Dynamics and Control of networks

Exam Topics

Contents

Lecture 1

1 Recount briefly the history of man-made networks.

In the 19$^{\text{th}}$ there was no theory for networks as there was no need for it. The development of technologies such as the power grid, telegraph, etc., was in an early stage, where the focus was on function of a single object rather than the topology of a whole network.

In the 20th century, the increase of motor and railroad vehicles lead to congestions. The need of finding places with increased traffic lead to the need of graph analysis tools.

The Internet relies on the topological structure of networks - small world property (top-heavy distribution).

2 Describe the general structure of a network.

A network consists of nodes connected by edges. We usually represent networks using graphs.

Definition 2.1: Graph

A graph $G(V, E)$ is a topological object, where V is a finite set of vertices and E is a set of graph edges. Vertices i, j are called adjacent if those are connected by an edge (i, j).

3 Name a few examples of physical, biological, social and information networks.

3.1 Physical

3.1.1 Power grids

- High-voltage lines
- Generating stations
- Switching substations
- Consumers

3.1.2 Transportation networks

- Roads, railroads, airline routes, shipping lanes
- Vertices represent geographical locations - road intersections, etc.
- Edges represent an existing link available for the considered type of transport - road, railway

- Differences between road networks and airline networks - Achieving connectedness with shortest paths of limited length while keeping the total number of edges as low as possible - relative weight give to those two opposing goals determines the topology of the network - emergence of hubs in airline network as opposed to road infrastructure

3.2 Biological

3.2.1 Neural network

- Vertices represent neurons
- Edges represent dendrites and axon synaptic interconnections

3.2.2 Food webs

- Vertices represent species and edges represent predator-prey relations in an ecosystem

3.3 Social

Social networks are comprised of people, groups, classes, companies, entities and their relations. Those exhibit opinion formation and propagation dynamics

- Affiliation networks - bipartite graph (groups and individuals) edges signify which individuals belong to which group

3.4 Information

3.4.1 Telecommunication networks

- A group of nodes interconnected by telecommunication links that are used to exchange messages between the nodes.
- Telecom links can be point-to-point, broadcast or multipoint using circuit switching, message switching or packet switching
- Today, the most important telecom network is the Internet

4 Describe how the structure of various networks is revealed empirically.

The empirical approach consists of 2 stages: observation and analysis.

The observations are done using web crawlers - a breadth first search which discovers different websites referred to from other websites using hyperlinks.

The raw topological data are analyzed to reveal properties, patterns - centralities, degree distributions, clustering, assortativity

Lecture 2

5 Define a binary relation and establish a connection with its graph.

A binary relation R is a subset of $V \times V$ - a Cartesian square of set $V - R \subseteq V \times V$
Classical graphs depict binary relations - nodes represent elements of V, edges signify pairs of elements in a given binary relation.

6 Tell the difference between a graph and a hyper-graph. Depict a given hyper-graph by a bipartite network.

A **hypergraph** $HG(V, E)$ is a pair of disjoint sets V and E, where the elements of E are non-empty subsets of V having arbitrary cardinality.
Hypergraphs can represent general relations, as can bipartite graphs.

A graph $G(V, E)$ is called a **bipartite** graph if its vertex set V can be partitioned into two disjoint classes V_1, V_2, where $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$, so that all edges of G have exactly one vertex in V_1 and exactly one V_2.

A graph is bipartite, if and only if it does not contain an odd cycle.

Affiliation graphs, as bipartite graphs, have two sets of distinct vertices. Vertices of one type, signifying the original elements of V, are affiliated with vertices of the other type, signifying classes of elements in V.

$$HG(V, W, E),\ E \subseteq V \times W, \tag{6.1}$$

where elements of the edge set E are ordered couples $(i, j) \in E, i \in V, j \in W$, signifying that the element i belong to the class j.
Membership of vertices to classes is indicated by the **vertex class incidence matrix** $B \in \mathbb{R}^{|V| \times |W|}$

$$B_{ij} = \begin{cases} 0 \\ 1, \text{ if vertex } j \text{ belongs to the group } i \end{cases} \tag{6.2}$$

7 Explain the difference between a multi-graph and a simple graph.

A simple graph allows no multi-edges, no self-loops

8 Define the graph adjacency matrix.

The adjacency matrix $A \in \mathbb{R}^{n \times n}$ is defined as

$$A_{ij} = \begin{cases} 1 \text{ if } (i, j) \in E \\ 0 \text{ if } (i, j) \notin E \end{cases} \tag{8.1}$$

nonzero elements do not need to be unity, but they are usually positive.
Unweighted graphs have binary adjacency matrices and weighted graphs have nonnegative adjacency matrices.

9 Compare the co-citation and bibliographic coupling.

9.1 Co-citation

$$C_{ij} = \sum_k A_{ik}A_{jk} - \text{number of vertices to which both the } i^{\text{th}} \text{ and the } j^{\text{th}} \text{ vertices point} \tag{9.1}$$

9.2 Bibliographic coupling

$$B_{ij} = \sum_k A_{ki}A_{kj} - \text{number of vertices which point both to the } i^{\text{th}} \text{ and the } j^{\text{th}} \text{ vertices} \tag{9.2}$$

10 Explain the two one-mode projections of a bipartite network.

$$P_{ij} = \sum_k B_{ki}B_{kj}, \quad P = B^T B, \tag{10.1}$$

where P_{ij} give the number of classes vertices i and j both belong to, with P_{ii} being the number of classes which vertex i alone belongs to.
Alternatively

$$P'_{ij} = \sum_k B_{ik}B_{jk}, \quad P' = BB^T, \tag{10.2}$$

where P'_{ij} gives the number of vertices that belong both to classes i and j, with P'_{ii} being the number of vertices that belong to class i.

11 Define planar networks and state the 'four color' theorem.

Planar network has a graph that can be drawn in a plane without any of its edges intersecting. Not all drawings of the graph need to have non intersecting edges but there needs to exist at least one.
For a planar graph the maximal number of colors required to accomplish vertex coloring, with no adjacent nodes having the same color, is 4.

12 Explain the difference between a sparse and a dense network.

The mean degree of a network is defined as

$$c = \frac{1}{N}\sum_{i=1}^{N} d_i \tag{12.1}$$

A network is sparse if for $n \to \infty$ the mean degree is bounded $c < \infty$, otherwise the network is sparse.

Lecture 3

13 Define a path and explain the difference between an Eulerian and a Hamiltonian path.

Definition 13.1: Path

A path is any sequence of connecting edges such that every consecutive pair of vertices in the sequence is connected by and edge in the sense of the edge direction.

13.1 Eulerian path

An Eulerian path is a path visiting each edge in a graph only once.
Each Eulerian path must leave a vertex it enters, except of the starting and ending vertex, therefore there must be 0 or exactly 2 vertices with an odd degree.

13.2 Hamiltonian path

A Hamiltonian path is a path visiting each vertex in the graph only once.
It is necessarily self-avoiding. Finding of such path is more difficult than finding of a Eulerian path.

14 State the min-cut max-flow theorem.

Definition 14.1: Cut set

An edge cut set is a subset of graph's edges whose removal disconnects a given pair of vertices
A vertex cut set is a subset of graph's vertices whose removal disconnects a given pair of vertices.

Definition 14.2: Maximum flow

Given a unit flow carried by each edge, what is the maximal flow through the whole network between a given pair of vertices.

Theorem 14.3: Max flow min cut

aximal flow equals the cardinality of the minimum edge cut set times the maximal throughput of each edge.
If each edge carries unit flow then the maximal flow between a given pair of vertices equals the edge connectivity of those two vertices.

15 Define the graph Laplacian matrix and its relation to diffusion processes on graphs.

Given the conventional nonnegative adjacency matrix of a graph $A \in \mathbb{R}^{n\times n}$, the **graph Laplacian matrix** $L \in \mathbb{R}^{n\times n}$ is defined as

$$L = D - A, \tag{15.1}$$

where $D = \operatorname{diag}(d_1, \ldots, d_n) \geq 0$ is a non-negative diagonal matrix of vertex in-degrees.
Similarly, the out-Laplacian is defined using vertex out-degrees and the transposed Adjacency matrix.
The sign-less graph Laplacian matrix $Q \in \mathbb{R}^{n\times n}$ is defined as

$$Q = D + A. \tag{15.2}$$

By construction $L\bar{1}_n = 0$, therefore 0 is always an eigenvalue of the Laplacian matrix and the Laplacian matrix is always singular. The sign-less Laplacian Q does not share this property.
The zero eigenvalue fo the graph Laplacian may be simple or multiple, depending on the graph topology.

15.1 Diffusion processes

One of the applications of the Laplacian matrix is description of diffusions. The Fourier equation for discrete places is

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = c\sum_j A_{ij}(u_j - u_i), \tag{15.3}$$

where $u_i(t)$ stands for countably many spatially sample values of the diffusion field $u(x,t)$.

In matrix notation that is

$$\dot{\boldsymbol{u}} = -L\boldsymbol{u}. \tag{15.4}$$

This allows for linear dynamical analysis.

16 State the Geršgorin disc theorem and explain how it is applied to the graph Laplacian.

Let $A = (a_{ij})$ be an $n \times n$ complex matrix. For each row i, define the Geršgorin disc

$$D_i = \{\, z \in \mathbb{C} : |z - a_{ii}| \leq R_i \,\}, \tag{16.1}$$

where

$$R_i = \sum_{j\neq i} |a_{ij}|. \tag{16.2}$$

Then every eigenvalue of A lies in the union of these discs:

$$\sigma(A) \subseteq \bigcup_{i=1}^{n} D_i. \tag{16.3}$$

Moreover, if the union of k of these discs is disjoint from all the others, then that union contains exactly k eigenvalues of A, counted with algebraic multiplicity.

