Applying Graph Theory

Intro to Graph Centrality Analysis

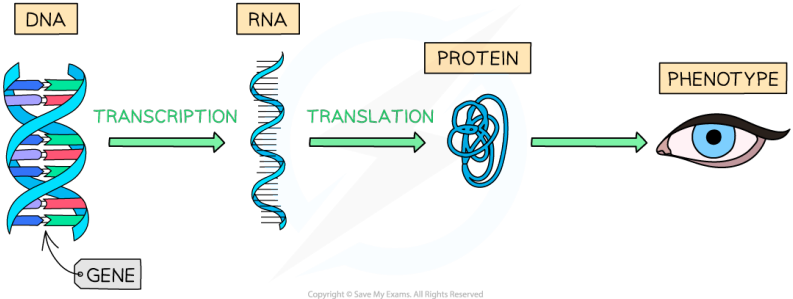
CPMSoc x BINFSoc Education

Attendance form :D

Who is BINFsoc

At BINFSOC, our mission is to educate, unite, and inspire students from bioinformatics as well as computer science, statistics, biology and chemistry disciplines, to create a network of future thinkers and innovators.

Some brief biology



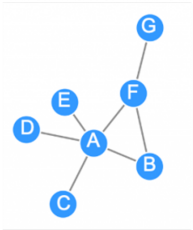
Networks as graphs

Networks describing the relationships between objects can be quantified as Graphs. Graphs are comprised of

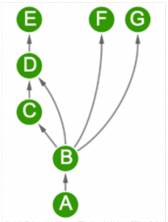
- nodes (drawn as dots) - representing objects
- and edges (lines connecting the dots) - representing relationships.

It is common to find graphs with edge attributes such as, direction or weight. For simplicity we will only discuss undirected, unweighted, connected graphs today.

