>
- You can also use the biological networks in Choobdar *et al.*
<https://www.nature.com/articles/s41592-019-0509-5>

Partitions of Biological Graphs

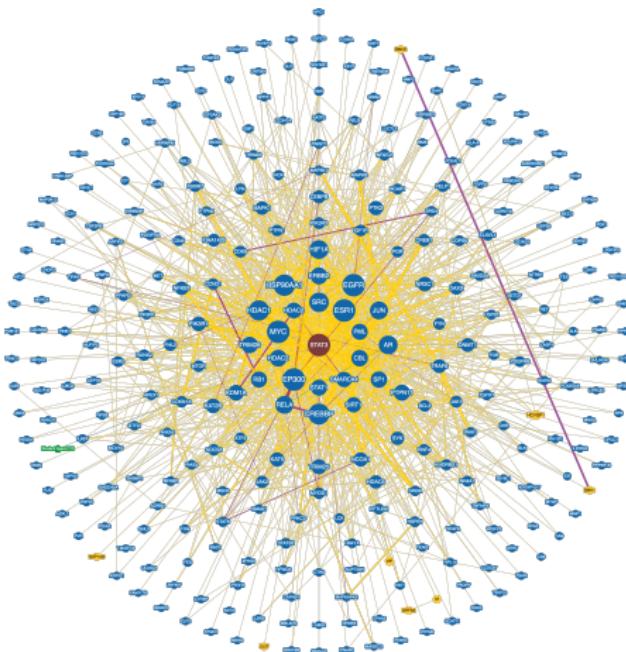


Figure: We are interested in the interactions with STAT3. These can be measured in different ways (gene expression, protein expression, physical interaction)

How Does Modularity-Based Optimization do in the Biological Network Benchmarking Challenge?

Here there is a really nice ‘ground truth’ understanding, which is gene and protein interactions linked to particular diseases.

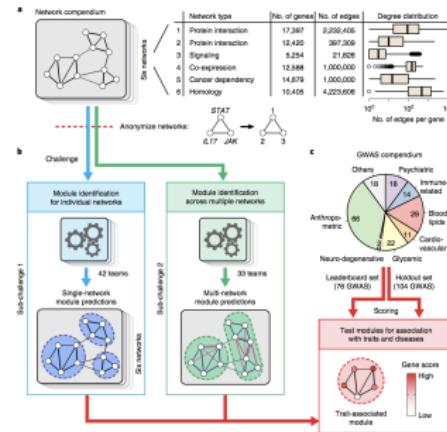
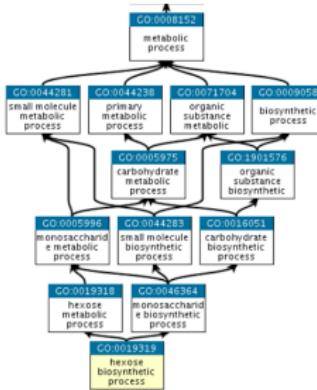


Figure: from Choobar et al. Nature Methods 2018.

Good Places to Look for Ground Truth to Validate Your Methods

- Gene Ontology → genes that are involved in the same pathway.
- MSigDB → genes related to certain categories (cancer, immune system, regulatory targets, etc.), text mined
- KEGG → metabolism
- Panther → genes and proteins



It seems that modularity is not only numerically useful

We use modularity to effectively score a candidate partition. This experiment shows that modularity is indeed correlated with the identification of biologically meaningful modules.