The center of each disc is a value on the diagonal of the Laplacian. The diagonal values of the adjacency matrix A are zero $a_{ii} = 0$, and every element of the degree matrix D is nonnegative $D_{ii} \geq 0$, therefore the diagonal values of the Laplacian matrix are $L_{ii} \geq 0$. This means that all of the centers of the discs lie in the right half-plane.
Furthermore, each row sum without the diagonal element $\sum_{j \neq i} |a_{ij}|$ is exactly the degree of the vertex $d_i = D_{ii}$.
This means that each disc has a center in the right half-plane and the disc is tangent to the imaginary axis. The discs boundaries intersect in 0.
The Laplacian is therefore always positive semi-definite.

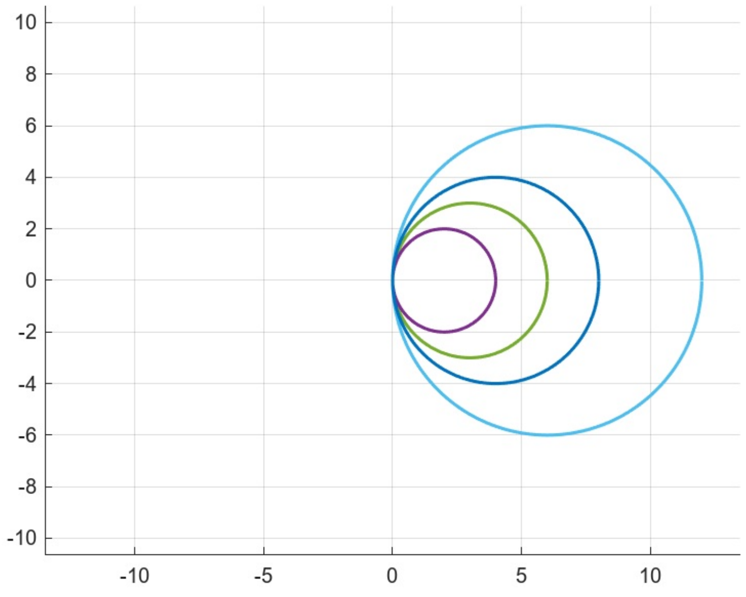

Figure 1: Geršorin disc theorem applied to the graph Laplacian

17 Define the Fiedler eigenvalue and explain its significance for the graph topology.

Definition 17.1: Fiedler eigenvalue

If zero is a simple eigenvalue of the Laplacian matrix L, the Fiedler eigenvalue $\lambda_2(L) > 0$ (or $\Re(\lambda_2) > 0$) in directed graphs.

It is the smallest Laplacian's non-zero eigenvalue. It gives the spectral gap of L - the value of which is proportional to the overall graph connectivity.
The Fiedler eigenvalue is often referred to as algebraic connectivity of the graph.

18 Define the Frobenius form of the graph Laplacian and explain how it reveals the graph topology.

There exists a permutation of the vertex labels that results in a block triangular Laplacian matrix. The permutation is performed using an orthogonal transformation matrix T, $T^{-1} = T^T$.

$$T^T L T = \begin{bmatrix} L_{11} & \dots & L_{1k} & L_{1(k+1)} & \dots & L_{1p} \\ 0 & \ddots & \vdots & \vdots & \vdots & \\ 0 & 0 & L_{kk} & L_{k(k+1)} & \dots & L_{kp} \\ 0 & 0 & 0 & L_{(k+1)(k+1)} & \dots & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & L_{pp} \end{bmatrix}, \tag{18.1}$$

The Frobenius form shows that

- Strongly connected graph is sufficient for zero to be a simple eigenvalue of the graph Laplacian
- Existence of a spanning tree is necessary and sufficient for zero to be a simple eigenvalue, this zero eigenvalue stems from the root irreducible component
- In general case of a spanning forest, the multiplicity of zero eigenvalue equals the minimum number of trees which together span all vertices

If zero is a simple eigenvalue of L, the Fiedler eigenvalue is smallest non-zero eigenvalue. The value of the Fiedler eigenvalue is proportional to the overall graph connectivity.

Lecture 4

19 Define the Metzler matrix.

Metzler matrix is a matrix, where all off-diagonal components are nonnegative, $M_{ij} \geq 0,\ i \neq j$. A dynamical system modelled by a Metzler matrix $\dot{x} = Mx$ has its dynamical trajectories contained in the positive orthant $\mathbb{R}^n_+$, if the initial values are from the orthant.
The matrix can be written as

$$M = -sI + A, \text{ where} \tag{19.1}$$

$$s \in \mathbb{R}, s \geq 0, A \succeq 0. \tag{19.2}$$

20 Define the M-matrix.

The Z-matrix has all off-diagonal elements non-positive, $Z_{ij} \leq 0,\ i \neq j$.
$-Z$ is Metzler.

A M-matrix is a Z-matrix with all eigenvalues having non-negative real parts.

- A Z-matrix $E = sI - A$ is singular M-matrix if all its principal minors are nonnegative
- A Z-matrix $E = sI - A$ is nonsingular M-matrix if all its principal minors are positive
- If E is a singular M-matrix, then $s \geq \rho(A)$ and $-E$ is a Metzler matrix
- ...

21 Explain how the geometric multiplicity of eigenvalues differ from the algebraic one.

21.1 Algebraic multiplicity

Definition 21.1: Algebraic multiplicity of an eigenvalue

Algebraic multiplicity of the eigenvalue λ is how many times it appears as a root of the characteristic polynomial

$$p(s) = \det(sI - A). \tag{21.1}$$

21.2 Geometric multiplicity

Definition 21.2: Geometric multiplicity of an eigenvalue

Geometric multiplicity of the eigenvalue λ is the dimension of the eigenspace

$$gm(\lambda) = \dim(\ker(A - I\lambda)). \tag{21.2}$$

It gives the amount of linearly independent eigenvectors associated with the eigenvalue λ.

22 Explain the meaning of the geometric multiplicity of a zero eigenvalue for graph Laplacians.

By construction, the Laplacian matrix always has at least one zero eigenvalue. The amount of zero eigenvalues of the graph Laplacian is the amount of spanning trees in the spanning forest of the graph = number of strongly connected components of the graph.

23 Define the left zero eigenvector.

For any graph

$$L = D - A, \tag{23.1}$$

then

$$L\mathbf{1} = 0. \tag{23.2}$$

There must also exist

$$\exists p^T,\ p^T L = 0, \tag{23.3}$$

a left eigenvector corresponding to the eigenvalue 0. The elements p_i of the left eigenvector p are nonnegative

$$p_i \geq 0, \quad p \succeq 0. \tag{23.4}$$

The elements which are greater than 0 correspond to nodes of the graph which can be roots of a spanning tree. For a strongly connected graph $\forall p_i,\ p_i > 0$.

A weight-balanced graph has the following property

$$d_i^{in} = d_i^{out} \tag{23.5}$$

$$L = L^T. \tag{23.6}$$

Then it is true that

$$p^T = \mathbf{1}^T, \tag{23.7}$$

$$\mathbf{1}^T L = 0. \tag{23.8}$$

24 Define a balanced graph.

A balanced graph is a graph where the indegree and outdegree of each vertex are equal. An example of a balanced graph is an undirected graph.

25 Define a strongly connected graph.

A strongly connected graph is a graph where there exists a directed path between any 2 vertices of the graph.

26 Define the degree centrality.

Degree centrality is a measure of the graph.

$$d_i = \sum_j A_{ij} = A\mathbf{1} \tag{26.1}$$

27 Motivate the eigenvector centrality.

Definition 27.1: Eigenvector centrality

Eigenvector centrality measures the importance of a node as a function of the importance of its neighbors. If a node is connected to highly important nodes, it will have a higher eigenvector centrality compared to the less important nodes.

$$x'_i = \sum_j A_{ij} x_j \tag{27.1}$$

$$x' = Ax \tag{27.2}$$

The algorithm of obtaining the eigenvector centrality of a graph is iterative.

- Choose an initial seed $x(0) = c^T V$, where V is the matrix of eigenvectors of the adjacency matrix and c is a weighing vector.
- Iterate

$$x'_i = \frac{1}{\lambda_1} \sum_j A_{ij} x_j, \tag{27.3}$$

where λ_1 is the maximal eigenvalue, the division is done to prevent blow-up or convergence to zero.

28 Explain the difference between the eigenvector centrality and the Katz centrality.

Definition 28.1: Katz centrality

Generalization of the eigenvector centrality

$$x_i = \alpha \sum_j A_{ij} x_j + \beta, \tag{28.1}$$

which allows to calculate the centrality in a non-iterative manner

$$x = (I - \alpha A)^{-1} + \bar{\beta} \tag{28.2}$$

$$\bar{\beta} = \mathbf{1}\beta. \tag{28.3}$$

Lecture 5

29 Define PageRank and modified PageRank.

29.1 PageRank

Definition 29.1: PageRank

Modified version of Katz centrality - normalized by its out-degree. Nodes pointing out to many other nodes have their importance proportionally discounted.

$$x_i = \alpha \sum_j A_{ij} \frac{x_j}{d_j^{out}} + \beta \tag{29.1}$$

$$x = D^{out} \left(D^{out} - \alpha A\right)^{-1} \mathbf{1}\beta \tag{29.2}$$

$$D^{out} = \operatorname{diag}(\max\left(d_i^{out}, 1\right)) \tag{29.3}$$

29.2 Modified PageRank

Definition 29.2: Modified PageRank

$$x_i = \alpha \sum_j A_{ij} \frac{x_j}{d_j^{out}} + \beta_i \tag{29.4}$$

$$x = D^{out} \left(D^{out} - \alpha A\right)^{-1} \bar{\beta} \tag{29.5}$$

$$D^{out} = \operatorname{diag}(\max\left(d_i^{out}, 1\right)) \tag{29.6}$$

30 Explain hubs and authorities and their relation to co-citation and bibliographic coupling.

Definition 30.1: Authories and Hubs

An important authority has a lot of useful Hubs pointing to it.

$$x_i = \alpha \sum_j A_{ij} y_j \tag{30.1}$$

A useful Hub points to a lot of important Authorities.

$$y_i = \beta \sum_j A_{ij} x_j \tag{30.2}$$

In vector form

$$x = \alpha A y, \qquad y = \beta A^T x \tag{30.3}$$
$$x = \alpha A \beta A^T x, \qquad y = \beta A^T \alpha A y \tag{30.4}$$
$$AA^T x = (\alpha\beta)^{-1} x, \qquad A^T A y = (\beta\alpha)^{-1} y. \tag{30.5}$$

Meaning that the resulting hub and authority centralities are given by the eigenvectors of matrices AA^T and $A^T A$.