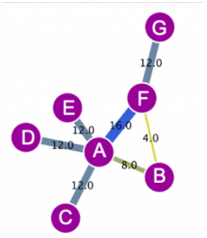
Undirected



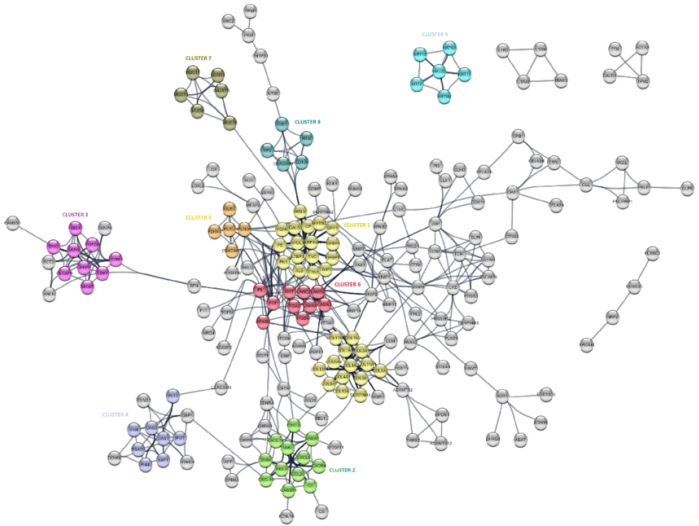
Directed



Weighted

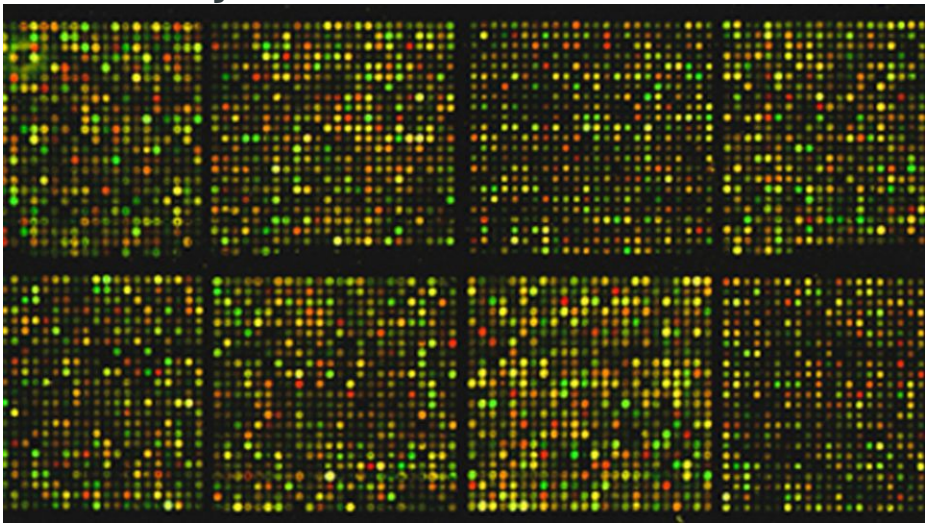


Protein-Protein interaction network

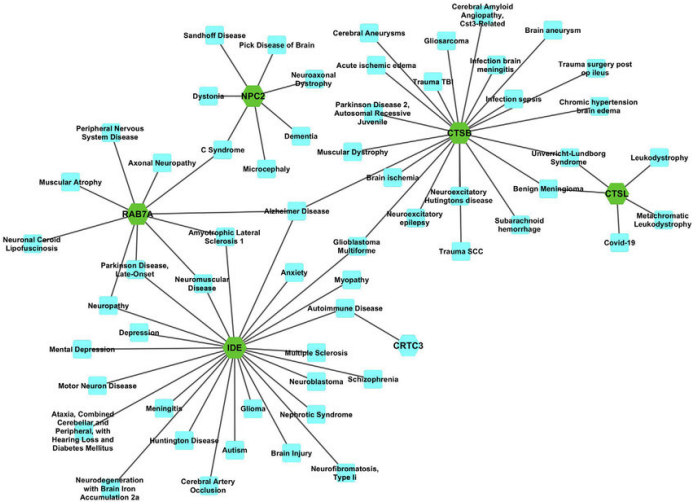


Atay, Sevcan. (2020). Integrated transcriptome meta-analysis of pancreatic ductal adenocarcinoma and matched adjacent pancreatic tissues. PeerJ. 8. 10.7717/peerj.10141.

Microarrays



Disease-gene interaction network



Prasad, Kartikay & Ahamad, Shahzaib & Gupta, Dinesh & Kumar, Vijay. (2021). Targeting cathepsins: A potential link between COVID-19 and associated neurological manifestations. Heliyon. 7. e08089. 10.1016/j.heliyon.2021.e08089.

Centrality Analysis

Centrality gives an estimation of how important a node or edge is for the connectivity or flow of the network. In interaction networks where each edge between nodes indicates the ability or extent for nodes to interact.

What 'important' means will vary depending on the network and user objectives.

Networks and importance...

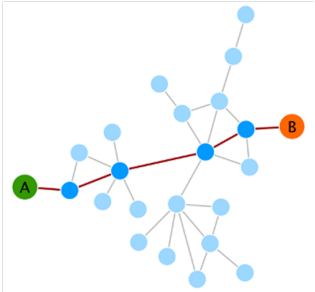
Given that an 'important' property of protein interaction networks may be the importance of that protein for keeping you alive. What might some other 'important' properties of nodes in the following networks be?

- Drug interaction networks?
- Disease interaction networks?
- Transport network? (with roads or railways as edges)
- Social networks?
- Professional networks?
- The world wide web...
  - as an information network?
  - as a social network?

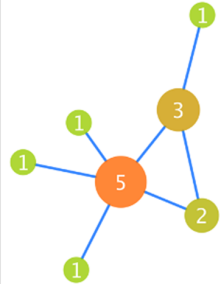
Understanding why these features are 'important' is helpful to then define ways to measure each type of 'importance' - centrality

Graphs: Network Topology

Graph topology refers to the particular arrangement of nodes and edges within a graph. These properties can be relevant when deriving conclusions about the behaviour of an aspect of the network. e.g. the shortest path between two nodes, or the node degree.