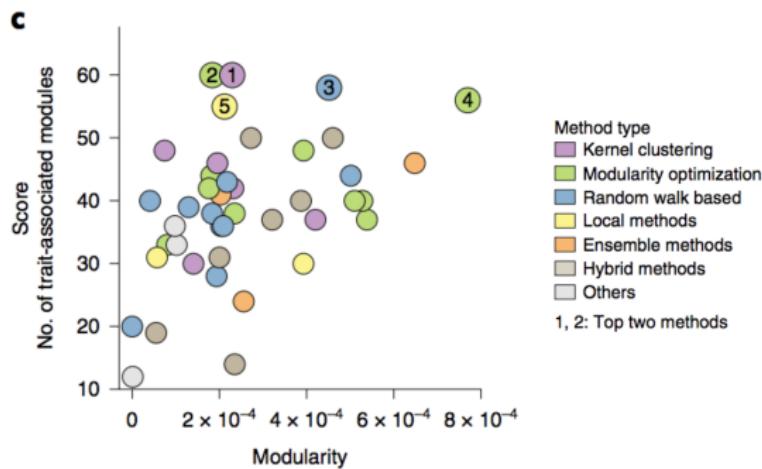


Figure: from Choobar *et al.* Nature Methods 2018.

Ranking Methods Based on Biological Relevance.

'M1' is a variant of what we have seen with Louvain. Interested readers can read more here, <https://arxiv.org/abs/physics/0703218>

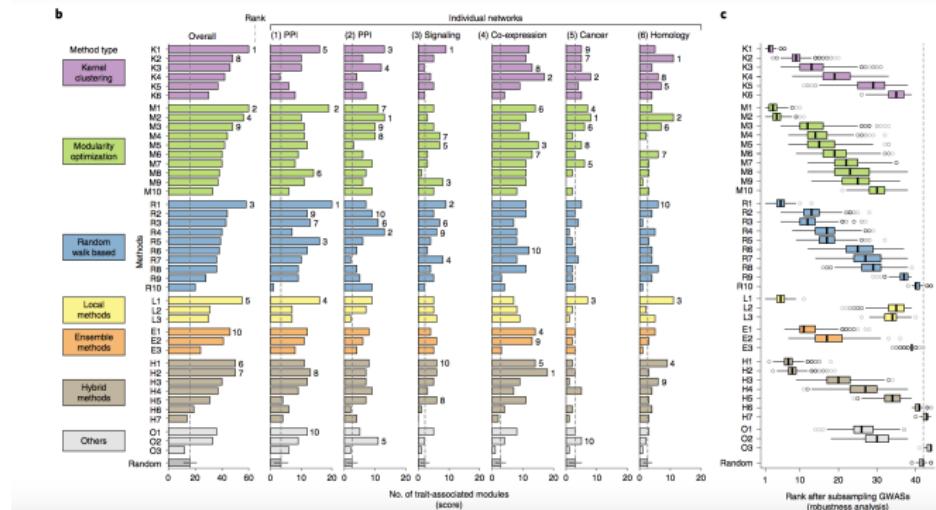


Figure: from Choobar *et al.* Nature Methods 2018.

Examples of Biologically-Relevant Modules

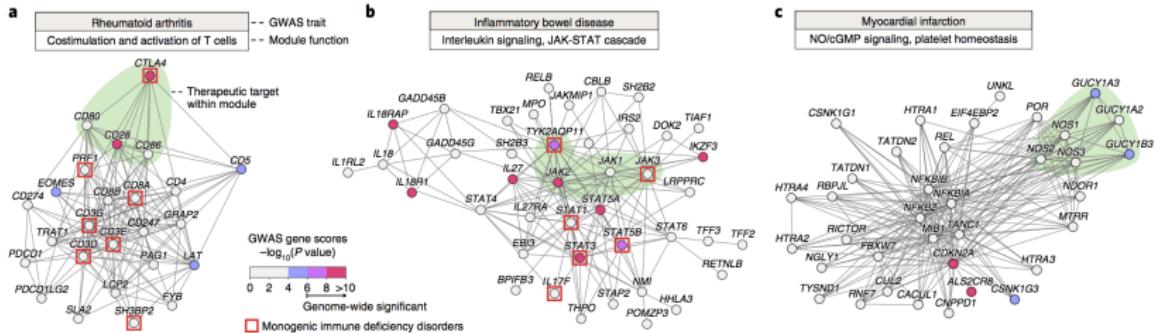


Figure: from Choobar *et al.* Nature Methods 2018.