31 Define closeness centrality and discuss different variants thereof.

31.1 Closeness centrality

Definition 31.1: Closeness centrality

Closeness centrality defines the importance of a node in a graph as being measured by how close it is to all other nodes in the graph. It is a local path-based measure

$$l_i = \frac{1}{n} \sum_j d_{ij}, \tag{31.1}$$

where d_{ij} is the geodesic distance between the pair of vertices (i, j). Alternatively $l_i = \frac{1}{n-1} \sum_j d_{ij}$ to exclude the origin vertex under consideration from the total count.
The values of this measure have a small range, therefore its inverse is more numerically meaningful

$$c_i = \frac{1}{l_i} = \frac{n}{\sum_j d_{ij}}. \tag{31.2}$$

The closeness centrality is often redefined as

$$c_i' = \frac{1}{n-1} \sum_{j \neq i} \frac{1}{d_{ij}} \tag{31.3}$$

32 Define betweenness centrality.

Definition 32.1: Betweenness centrality

Betweenness centrality measures the importance of a node in a graph based upon how many times it occurs in the shortest path between all pairs of nodes in a graph.

$$x_i = \sum_{s,t} n_{st}^i, \tag{32.1}$$

where $n_s t^i = 1$ if vertex i lies in the geodesic path between vertices s and t, and 0 otherwise.
If we define the total number of geodesic paths between vertices s and t as g_{st}, we can

modify the definition of betweenness centrality as

$$x_i = \sum_{s,t} \frac{n^i_{st}}{g_{st}}, \tag{32.2}$$

33 Explain the difference between cliques, plexes and cores.

33.1 Clique

Definition 33.1: Clique of size n

clique of size n is a maximal set of n verices in an undirected graph such that all its members are connected to all its other members via single edges. Every 2 vertices in a clique are adjacent.

33.2 k-plex

Definition 33.2: k-plex

k-plex (of size n) is a maximal subset of n vertices such that each one member is connected to at least $n - k$ others in that subset.

33.3 k-core

Definition 33.3: k-core

A k-core (of size n) is a maximal subset of n vertices such that each one member is connected to at least k others in that subset

- k-core $= (n - k)$-plex

33.4 Independent subset

Definition 33.4: Independent subset

A subset of vertices is termed independent if any pair of its elements are not adjacent.

33.5 Coclique

Definition 33.5: Coclique

A coclique (of size n) is a set of n vertices no two of which are adjacent. A coclique is an independent, stable, subset of graph vertices.
A maximum coclique is a maximal independent set, the cardinality of which is the graph independence number. A subset of graph vertices is independent, if and only if those same vertices comprise a clique in the graph complement.

33.6 k-clique

Definition 33.6: k-clique

A k-clique is a maximal set of vertices such that each member is no more than k edges away from any other member.

33.7 Component

Definition 33.7: Component

A subset of graph vertices where each 2 vertices are connected by a path

33.8 k-component

34 Define k-components and compare them with connected components.

Definition 34.1: k-component

A subset of graph vertices where each 2 vertices are connected via at least k vertex independent paths - paths with no common vertex other than start and end

- A 1-component is a regular component

Lecture 6

35 Define transitivity of a network in any of equivalent ways.

Transitivity measures how strongly a network tends to form triangles - weather "a fried of my friend is also my friend".

Definition 35.1: Transitivity

Transitivity is a global measure defined using any of the following formulas

$$C = \frac{\text{number of closed paths of length 2}}{\text{number of paths of length 2}} \tag{35.1}$$

(35.2)

This formula shows a probability that two neighbors of the same node are connected. Among all two-step paths $A - B - C$, how many are closed by an edge $A - C$?

$$C = \frac{\text{number of triangles} \times 6}{\text{number of paths of length 2}} \tag{35.3}$$

(35.4)

Each triangle can have its vertices ordered in 6 ways. It has the same meaning as the previous definition.

$$C = \frac{\text{number of triangles} \times 3}{\text{number of connected triples}} \tag{35.5}$$

(35.6)

A single triangle can be written as 3 connected triples, one centered at each node. The formula is then: how many closed triples among all triples.

36 Define local clustering.

$$C_i = \frac{\text{number of pairs of neighbors of vertex } i \text{ that are connected}}{\text{number of pairs of neighbors of vertex } i} \tag{36.1}$$

37 Explain the concept of reciprocity and how it relates to loops of length two.

Definition 37.1: Reciprocity

A global measure of the graph r

$$r = \frac{1}{m} \sum_{i,j} A_{ij} A_{ji}. \tag{37.1}$$

Equals to the number of 2-loops normalized to the number of edges.
It gives a probability that if there is an edge (i, j) there is also an edge in the opposite direction (j, i).

38 Define structural balance for networks with signed edges. Show that structurally balanced network is certainly clusterable (Harrary's theorem).

Signed edges carry a positive or negative sign.
A graph is structurally balanced if all loops in the graph have even number of negative edges.
A signed network is clusterable if it is possible to clearly partition it into two subsets of vertices such that all edges between vertices within each subset are positive while all the edges between the two subsets are negative.
Clustering algorithm - color the vertices along paths using 2 colors, change the color when traversing a negative edge. After finishing the loop, the color changed even times - so it has changed to the original color of the first vertex in the loop.

39 Show by counterexample that a clusterable network need not be structurally balanced.

A triangle with every edge negative. Each vertex is a cluster of 1 node. There are 3 clusters.

Note

Definition above defines a 2-clusterable network, but during the lecture, the counterexample was this one exactly, with three clusters

40 Explain vertex similarity.

A measure defined between two vertices. There are two types of similarity

- Structural equivalence
- Regular equivalence

41 Define structural and regular equivalence and explain the differences of them.

41.1 Structural equivalence

The amount of shared neighbors

$$n_{ij} = \sum_k A_{ik} A_{kj} = (A^2)_{ij}. \tag{41.1}$$

Intended use for undirected graphs.
Gives the number of length 2 paths between the ordered pair of nodes.

41.2 Regular equivalence

Outcome of an iterative process

$$\sigma_{ij} = \alpha \sum_{kl} A_{ik} A_{jl} \sigma_{kl} = \alpha \sum_{kl} A_{ik} \sigma_{kl} A^T_{lj} \tag{41.2}$$

$$\sigma = \alpha A \sigma A^T \tag{41.3}$$

42 Define homophily and assortative mixing in networks.

42.1 Homophily

A global property based on similarity

- Enumerative characteristics
- Scalar characteristics

It shows how likely are similar nodes to form ties.

42.1.1 Enumerative characteristics

Defines a finite number of distinct classes c_i.

We count the edges between the nodes of the same class

$$\sum_{i,j} \delta(c_i, c_j) = \sum_{i,j} A_{ij} \delta(x_i, c_j) \tag{42.1}$$

This number is compared to the number expected if connections were made at random

$$\frac{1}{2} \sum_{i,j} \delta(c_i, c_j). \tag{42.2}$$

The difference is

$$\frac{1}{2} \sum_{i,j} A_{ij} \delta(x_i, c_j) - \frac{1}{2} \sum_{i,j} \frac{d_i d_j}{2m} \delta(c_i, c_j). \tag{42.3}$$

The modularity is a global property defined as

$$Q = \frac{1}{2m} \sum_{i,j} \left(A_{ij} - \frac{d_i d_j}{2m} \right) \delta(c_i, c_j). \tag{42.4}$$

A new algebraic object nonsparse modularity matrix B describes network structure

$$B_{i,j} = A_{i,j} - \frac{d_i d_j}{2m}. \tag{42.5}$$

The maximal value of modularity equals

$$Q_{\max} = \frac{1}{2} \left(2m - \sum_{i,j} \frac{d_i d_j}{2m} \delta(c_i, c_j) \right) \tag{42.6}$$

42.1.2 Scalar characteristics

There are no distinct classes but rather a continuum of different values. Scalar characteristics relates to covariance of characteristics (value) found on both ends of edges in the graph. It measures how correlated are the values of the vertices at the ends of a single edge. For each edge $(i, j) \in E$ there are scalar values x_i, x_j assigned to vertices i, j. The mean value is

$$\mu = \frac{\sum_{i,j} A_{ij} x_i}{\sum_{i,j} A_{ij}} = \frac{\sum_i d_i x_i}{\sum_i d_i} = \frac{1}{2m} \sum_i d_i x_i \tag{42.7}$$

The covariance is defined as

$$\operatorname{cov}(x_i, x_j) = \frac{\sum_{i,j} A_{ij} (x_i - \mu)(x_j - \mu)}{\sum_{i,j} A_{ij}} \tag{42.8}$$

$$\operatorname{cov}(x_i, x_j) = \frac{1}{2m} \sum i, j \left(A_{ij} - \frac{d_i d_j}{2m} \right) x_i x_j. \tag{42.9}$$

42.2 Assortative mixing

For example: based on node in-degree. Shows how similar nodes (in this case nodes with the same in-degree) are connected together.

Assortatively mixed network will have a high in-degree nodes connected with each other. The remained of nodes will be poorly connected to each other and usually even to the core by some path.

Dissortatively mixed network will have high in-degree nodes (hubs) connected to vast amount of low in-degree nodes.

Lecture 7

43 Define assortative mixing with respect to scalar characteristics. Explain how does it differ from that for enumerative characteristics.

43.1 Scalar characteristics

The assortative coefficient for scalar characteristics is defined as

$$r = \frac{\sum_{i,j}\left(A_{ij} - \frac{d_i d_j}{2m}\right) x_i x_j}{\sum_{i,j}\left(d_i \delta_{ij} - \frac{d_i d_j}{2m}\right) x_i x_j} \tag{43.1}$$

43.2 Enumerative characteristics

The assortative coefficient for enumerative characteristics is defined as

$$\frac{Q}{Q_{\max}} = \frac{\sum\left(A_{ij} - \frac{d_i d_j}{2m}\right)\delta(c_i, c_j)}{2m - \sum \frac{d_i d_j}{2m}\delta(c_i, c_j)}. \tag{43.2}$$

The difference is that enumerative characteristics allows for a finite set of distinct classes, while scalar characteristics allows for a continuum of values and calculates the correlation between the values at the ends of edges rather then whether the classes are the same.