Topological property — Shortest path.



Topological property — Node degree.

Centrality measures

Representing networks as graphs with nodes holding the 'important' property, given some sufficiency condition for the relationship between the 'important' property and network properties, we can construct a ranking of the nodes where the important property is quantified by some measure on the edges between the nodes.

- Are those sufficient conditions? - often it doesn't matter
- How might we come up with an appropriate measure that quantifies a nodes importance using the information?

Centrality measures

We can combine through mathematical operators e.g.addition, subtraction... any number of topological properties, and other centrality measures, to create new centrality measures.

Note: measures can be constructed to fit certain 'importance' rankings or other existing data. However if this is done without grounding you lose the insight gained from a centrality measure approach and you might as well just build a regression model.

What are some measures that you can think of?

Can you justify what this might represent for an example network?

Centrality measures

For a graph of nodes of viral diseases[1] and genes with associations as edges, then naively the inverse of the node degree could be used to identify which diseases are most approachable[2] to study[3]. i.e. the least complex disease

[1] restricted to viral diseases since viral diseases 'only' manipulate genetic material so their entire behaviour can be described through disease-gene association networks

[2] in terms of needing to study fewer genes to completely understand the disease

[3] In case you haven't studied much biology before, general statements have exceptions, so don't expect this to necessarily hold true

Direct translation

Consider a graph with nodes representing users or web-pages, and edges representing clicks. If the popularity of web page nodes is the 'importance' that might be used to know how to display search results... Where popularity is defined as the number of individuals who view something(by having at least one click) ...the degree centrality is the obvious choice to rank nodes by.

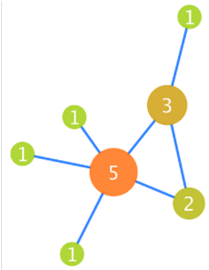


Figure: Toy graph with degree centrality of nodes

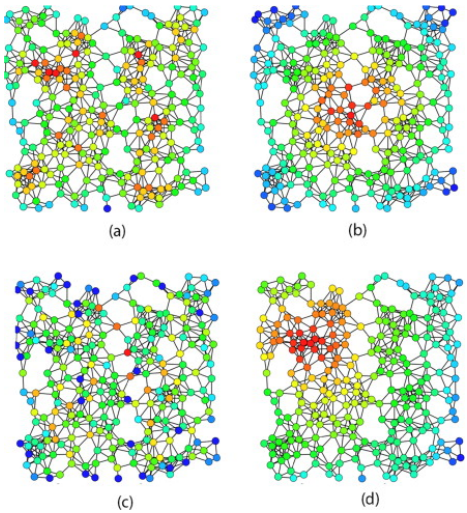
What changes

For that graph with nodes representing users or webpages, edges representing clicks, and popularity of a thing defined by the number individuals that view that thing. How would the approach and analysis change when the graph being studied with has

- hyperlinks instead of webpages?
- hyperlinks instead of users?
- edges denoting _ instead of clicks?
  - downloads
  - uploads
  - hover time
- popularity of a thing defined by users that care, instead of viewing, the thing?

This leads to a desire for interesting data but also a need to understand the data.

Some common centrality measures



In the context of social networks:

(a) Degree centrality shows people with many social connections.

(b) Closeness centrality shows people with many social connections.

(c) Betweeness centrality shows people with many social connections.

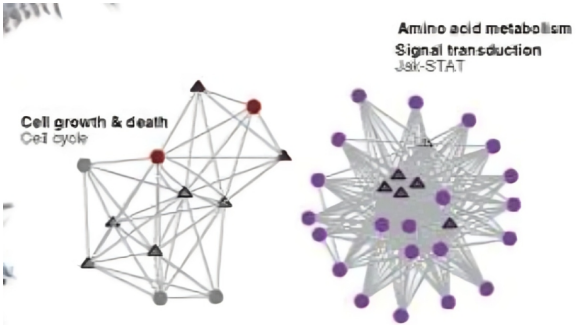
(d) Eigenvector centrality shows people with many social connections.