44 Define the modularity matrix for undirected networks.

$$B_{i,j} = A_{i,j} - \frac{d_i d_j}{2m.} \tag{44.1}$$

It assumes an undirected graph. The matrix is always nonsparse regardless of the graph topology and it is used to calculate both scalar and enumerative characteristics.

45 Explain assortative mixing with respect to the degree. Discuss what are its implications on the network topology.

Assortatively mixed networks will have high in-degree nodes well connected to each other-comprising the network's core. The remaining low in-degree nodes are poorly connected to each other but usually connected to the core via some path.

Dissortatively mixed networks usually consist of high-degree hubs weakly connected to each other but each connecting into many low in-degree nodes.

46 Define the power law distributions, explain how to calculate their moments.

A statistical characterization of a large network uses power-law degree distributions.

Probability p_k of finding a vertex of degree $d_i = k$ is defined as

$$p_k \sim k^{-\alpha} \ln p_k \qquad \sim -\alpha \ln k + C. \tag{46.1}$$

The distribution allows for high number of low-degree nodes and very low number of high-degree nodes. This is the case of the Internet with a low number of hubs and high amount of users connecting to the hubs.

46.1 Moments

The expected value can be calculated as

$$\mathbb{E}\left[d_i\right] = \sum_{k=0}^{\infty} k p_k = \sum_k k k^{-\alpha} = \sum_k k^{1-\alpha}. \tag{46.2}$$

The sum converges for $\alpha > 2$.

47 Explain the effect of top heavy distribution of vertex degrees for the networks.

A minor proportion of highly connected vertices accounts for a sizeable portion of the graph edges. High clustering coefficient.
Such networks are remarkably robust to random node failure, but extremely susceptible to targeted attacks.

A fraction of edges attributed to the fraction p of highest degree vertices can be derived from the power law distribution

$$W = p^{\frac{\alpha-2}{\alpha-1}} \tag{47.1}$$

"If I sort the vertices by degree, take only the top 1% ($\frac{p}{100}$%) , how many edges will I have."

48 Explain the importance of the small world effect for the functionality of computer networks.

Definition 48.1: Small world networks

The length of the shortest path between any 2 vertices grows with the logarithm of n.

Appears in top-heavy distribution network. It is very important for the functionality of the Internet - in relatively reasonable amount of hops (10-15), you can reach any computer in the world.

Lecture 8

49 Explain the concept of algorithm complexity. Define the O notation.

Big O notation $O(n)$ expresses the leading order of duration for the worst possible case as a function of the input size n (the amount of elements, vertices, ...).

The $O()$ analysis only takes the highest order into account - $O(\alpha n^2 + \beta n + \gamma) = O(n^2)$.

Algorithms of complexity $O(n^3)$ and higher are hardly practical and would be unfeasible even for a few hundred/thousand nodes.
One should aim at $O(\log n) \leq O(n) \leq O(n \log n) \leq O(n^2) \leq O(n^2 \log n)$.

Log complexities usually appear when dealing with trees.

50 Explain how the network topology is represented in computer memory. Compare the adjacency matrix and the adjacency list.

- Adjacency matrix
- Adjacency list
- Adjacency tree
- Hybrid representation
- Heap

50.1 Adjacency matrix

It is the most straightforward method. Requires a lot of space for storing zeros, especially for a sparse graph.

Complexity

- Changes in the network - $O(1)$
- Calculation of degree d_i - $O(n)$, $O(n^2)$ to calculate all degrees

50.2 Adjacency list

The adjacency list is most useful in the majority of applications. Avoids storing 0 for nonexistent edges. It lists a vertex and its outgoing edges or incoming edges. Storing both doubles the space requirement but remains in the same complexity class.

The adjacency list has storage complexity $O(n \times m)$ which is less then $O(n^2)$ for sparse networks.

A modification of the adjacency also stores the in-degree which reduces computational complexity of many algorithms (for example degree centrality).

Complexity

- Addition of a new node - $O(1)$
- Deletion of a node - $O(\frac{m}{n})$
- Calculation of degree d_i - $O(\frac{m}{n})$, $O(m)$ to calculate all degrees

51 Define the tree data structure. What is a balanced tree and what is the worst case complexity of finding an element in it?

Useful for specific algorithms when a neighbor with the highest degree needs to be accessed immediately. Binary tree - each entry in this data structure has none, one or two children entries Stores lists of neighbors of a given vertex
A sorted binary tree has the highest degree value in the root and every child has lower degree than the parent.

Balanced tree - a sorted binary tree, for every node every lower valued node is in the left subtree and every higher valued node is in the right subtree

Complexity

- Finding a specific entry is at worst $O(k)$, where k is the depth of the tree. For a balanced tree $O(log(k))$.

52 Explain how trees are used to represent networks.

A node stores all its neighbors in a tree structure. For every node, there is a unique tree. The edges in the tree do not represent edges in the network. It is used when for example a neighbor with a highest degree needs to be accessed in $O(1)$.

53 Define a binary heap. Why would one use such a structure in network algorithms?

Based on a tree. Each entry consists of a label and a value, this value is less than any of its children. Uses dynamic memory allocation. Adding or removing a node has complexity $O(\log n)$. Reading the value of the smallest element has complexity of $O(1)$ as it always is at the root of the heap.

For example, Dijkstra's algorithm benefits from being able to access a node with smallest distance estimate.

Lecture 9

54 Describe the breadth-first search algorithm.

BFS is used for finding the shortest distance between a pair of nodes. The shortest distance is a nonlocal property.

- In a single run for a given vertex the algorithm returns the shortest distance to all other vertices in the same connected component
- There exists no other procedure to calculate the shortest distances between a given pair of vertices in the same connected component faster in the worst case
- The algorithm also returns the geodesic path

54.1 The algorithm

1. Given a starting vertex s - the distance to s is 0, the distances to other vertices are unknown
2. All neighbors of s are given a distance of 1
3. All of their neighbors are assigned distance of 2 if they have not been assigned a distance before
4. Repeat until there are no unvisited vertices

55 Explain the computational complexity of the naive implementation of the breadth-first search.

- Uses an array of vertex distances of length n and distance counter d. At each step finds vertices of distance d and look for their neighbors, update the distances which are presently unknown
- Maximal number of iterations is r - which is limited by the diameter of the graph. In the very worst case, the depth is n
- In each iteration, you need to go through all n values of the vertex distance array with complexity $O(n)$
- The amount of neighbors of a single vertex is on average m/n. Using an adjacency list, finding the neighbors takes $O(m/n)$. We need to find the neighbors of all vertices so the complexity is $O(nm/n) = O(m)$.
- The total complexity is then $O(rn + m)$
- In the worst case where $r = n$ the complexity is $O(n^2 + m)$.

56 Explain the computational complexity of the more sophisticated implementation of the breadth-first search using a buffer.

Using a FIFO buffer (Queue) data structure, we only store vertices at distance $d+1$ we reduce the complexity of finding vertices at distance d

- Finding all neighbors of a single vertex is the same $= O(m/n)$ - in total $O(m)$
- In each iteration, we only need to go through vertices at distance d so across all iterations, the algorithm goes over n vertices in total. The complexity of finding vertices at a distance reduces to $O(n)$.
- Total complexity $O(n+m)$
- On a sparse network $m \sim n$ results in $O(n)$
- On a dense network $m \sim n^2$ - $O(n^2)$

57 Describe how to find the actual shortest paths. Assume first single shortest paths, then generalize to possible multiple shortest paths.

Note

Complexities are not required

- Construct a directed network representing the shortest paths
- Start by creating a new graph having the same number of vertices with the same labels and 0 edges
- Start BFS for s. Each time a neighboring vertex $j \in N_i$ is examined add an edge (j,i) which results in a tree, a path leading to s can be found by following the parents of nodes. Adding the edge has complexity $O(1)$ in an adjacency list
- The algorithm requires the standard $O(n+m)$ run of BFS the path reconstruction then takes $O(k)$, where $k \leq n$ is the length of the path. So at worst it is $O(n+m+n) = O(n+m)$.
- For multiple paths - $O(n \log n)$

57.1 Method for multiple paths

- When exploring $u \to v$
- If v is unvisited, set its distance to $dist(u)+1$ and add u to v's parents
- If v is visited and its distance is equal $dist(u)+1$ add u to v's parents
- Otherwise ignore the node, the discovered path would have been longer

This creates a directed graph, which is not necessarily a tree, in this directed graph, you can find all paths using recursion.

58 Describe Dijkstra's algorithm, explain why it works and analyze its computational complexity.

- Uses an array of n elements to contain current weighted distance estimates from a given starting node s
- Estimates an upper bound on the shortest weighted distance - if the estimate is true then it must equal the actual shortest weighted distance
- Distance estimate of the node s from the node s is initialized to 0 all others are set to infinity
- Another array records if one is certain that a particular distance to the given vertex is indeed the smallest possible initially all are set to False

1. Find a vertex $v \in V$ that has the smallest weighted distance estimate from s. This estimate does not need to be certain at this point
2. Set it to certain
3. Calculate distance estimates from $s \to v \to i \forall i \in N_v$. If any of these is smaller than the current estimate for that neighbor, replace the old estimate with the newer one.
4. Repeat from step 1 until all distances are certain

58.1 Complexity

58.1.1 Naive implementation

- Search through n vertices to find the one with shortest distance estimate from s - takes $O(n)$
- On average each iteration goes through m/n neighbors - $O(m/n)$
- In the worst case the algorithm needs n iterations - total complexity $O(n^2 + m)$

58.1.2 Improved implementaiton

- Distance estimates are stored in a binary heap - reading and removing an item with the smallest value takes $O(\log n)$
- In each iteration estimates of $O(m/n)$ vertices are taken and then in the worst case it takes $O(\log n)$ to replace each of the estimate in the heap
- Each iteration then takes $O(\log n + \frac{m}{n} \log n)$
- The amount of iterations in the worst case remains n so the total complexity is $O((n + m) \log n)$

59 Explain how to apply Dijkstra's algorithm to find the actual least weight path tree for a given starting vertex.

- Maintains a shortest path tree
- Adds an edge each time an estimate distance less than infinity is assigned
- Move the edge to point to a different vertex each time one finds an estimate lower than the current one
- The last edge position gives the true shortest path
- If two successive estimates are equal add two directed edges implying alternative paths

60 Explain how to find the betweenness centrality of a given vertex using either the breadth-first search or Dijkstra's algorithm.

To find the betweenness centrality of a vertex v for every distinct pair of vertices s, t find the shortest path and check how many of these contain v

61 Describe the augmenting path algorithm (Ford-Fulkerson algorithm).

An algorithm to calculate a maximum flow in a graph with capacities on edges. A flow $f(u, v)$ must satisfy

- $0 \leq f(u, v) \leq c(u, v)$, where $c(u, v)$ is the capacity of a given edge,
- $\sum_{(u,v)\in E} f(u, v) = \sum_{(v,w)\in E} f(v, w)$.

The algorithm assigns all edges zero flow, then repeatedly, using BFS/Dijkstra, finds paths to push more flow from s to t

Given a current flow f, the residual capacity is r

$$r(u, v) = c(u, v) - f(u, v) \text{ the residual capacity says how much more flow can be pushed through the edge.} r(v, \tag{61.1}$$

The flow of a path is given by a bottleneck Δ- the minimal capacity of the edges in the path. We update the flow $f(u, v)$ of edges (u, v) on the path as $f(u, v) = f(u, v) + \Delta$. On the backward edges we update it as $f(v, u) = f(v, u) - \Delta$. The backward edge may remove previously assigned flow of the forward edge to increase the total flow.

Given the residual capacities, we can create a residual graph, which contains the same vertices but and edge (u, v) only of $r(u, v) > 0$. Then we search for paths on the residual graph and repeat until there are no more paths from s to t.

Lecture 10

62 Describe the power method for finding leading eigenvalues and eigenvectors. What is the importance of choosing the initial seed?

Definition 62.1: Leading eigenvalues and eigenvectors

For a square matrix $A \in \mathbb{R}^{n\times n}$, an eigenvalue-eigenvector pair (λ, v) satisfies

$$Av = \lambda v, \quad v \neq 0. \tag{62.1}$$

The leading eigenvalue means the eigenvalue with the largest magnitude

$$|\lambda_1| = \max_i |\lambda_i|. \tag{62.2}$$

The corresponding eigenvector is the leading eigenvector v_1.

A seemingly easy way of finding such pair is to find all eigenvalues and choose the one with the largest magnitude. This however requires too much computing power and is rather wasteful.

62.1 Power method for finding leading eigenvalues and eigenvectors

The core idea of the power method is a repeated multiplication of a vector by A. Components in the direction of the dominant eigenvector grow fastest in magnitude, so after normalization the iterate aligns with v_1.

62.1.1 The algorithm

1. Pick an initial vector $x_0 \neq 0$ and normalize it
2. For $k = 0, 1, \dots$:
   - Multiply: $y_{k+1} = Ax_k$
   - Normalize: $x_{k+1} = \frac{y_{k+1}}{\|y_{k+1}\|}$
   - Estimate eigenvalue: $\mu_{k+1} = x_{k+1}^T A x_{k+1}$
   - Stop when either $\|x_{k+1} - x_k\|$ or $\|Ax_{k+1} - \mu_{k+1}x_{k+1}\|$ are small (bellow tolerance)

62.2 Importance of initial value

Let he eigenvalues of A be ordered $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$. In each iteration the product is roughly

$$x_k = A^k x_0 \approx \lambda_1^k(\text{component along } v_1) + \lambda_2(\text{component along } v_2) + \dots. \tag{62.3}$$

The ratio $\frac{\lambda_2^k}{\lambda_1^k}$ gradually diminishes, only leaving the v_1 component. However, for that to happen the initial vector x_0 has to have a nonzero component along v_1. If the initial vector were orthogonal to v_1, the algorithm will fail.

63 Explain how to efficiently find all eigenvalues and eigenvectors of a given matrix. Specify which algorithms are used for matrix transformation and efficient solution of the transformed eigen-problem, given different starting matrices (symmetric/asymmetric, sparse/dense).

In general we use similarity transforms and apply efficient algorithms on the matrices in a special form.

For symmetric matrices $A = A^T$ there exists an orthogonal matrix Q, $Q^{-1} = Q^T$, such that $Q^T AQ = T$, where T is a tridiagonal matrix - nonzero elements are only on the diagonal and on both subdiagonals. For general matrices a similar orthogonal transformation exists, such T is a Hessenberg matrix. If $Av_i = \lambda_i$ then

$$\lambda_i Q^T v_i = Q^T Av_i = Q^T AQQ^{-1} v_i = (Q^T AQ)Q^T v_i \tag{63.1}$$

$$(Q^T AQ)Q^T v_i = TQ^T v_i, \text{ therefore} \tag{63.2}$$

$$Tw_i = \lambda_i w_i, \text{ where} \tag{63.3}$$

$$w_i = Q^T v_i. \tag{63.4}$$

There exist efficient numerical algorithms for finding eigenvectors and eigenvalue of triagonal or Hessenberg T matrices.

- QL algorithm - complexity $O(n)$ for a triagonal matrix, $O(n^2)$ for a Hessenberg matrix

Algorithms for finding the transformation matrix Q

- For a general symmetric - Hauseholder algorithm - $O(n^3)$
- For a sparse symmetric - Lanczos algorithm - $O(nm) \sim O(n^2)$
- For an assymetric matrix - Arnoldi algorithm

64 Why to use heuristic algorithms in general, even if no proof of correctness is available?

Heuristic algorithms return a result which is not guaranteed to be best for all cases, but for most practical purposes it is often good enough. It returns a fairly good divisions - approximate but acceptable solutions - no proof of validity is available
We use heuristic algorithms, when it comes to computationally difficult problems - either the algorithm runs fast but fails to find the best solution almost always, or it always finds the best solution but takes prohibitively long time to return the result
Characteristically for heuristic algorithms, this assertion is not rigorously proven, hence it remains a conjecture, albeit one that points directly to the presumed fundamental difference between P and NP type problems

64.1 P vs NP hard problem

A P hard problem has a polynomial complexity - $O(n)$, $O(n^2)$, $O(\log n)$, etc. An NP hard problem has a non-polynomial complexity - algorithm fails almost always or takes prohibitively very long time - for example $O(2^n)$

65 What is the difference between the graph partitioning and the community detection problems?

65.1 Graph partitioning

In graph partitioning, we are dividing graph vertices into a given number of non-overlapping groups of a fixed size. **Number of inter-group edges is minimized**.

Motivated by task allocation in distributed computing - tasks which rely on each other should run on a single processor - faster communication within a single processor. Applications in network process simulations on parallel computers

Usually bi-partitioning - two clusters with sizes n_1 and n_2.

65.2 Community detection

Number and sizes of groups are not given but are result of an algorithm. It is primarily used as a tool for analysis and understanding network data. The criterion of division can be defined in various ways, through the extent of modularity. Reveals hidden structures in a network. **The goal is also to minimize inter-group edges**.

66 Describe the Kernighan-Lin algorithm for graph partitioning. What is its computational complexity? What is roughly the size of the network for which it can be reasonably expected to work?

A heuristic algorithm for graph partitioning.

1. Divide vertices into 2 groups V_1 and V_2 of required sizes n_1, n_2 in any way
2. For all pairs $i, j \in V_1, V_2$ calculate the change in cut set size between the groups if the vertices are interchanged. The cut set is the set of inter-group edges.
3. From all such pairs find the one $(i, j)^*$ that reduces the cut set size the most, or in absence of any such, the one that increases the cut size the least
4. Swap the pair - this preserves assigned sizes of both groups
5. Repeat the process from step 2 with the exception that the moved pairs cannot be moved again in this round
6. Stop when there are no more pairs to swap
7. When all swaps are completed - select from all partitions the one with the smallest cut set
8. Repeat the whole process with this partition. Stop when there is no improvement

66.1 Complexity

- Number of swaps in each round is $\min(n_1, n_2) \in [0, \frac{n}{2}]$ resulting in $O(n)$ in the worst case
- For each swap the amount of pairs is $\frac{n^2}{2}$ in the worst case resulting in $O(n^2)$
- For each examined swap the reduction in cut set size is calculated $O(\frac{m}{n})$
- The total complexity for one round is $O(mn^2)$ which is $O(n^3)$ on sparse networks and $O(n^4)$ on dense networks
- It is applicable for $n \leq 10^3$

67 Describe the spectral partitioning algorithm. Explain the importance of the graph Laplacian matrix and its Fiedler eigenvalue. What is its computational complexity? What is roughly the size of a network for which it can be reasonably expected to work?