Right is the same network shown four times. Color coding indicates centrality according for different measures. Red nodes are more central and blue nodes are less central.[a]

[a] Golbeck, J. (2015). Introduction to social media investigation: A Hands-on Approach. Syngress.

Any thoughts?

Now we have some idea of centrality measures, take a look at the below protein interaction networks where cancer proteins are shown as triangles - is there anything we can say about cancerous proteins? in each or both networks? Could this imply anything about the behaviour of cancer?



Protein communities in human interactome. Cancer proteins are shown as triangles. From Josson, P. F., and Bates, P. A. (2006). Global topological features of cancer proteins in the human interactome. Bioinformatics 22, 22917.

Degree Centrality

We start with a graph G, which we want to calculate the degree centrality of. G has nodes each numbered from 1 to N. For each index $1 \leq i \leq N$, we need to calculate the node degree. Mathematically, we have

$$\deg(i) = \sum_{j \in \mathrm{adj}(i,n)} 1, i \neq j$$

($\mathrm{adj}(i,n)$ is the set of neighbours of node i with radius $n \geq 1$), and so the degree centrality of G is

$$\mathrm{DC}(G) = \{\deg(1), ..., \deg(N)\}$$

. The degree centrality DC produces an ordered list indexed by node.

Degree Centrality

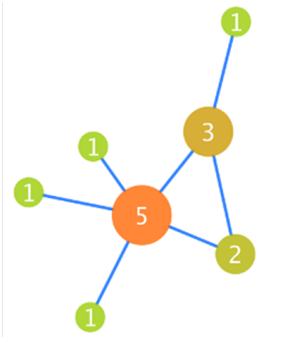


Figure: Topological property — Node degree.

Here is a sample dataset of a subset of the NSW train network for you to use.
Dataset: https://shorturl.at/C5klr

Sample Solution

```
template <size_t N>
inline int deg(int graph[N][N], int i, int n)
{
    std::set<int> visited; visited.emplace(i);
    std::queue<std::pair<int, int>> q;
    q.push(std::make_pair(i, 0));
    int degree = 0;

    while (!q.empty())
    {
        auto cur = q.front();

        for (int j = 0; j < N; j++)
        {
            if (cur.first != j &&
                graph[cur.first][j] &&
                cur.second < n)
            {
```

Sample Solution

```
                    q.push(std::make_pair(j, cur.second + 1));
                    if (!visited.count(j))
                    {
                        degree += (int)(bool)graph[cur.first][j];
                        visited.emplace(j);
                    }
                }
            }

            printf("%d\n", cur.first);

            q.pop();
        }

        return degree;
    }
```

Sample Solution

, gives us the node degree, and ...

```
template <size_t N>
inline std::vector<int> DC(int graph[N][N])
{
    std::vector<int> ret; ret.reserve(N);
    for (int i = 0; i < N; i++)
        ret.push_back(deg(graph, i));
    return ret;
}
```

, gives us the degree centrality, as required.

Closeness Centrality

Once again, we start with a graph G, and we want to determine the closeness centrality. The nodes of G are indexed $1 \leq i \leq N$.

For each node, we need to determine the shortest paths between the given node, and every other node. For brevity, we define $d(i, j)$ as the shortest path length between nodes i and j[4]. Thus, the closeness centrality at the node is

$$\mathrm{cc}(i) = \frac{N-1}{\sum_{j \in G} d(i,j)}, i \neq j$$

, and the closeness centrality of G is

$$\mathrm{CC}(G) = \{cc(1), ..., cc(N)\}$$

, as required. Dataset: https://shorturl.at/C5klr

[4]We could give you Djikstra's Algorithm for calculating shortest paths, what does the audience want?