Warning

Proof/Derivation is required as a part of the exam

- Spectral graph partitioning is a method for graph bipartition
- Assumes an undirected graph - $A = A^T$
- The sizes n_1, n_2 of the 2 clusters $\mathcal{V}_1, \mathcal{V}_2$ are given, the clusters are disjoint and they mutually exhaust the entire vertex set $\mathcal{V}$.
- The cut set size equals

$$R = \frac{1}{2}\sum_{i,j} A_{ij}, \text{ where} i \in \mathcal{V}_1 \wedge j \in \mathcal{V}_2 \tag{67.1}$$

Define a vector $s \in \mathbb{R}^n$ with components s_i

$$s_i = \begin{cases} +1 \text{ if } i \in \mathcal{V}_1, \\ -1 \text{ if } i \in \mathcal{V}_2. \end{cases} \tag{67.2}$$

With the help of s_i vertex labels one can construct the expression

$$\frac{1}{2}(1 - s_i s_j) = \begin{cases} 1 \text{ if } (i,j) \text{ vertices are in different groups} \\ 0 \text{ if } (i,j) \text{ vertices are in the same group} \end{cases} \tag{67.3}$$

We can then express the size of the cut set as

$$R = \frac{1}{2}\sum_{i,j} A_{ij}\frac{1}{2}(1 - s_i s_j) = \frac{1}{4}\sum_{i,j} A_{ij}(1 - s_i s_j). \tag{67.4}$$

The sum of elements of A can be written as

$$\sum_{i,j} A_{ij} = \sum_i d_i = \sum_i d_i s_i^2, \text{ because } s_i^2 = 1 \tag{67.5}$$

$$\sum_i d_i s_i^2 = \sum_{i,j} d_i \delta_{ij} s_i s_j. \tag{67.6}$$

This will give us the size of the cut set as

$$R = \frac{1}{4} \sum_{i,j} (d_i \delta_{ij} - A_{ij}) s_i s_j. \tag{67.7}$$

Using the definition of the graph Laplacian

$$L = D - A \tag{67.8}$$

$$L_{ij} = D_{ij} - A_{ij} = d_i \delta_{ij} - A_{ij}, \tag{67.9}$$

we can write the cut set size as

$$R = \frac{1}{4} \sum_{i=1}^{n} L_{ij} s_i s_j \tag{67.10}$$

$$R = \frac{1}{4} s^T L s. \tag{67.11}$$

The goal of the algorithm is to minimize $R = \frac{1}{4} s^T L s$, which is a restricted quadratic optimization problem. The values of s lie on a hyper-cube.
This problem unfortunately cannot be solved by calculating the jacobian as with standard quadratic optimization problems. It is also not feasible to evaluate the criterion function for every vector s as the amount of possible values grows with 2^n.

The problem has additional constraints

$$\sum_i s_i = n_1 - n_2 \tag{67.12}$$

$$\sum_i s_i^2 = n. \tag{67.13}$$

We can now relax the problem and constraint the values of s on a surface of a hyper-sphere with a diameter of $\sqrt{n}$. This relaxes the problem from restricted optimization to constrained optimization. Such problem can be solved using Lagrangian optimization method

$$\frac{\partial}{\partial s_i} \left[\sum_{jk} L_{jk} s_j s_k + \alpha \left(n - \sum_j s^2 \right) + 2\beta \left(n_1 - n_2 - \sum_j s_j \right) \right] = 0, \; \forall i 2 \sum_j L_{ij} s_j - 2\alpha s_i - 2\beta = 0 \tag{67.14}$$

$$Ls - \alpha s - \beta \mathbf{1} = 0. \tag{67.15}$$

Using the left eigenvector of the Laplacian, we can solve for β

$$\mathbf{1}^T L s - \alpha \mathbf{1}^T s - \beta \mathbf{1}^T \mathbf{1} = \tag{67.16}$$

$$0 - \alpha(n_1 - n_2) - \beta n = 0 \tag{67.17}$$

$$\beta = \alpha \frac{n_2 - n_1}{n}. \tag{67.18}$$

We can now substitute for β

$$Ls - \alpha s - \alpha \frac{n_2 - n_1}{n}\mathbf{1} = 0 \tag{67.19}$$

$$Ls = \alpha \left(s + \frac{n_2 - n_1}{n}\mathbf{1} \right). \tag{67.20}$$

The Laplacian has the following property $L\mathbf{1} = 0$ so we can write

$$L \left(s + \frac{n_2 - n_1}{n}\mathbf{1} \right) = \alpha \left(s + \frac{n_2 - n_1}{n}\mathbf{1} \right). \tag{67.21}$$

This allows us to define a new vector x

$$x = s + \frac{n_2 - n_1}{n}\mathbf{1} \tag{67.22}$$

$$Lx = \alpha x. \tag{67.23}$$

Therefore x is the eigenvector to the α eigenvector to the Laplacian L.

This means that the cut size is

$$R = \frac{1}{4} s^T L s = \frac{1}{4} x^T L x = \frac{1}{4} \alpha x^T x, \tag{67.24}$$

since s and x by $\frac{n_2 - n_1}{n}\mathbf{1}$ which is in kernel of L.

Then

$$x^T x = \left(s + \frac{n_2 - n_1}{n}\mathbf{1} \right)^T \left(s + \frac{n_2 - n_1}{n}\mathbf{1} \right) \tag{67.25}$$

$$x^T x = s^T s + \frac{n_2 - n_1}{n} \left(s^T \mathbf{1} + \mathbf{1}^T s \right) + \left(\frac{n_2 - n_1}{n} \right)^2 \mathbf{1}^T \mathbf{1} \tag{67.26}$$

$$s^s = n, \; s^T \mathbf{1} + \mathbf{1}^T s = 2(n_1 - n_2), \; \mathbf{1}^T \mathbf{1} = n \tag{67.27}$$

$$x^T x = n - 2\frac{(n_2 - n_1)^2}{n} + \left(\frac{n_2 - n_1}{n} \right)^2 n \tag{67.28}$$

$$x^T x = n - 2\frac{(n_2 - n_1)^2}{n} + \frac{(n_2 - n_1)^2}{n} \tag{67.29}$$

$$x^T x = n - \frac{(n_2 - n_1)^2}{n} = \frac{n^2 - (n_2 - n_1)^2}{n} \tag{67.30}$$

$$x^T x = \frac{(n_2 + n_1)^2 - (n_2 - n_1)^2}{n} = \frac{n_2^2 + 2n_1 n_2 + n_1^2 - n_2^2 + 2n_1 n_2 - n_1^2}{n} \tag{67.31}$$

$$x^T x = \frac{4n_1 n_2}{n}. \tag{67.32}$$

Finally, the cut set size is

$$R = \frac{1}{4} \alpha \frac{4n_1 n_2}{n} = \alpha \frac{n_1 n_2}{n}. \tag{67.33}$$

To minimize the size of the cut set, we need to find the eigenvector corresponding to the minimal eigenvalue, but also

$$\mathbf{1}^T x = \mathbf{1}^T s + \frac{n_2 - n_1}{n} \mathbf{1}^T \mathbf{1} = (n_1 - n_2) + (n_2 - n_1) = 0. \tag{67.34}$$

Choosing the zero eigenvalue of the Laplacian, the corresponding eigenvector is $\mathbf{1}$, but then

$$\mathbf{1}^T x = \mathbf{1}^T \mathbf{1} = n \neq 0. \tag{67.35}$$

Therefore, we need to choose the second smallest eigenvealue and the corresponding eigenvector. Such pair is called the Fiedler eigenvalue and the Fiedler eigenvector.

However, after you find the optimum, you need to find actual restricted values of $s_i = \pm 1$ closest to the nonrestricted optimum $s = x - \frac{n_2 - n_1}{n}\mathbf{1}$, that is

$$\text{maximize} \quad s^T\left(x - \frac{n_2 - n_1}{n}\mathbf{1}\right) = \sum_i s_i\left(x_i - \frac{n_2 - n_1}{n}\right). \tag{67.36}$$

Such optimum is achieved when we assing $s_i = +1$ to the n_1 largest components of x_i and $s_i = -1$ to the rest.

67.1 Spectral Partitioning Algorithm

1. Calculate the Fiedler eigenvector $v \in \mathbb{R}^n$ of L
2. Sort the elements $v_2 i$ from largest to smallest
3. Assign n_1 most positive elements to V_1 and the remaining elements to V_2
4. Assign n_1 most negative elements to V_1 and the remaining elements to V_2
5. Identify which partition results in smaller cut set size

67.2 Complexity

- Finding the Fiedler eigenvector can be done with $O(mn)$ - on a sparse network $O(n^2) < O(n^3)$ of Kernighan-Lin
- However, the cut sets returned are usually larger than KL method
- It is applicable for $n \leq 10^5$
- The smaller is the Fielder eigenvalue, the easier it is to partition a network
- The higher the Fiedler eigenvalue, the larger is the network connectivity - it is harder to separate into clusters

Lecture 11

68 Describe the variant Kernighan-Lin algorithm for community detection. What is its complexity? How does it compare to the original Kernighan-Lin algorithm for graph partitioning?

In case of community detection, it is not possible to use the size of the cut set as a criterion as it would be always possible to set the size of one of the clusters to 0, rendering the cut set size also 0.
Instead, we consider how many edges between vertices belonging to same communities as compared to randomly expected number of connections. Such property is measured with modularity

$$Q = \frac{1}{2m}\sum_{i,j}\left(A_{ij} - \frac{d_i d_j}{2m}\delta(c_i, c_j)\right) = \frac{1}{2m}\sum_{i,j} B_{ij}\delta(c_i, c_j). \tag{68.1}$$

This variant of the Kernighan-Lin is also a heuristic algorithm. Divides the network into 2 clusters of vertices. Calculate the change of modularity when moving a vertex to the other group. In each iteration choose the vertex which increases the modularity the most or decreases the least. A vertex that has been moved cannot be moved in the same iteration. Then we choose the best resulting partition from which we start another iteration.

In each round $O(n)$ modularity changes are considered, for every vertex $O(m/n)$ possible connections to other vertices are evaluated. That is $O(m)$ per every iteration of the algorithm. The amount of iterations is proportional to $O(n)$ so in total the overall complexity is $O(nm)$. The original varian of the algorithm has complexity $O(n^2m)$.

69 Describe the spectral modularity maximization method of community detection. Explain the importance of the modularity matrix and its leading eigenvector.

Warning

Proof/Derivation is required as a part of the exam

The method is based on the modularity matrix

$$Q = \frac{1}{2m}\sum_{i,j} A_{ij} - \frac{d_i d_j}{2m}\delta(c_i, c_j) = \frac{1}{2m}\sum_{i,j} B_{i,j}\delta(c_i, c_j), \tag{69.1}$$

where c_i are the community labels.

By definition

$$\sum_j B_{ij} = \sum_j A_{ij} - \frac{d_i}{2m}\sum_j d_j = 0, \tag{69.2}$$

$$\sum_j B_{ij} = \sum_j A_{ij} - \frac{d_j}{2m}\sum_i d_i = d_j^{out} - d_j^{in}, \tag{69.3}$$

which is also equal to zero on undirected and balanced graphs.

We define a vector $s \in \mathbb{R}^n$ such that

$$s_i = \begin{cases} +1 \text{ if } i \in \mathcal{V}_1, \\ -1 \text{ if } i \in \mathcal{V}_2. \end{cases} \tag{69.4}$$

With the help of s_i vertex labels one can construct the expression

$$\frac{1}{2}(1 - s_i s_j) = \begin{cases} 1 \text{ if } (i,j) \text{ vertices are in different groups} \\ 0 \text{ if } (i,j) \text{ vertices are in the same group.} \end{cases} \tag{69.5}$$

Using this expression, we rewrite the original expression for modularity

$$Q = \frac{1}{4m}\sum_{i,j} B_{ij}(s_i s_j + 1) = \frac{1}{4m}\left(\sum_{i,j} B_{ij} s_i s_j + \sum_{i,j} B_{ij}\right) \tag{69.6}$$

$$Q = \frac{1}{4m}\sum_{i,j} B_{ij} s_i s_j = \frac{1}{4m} s^T B s. \tag{69.7}$$

The maximization is then a restricted quadratic optimization problem, which is difficult to solve, but we can relax the problem to a constrained optimization problem by rather than considering $s_i = \pm 1$ being the vertices of a hyper-cube, we allow them to attain values on a hyper-sphere. The diameter of the hyper-sphere is given by the constraint $s^T s = \sum_i s_i^2 = n$ giving us the diameter $\sqrt(n)$.

The relaxed problem is a constrained unrestricted optimization problem, which can be optimized using the Lagrange method

$$\frac{\partial}{\partial s_i}\left[\sum_{j,k} B_{jk} s_j s_k + \beta\left(n - \sum_j s_j^2\right)\right] = 0, \ \forall i \tag{69.8}$$

$$\sum_j B_{ij} s_j - \beta s_i = 0, \tag{69.9}$$

which can be written in a compact version

$$Bs = \beta s. \tag{69.10}$$

This means that β is an eigenvalue of B and the modularity value is

$$Q = \frac{1}{4m} s^T B s = \frac{1}{4m}\beta s^T s = \frac{1}{4m}\beta n. \tag{69.11}$$

The modularity value is then maximized by the leading eigenvector u_1 of the modularity matrix. However, this is only the optimum of the relaxed problem. To find an optimum to the original problem, we have to choose $s_i = \pm 1$ nearest to the optimum of the relaxed problem. This can be done by maximizing their product

$$s^T u_1 = \sum_i s_i u_{1i}, \tag{69.12}$$

by choosing

$$s_i = \begin{cases} +1, \ u_{1i} > 0 \\ -1, \ u_{1i} < 0. \end{cases} \tag{69.13}$$

If an element of the leading eigenvector is equal to zero, either assignment is equally fine.

69.1 Spectral modularity maximization algorithm

- Find the leading eigenvector of B - $u_1 \in \mathbb{R}^n$
- Assign vertices to the two communities based on the sign of the elements of u_1

Unfortunately, the modularity matrix B is never sparse as opposed to the Laplacian, finding the leading eigenvector is therefore $O(n^3)$.

70 How does repeated bisection work for modularity maximization? Compare it with the same approach in graph partitioning.

Repeated bisection allows for division into more than two communities. The algorithm first separates the network into two communities. Then it runs again on each of the communities. This approach is applicable in graph partitioning, but in the case of community detection, modularity maximization upon bisecting each community does not in general maximize the total modularity of the resulting network division. We must therefore consider the resulting change in the total modularity.

For a community C of n_c vertices, the change in the modularity of the entire network upon bisection of C equals

$$\Delta Q = \frac{1}{4m} s^T B^C s, \text{ where} \tag{70.1}$$

$$B_{ij}^C = B_{ij} - \delta_{ij} \sum_{k \in C} B_{ik}. \tag{70.2}$$

We can now use the spectral modularity maximization is applicable on modularity change matrix B^C. We continue bisection of communities as long as modularity increase is possible.

71 Briefly describe simulated annealing, genetic algorithm and greedy algorithm for modularity maximization.

71.1 Simulated annealing

A stochastic optimization approach used for maximization of the modularity. The method finds approximately maximal Q without getting stuck in a local maxima. The algorithm uses the following steps

- Change an assignment of a vertex
- Calculate the change in modularity ΔQ
- If $\Delta Q \geq 0$, accept the change
- If $\Delta Q < 0$, accept the change with probability $p = \exp\left(\frac{\Delta Q}{T}\right)$, where T is the "temperature"
- The "temperature" decays with time in:
  - Geometric manner - $T_{k+1} = \alpha T_k$, where $\alpha in [0.9, 0.999]$
  - Linear manner
  - Adaptive manner

71.2 Genetic algorithm

- Generate an initial seed with semi-random partitions
- Select which partitions continue into the next generation - modularity evaluation of partitions
- Crossover selected partitions
- Mutation - random variation keeps diversity

71.3 Greedy algorithm

Starts from individual nodes. Joins them together so that modularity is maximized.
In the next step it joins the communities from previous step.
Ends when the whole graph is a single community. Checks all intermediate states and finds the one with the largest modularity.
Only algorithm the sort of works on very large networks $O(n \log^2 n)$.

72 Describe the algorithm using betweenness centrality for community detection. How does the Radicci algorithm differ from it?

Betweenness is measure how many shortest path between edges go through a node. The vertices that connect communities tend to have higher betweenness centrality.

In each step remove the edge with a largest betweenness centrality. Recalculate betweenness centrality. Repeat the steps until there are no edges remaining - choose which step had the highest modularity.

Complexity $O(mn^2 + nm^2)$. On a sparse graph $O(n^3)$

Radicci modification finds if an edge is a part of a short loop (4 edges).It has been empirically found, that edges connecting communities are often not parts of short loops. Works well if the graph has a lot of short loops (social networks). Technical networks usually avoid short loops.

Ends only after avery single vertex is separated. It is up to us to choose which division has maximal Modularity.

73 Explain how the agglomerative algorithms proceed in community detection.

- Agglomerative algorithm starts with individual vertices and joins them into ever greater communities.
- An divisive method does the opposite.
- The algorithm requires a definition of a measure of similarity of groups of vertices
  - First define similarity for pairs of vertices
  - Then generalize this measure for groups - maximal between any, minimal between any, average, etc.

74 Define hierarchical clustering. What is a dendrogram? Explain the similarity-based hierarchical clustering with single-, complete- and average-linkage clustering.

Dendogram - tree like structure which shows how the network is separated. In the root, the whole graph is a single cluster. In each level, the cluster is separated into 2. The lower in the Dendogram, the more separated the network is. It is then up to us to choose which division has maximal Modularity.

Betweenness centrality and Radicci methods started with the whole graph and gradually separated the network into single vertices. Similarity based clustering starts with single vertices and connects them with most similar. Then it does the same for the created communities (tuples of vertices) etc. The algorithm ends with the whole network in a single community. Again, we then evaluate at which level of the Dendogram was the modularity maximal.

For similarity clustering we need to define similarity of 2 vertices - (for example hierarchical equivalence, structural equivalence) and then we need to generalize how to calculate similarity for communities:

- minimal similarity of 2 vertices each in one cluster - $O(1)$
- maximum similarity of 2 vertices each in one cluster - $O(1)$
- average similarity - relatively easy to recalculate - $O(1)$
- etc.

The $O(1)$ complexity can be achieved when storing previous similarities in a binary heap. Calculating the average of larger communities does not need to sum every single vertex, we only need to average the similarities of the previous connection - $O(1)$

$$\bar{a}_1 = \frac{1}{n_1}\sum a_i \tag{74.1}$$

$$\bar{a}_2 = \frac{1}{n_2}\sum a_i \tag{74.2}$$

$$\bar{a}_{1,2} = \frac{1}{n_1+n_2}\left(n_1\bar{a}_1 + n_2\bar{a}_2\right). \tag{74.3}$$

With the binary heap addition and removal take $O(n\log n)$. The total complexity is $O(n^2\log n)$.

The result is usually tightly knit core and separate peripherals.

75 Comment on which of the algorithms detect a fixed number vs. an unspecified number of communities.

75.1 Specified

- Kernihan-lin algorithm for modularity maximization
- Spectral Modularity maximization

75.2 Unspecified

- Repeated bisection
- Simulated annealing
- Genetic algorithm
- Greedy algorithm
- Hierarchical clustering

Lecture 12

76 Explain the difference between site and bond percolation.

Percolations are stochastic processes. We distinguish between site and bond percolations which correspond to random removal of vertices or edges from a graph.

- Site percolation - vertices together with all their pertaining edges, are removed from the graph randomly with probability p
- Bond percolation - edges are removed from the network randomly with probability p

We remove vertices or edges from the graph with increasing probability and observe how is the network affected. The expected amount of removed vertices/edges is np/mp.

- Phase transition - network transitions abruptly from being connected to being a set of disconnected components. The transition happens for some probability p

Percolations are used for modelling the spread of forest fires, spread of epidemics in a network.

77 Describe uniform and non-uniform vertex removal. Comment on the robustness of power law networks to vertex removal.

A modification of the usual model removes the vertices non-uniformly, for example by their degrees - p_d is a probability that a vertex of degree d is removed.

Power law configuration model is found to be remarkably resilient to random uniform removal of vertices but far more susceptible to specific targeted attacks. Removal of a small fraction of highest-degree vertices can be extremely detrimental - attacking a hub.

78 Describe a few examples of fully mixed epidemics models, e.g. SI, SIR, SIS, SEIR.

Used for purposes of mathematical description of epidemics.