Sample Solution

```
unordered_map<string, vector<pair<string, double>>> graph;

void addEdge(string u, string v, double weight) {
    graph[u].emplace_back(v, weight);
    graph[v].emplace_back(u, weight);
}

unordered_map<string, double> dijkstraShortestPaths(string src) {
    unordered_map<string, double> dist;
    for (auto node : graph) {
        dist[node.first] = 1e9;
    }
    dist[src] = 0;

    using Pair = pair<double, string>;
    priority_queue<Pair, vector<Pair>, greater<Pair>> pq;
    pq.emplace(0, src);
```

Sample Solution

```
        while (!pq.empty()) {
            double u_dist = pq.top().first;
            string u = pq.top().second;
            pq.pop();

            if (u_dist > dist[u]) continue;

            for (auto neighbor : graph.at(u)) {
                string v = neighbor.first;
                double weight = neighbor.second;
                if (dist[u] + weight < dist[v]) {
                    dist[v] = dist[u] + weight;
                    pq.emplace(dist[v], v);
                }
            }
        }

        return dist;
```

Sample Solution

```
        while (!pq.empty()) {
            double u_dist = pq.top().first;
            string u = pq.top().second;
            pq.pop();

            if (u_dist > dist[u]) continue;

            for (auto neighbor : graph.at(u)) {
                string v = neighbor.first;
                double weight = neighbor.second;
                if (dist[u] + weight < dist[v]) {
                    dist[v] = dist[u] + weight;
                    pq.emplace(dist[v], v);
                }
            }
        }

        return dist;
```

Sample Solution

```
    }

    unordered_map<string, double> calculateClosenessCentrality() {
        unordered_map<string, double> closenessCentrality;

        for (auto node : graph) {
            string u = node.first;
            unordered_map<string, double> shortestPaths = dijkstraShortestPaths(u);
            double totalDistance = 0.0;
            int reachableNodes = 0;

            for (auto dist : shortestPaths) {
                if (dist.second < 1e9) {
                    totalDistance += dist.second;
                    reachableNodes++;
                }
            }
```

Sample Solution

```
    }

    unordered_map<string, double> calculateClosenessCentrality() {
        unordered_map<string, double> closenessCentrality;

        for (auto node : graph) {
            string u = node.first;
            unordered_map<string, double> shortestPaths = dijkstraShortestPaths(u);
            double totalDistance = 0.0;
            int reachableNodes = 0;

            for (auto dist : shortestPaths) {
                if (dist.second < 1e9) {
                    totalDistance += dist.second;
                    reachableNodes++;
                }
            }
```

Sample Solution

```
    }

    unordered_map<string, double> calculateClosenessCentrality() {
        unordered_map<string, double> closenessCentrality;

        for (auto node : graph) {
            string u = node.first;
            unordered_map<string, double> shortestPaths = dijkstraShortestPaths(u);
            double totalDistance = 0.0;
            int reachableNodes = 0;

            for (auto dist : shortestPaths) {
                if (dist.second < 1e9) {
                    totalDistance += dist.second;
                    reachableNodes++;
                }
            }
```

Sample Solution

```
    }

    unordered_map<string, double> calculateClosenessCentrality() {
        unordered_map<string, double> closenessCentrality;

        for (auto node : graph) {
            string u = node.first;
            unordered_map<string, double> shortestPaths = dijkstraShortestPaths(u);
            double totalDistance = 0.0;
            int reachableNodes = 0;

            for (auto dist : shortestPaths) {
                if (dist.second < 1e9) {
                    totalDistance += dist.second;
                    reachableNodes++;
                }
            }
```

Further events

Please join us for:

- CSESoc x CPMSoc 2521 Revision Session Next Week, Time and Location TBD!

Further readings

- Cytoscape - graph analysis software
- Complex Networks by Estrada - more mathematics
- Genemania - returns an interaction network for a list of genes
- EMBL Network analysis of protein interaction data - comprehensive beginner tutorial for protein interaction network analysis
- SNAP, Stanford network analysis project - many interaction network datasets