78.1 SI

- Divides the population into two groups susceptible of size S and infected of size X. The fraction of the total population is denoted by s and x.
- The size of these groups cannot be negative
- No entry or departure of individuals from the population is assumed
- The system is compartmental - $x \in \mathbb{R}_+^n$, $\dot{x} = f(x)$, $x_i = 0 \Rightarrow f_i(x_i) \geq 0$ - (in this context x is a general vector, not the fraction of infected)

The system is modelled as

$$\dot{s} = -\beta s x \tag{78.1}$$
$$\dot{x} = \beta s x. \tag{78.2}$$

The coefficient β represents the probability of a single infected infecting a susceptible individual. The probability of an infected person meeting a susceptible person is given by the sizes of each of the groups. With constraint $s + x = 1$ the dynamics are reduced to a one dimensional system

$$\dot{x} = \beta(1 - x)x, \tag{78.3}$$

which has a closed form solution

$$x(t) = \frac{x(0)\exp(\beta t)}{1 - x(0) + x(0)\exp(\beta t)}. \tag{78.4}$$

78.2 SIR

This model adds a group of recovered subpopulation of size R. The model is

$$\dot{s} = -\beta s x \tag{78.5}$$
$$\dot{x} = \beta s x - \gamma x \tag{78.6}$$
$$\dot{r} = \gamma x. \tag{78.7}$$

The model does not have a closed form solution.

78.3 SIS

The SIS model allows for recovered individuals to be infected again and so it is modelled as

$$\dot{s} = \gamma x - \beta s x \tag{78.8}$$
$$\dot{x} = \beta s x - \gamma x. \tag{78.9}$$

The constraint $s + x = 1$ implies

$$\dot{x} = \beta(1 - x)x - \gamma x \tag{78.10}$$
$$\dot{x} = (\beta - \gamma)x - \beta x^2 \tag{78.11}$$

which has a closed form solution

$$x(t) = \left(1 - \frac{\gamma}{\beta}\right) \frac{c\exp[(\beta - \gamma)t]}{1 + c\exp[(\beta - \gamma)t]}, \text{ where} \tag{78.12}$$
$$c = \frac{\beta x_0}{\beta - \gamma - \beta x_0} \tag{78.13}$$

The system has a globally asymptotically stable equilibrium $x = 0$ for $\beta - \gamma < 0$.

79 Explain how to model epidemics on networks. What is the role of vertices and vertex variables?

Network epidemics models divide the whole population into subgroups having different contact patterns to give a detailed predictions including the effects of various management strategies that treat various segments of the population differently.

- Each graph node describes a subpopulation $s_i, x_i, r_i, \ldots$ - for example cities, regions or different groups in the population - seniors, children
- The spread across the network depends on the connectivity of the graph
- In networked epidemics models we use degree approximation - all nodes of the same degree are assumed to behave similarly regardless of their precise position in the network

79.1 nSI

Diagonal adjacency matrix elements $A_{ii} = 1$

$$\dot{s}_i = -\beta s_i \sum_k A_{ij} x \tag{79.1}$$

$$\dot{x}_i = \beta s_i \sum_j A_{ij} x_j \tag{79.2}$$

79.2 A stochastic variant of the networked model

This variant uses correlations instead of products

$$\frac{\mathrm{d}}{\mathrm{d}t} < s_i > = -\beta \sum_j A_{ij} < s_i x_j > \tag{79.3}$$

80 Describe the Lotka-Volterra predator-prey model. Explain various ways how it extends to the network setting.

A population dynamics model is given in a form

$$\dot{x}_i = x_i \alpha_i (x_i, x_1, \ldots, x_n), \tag{80.1}$$

where α_i is the growth rate for species i, depending generally on all other species' populations in the ecosystem. The whole system is a non-negative compartmental system.

The Lotka-Volterra model is defined as

$$\dot{x} = x(\alpha - \beta y) \tag{80.2}$$

$$\dot{y} = y(-\gamma + \delta x), \tag{80.3}$$

where x describes the prey population, y describes the predator population and $\alpha, \beta, \gamma, \delta > 0$. There exists a preserved quantity

$$V = -\delta x + y \ln x - \beta y + \alpha \ln y. \tag{80.4}$$

There is a single equilibrium state surrounded by a continuum of periodic orbits.

80.1 Network variants

80.1.1 Multiple species

$$\dot{x}_i = x_i \left(\alpha_i + \sum_j A_{ij} x_j \right), \tag{80.5}$$

where A_ij has signed edges

$$A_{ij} \begin{cases} > 0, \text{ if species } i \text{ preys on species } j \\ < 0, \text{ if species } j \text{ preys on species } i \end{cases} . \tag{80.6}$$

Graph edges model predator-prey relations within one ecosystem.

80.1.2 Patchy environment

This variant models interaction of two species in multiple locations

$$\dot{x}_i = x_i(\alpha - \beta y_i) + \sum_j A_{ij} x_j \tag{80.7}$$

$$\dot{y}_i = y_i(-\gamma + \beta x_i) + \sum_j A_{ij} y_j, \tag{80.8}$$

where x_i describes the size of prey population in location i. Graph edges model the migration from neighboring areas in proportion to populations in the i^{th} location.

Lecture 13

Warning

Questions 81-87 are not part of the exam

81 (Enumerate the stages of the traditional (offline) web search. Explain how the algorithms used in modern web search engines such as Google differ from the traditional web search.)

82 (Explain the web crawling procedure. Give examples of advanced techniques that facilitate the web crawling process.)

83 (Describe the process of indexing in web search, both the simple version and its possible extensions.)

84 (Describe simple search in distributed databases. What is its complexity if the file is present only on a single computer in the network? How does the complexity change if the file exists on a fixed fraction of computers in the network?)

85 (Explain how supernodes improve the search in distributed networks.)

86 (Describe the two models of message passing, i.e., the Kleinberg's model and the hierarchical model, and enumerate their limitations (assumptions). What is the complexity of message passing in these models?)

87 (Elaborate on the constraints that have to be imposed on networks as the main conclusion of message passing analysis using models.)

Lecture 14

88 Define the consensus/synchronization problem in states and outputs. Explain the difference between homogeneous and heterogeneous agents.

Definition 88.1: Consensus/synchronization problem

The goal is to reach a state or output consensus

$$||x_i - x_j|| \to 0, \text{ for } t \to \infty, \text{ or} \tag{88.1}$$
$$||y_i - y_j|| \to 0, \text{ for } t \to \infty. \tag{88.2}$$

In case of leader following the problem is defined as

$$||x_i - x_0|| \to 0, \text{ for } t \to \infty, \text{ or} \tag{88.3}$$
$$||y_i - y_0|| \to 0, \text{ for } t \to \infty, \tag{88.4}$$

where the leader is uncontrolled

$$\dot{x}_0 = Ax_0 \tag{88.5}$$
$$y_0 = Cx_0 \tag{88.6}$$

Definition 88.2: Consensus

Consensus is a state which is constant. After reaching the consensus, the agents remain it such state indefinitely.

Definition 88.3: Synchronization

If agents are synchronized, they have reached a common state, which evolves in time. After achieving synchronization, the agents' states still evolve, but all in the same manner.

Definition 88.4: Agents

Agents are autonomous subsystems, with dynamics having a state description generally in $\mathbb{R}^n$ - $x_i \in \mathbb{R}^{n_i}$, $i = 1, \ldots, N$. Agents are assumed to have local computation and communication capabilities.

88.1 Homogeneous agents

Each of the agents can be described by a general LTI system

$$\dot{x}_i = Ax_i + Bu_i \tag{88.7}$$
$$y_i = Cx_i + Du_i \tag{88.8}$$

In general the systems may not be LTI

$$\dot{x}_i = f(x_i, u_i) \tag{88.9}$$
$$y_i = h(x_i, u_i). \tag{88.10}$$

The control problem for an multi-agent system is asymptotic state synchronization

$$||x_i - x_j|| \to 0,\ \forall (i,j) \text{ as } t \to \infty \tag{88.11}$$

88.2 Heterogeneous agents

Heterogeneous agents assume that systems of individual agents may differ from each other, their states may differ in dimensions $x_i \in \mathbb{R}^{n_i}$, $n_i \neq n_j$ and therefore the state synchronization problem cannot be posed. Even if the dimensions were the same, the state variables may be different and it would make no sense to synchronize such different state variables.

In the case of LTI systems

$$\dot{x}_i = A_i x_i + B_i u_i \tag{88.12}$$
$$y_i = C_i x_i + D_i u_i. \tag{88.13}$$

In general, the state-state equations are

$$\dot{x}_i = f_i(x_i, u_i) \tag{88.14}$$
$$y_i = h_i(x_i, u_i). \tag{88.15}$$

Instead, the control goal is asymptotic synchronization of outputs.

$$||y_i - y_j|| \to 0,\ \forall (i,j) \text{ as } t \to \infty \tag{88.16}$$

89 Write the single-integrator leaderless consensus dynamics in continuous time. How to include a leader?

The single-integrator agents have a dynamic model

$$\dot{x}_i = u_i, \tag{89.1}$$

with local neighborhood error in the states for control,

$$u_i = \sum_j e_{ij}(x_j - x_i), \tag{89.2}$$

where e_{ij} are the elements of the adjacency matrix E. The standard notation conflicts with state-space matrix A. The elements can either be binary to signify which agents are connected to which or they can be weights.

The control input is available to each single-agent in a distributed way. This results in a closed-loop system

$$\dot{x}_i = \sum_j e_{ij}(x_j - x_i) \tag{89.3}$$
$$\tag{89.4}$$

For every edge from/to agent x_i (there are d_i in total) the control input subtracts its value and for every neighbouring state its value is added. Which can be written as

$$u_i = -d_i x_i + \sum_j e_{ij} x_j. \tag{89.5}$$

This can be written in matrix notation using the Laplacian

$$\dot{x} = -(D - E)x, \quad \text{where } x \text{ is the multi-agent state vector} \tag{89.6}$$
$$\dot{x} = -Lx \tag{89.7}$$
$$x = \begin{bmatrix} x_1 & x_2 & \dots & x_N \end{bmatrix}^T \tag{89.8}$$

89.1 Leader - Pinning control

A leader is an uncontrolled agent

$$\dot{x}_0 = Ax_0 \tag{89.9}$$
$$y_0 = Cx_0. \tag{89.10}$$

The control of other agents is augmented with a non-negative pinning gain

$$g_i(x_0 - x_i), \tag{89.11}$$

where g_i can be either binary element signifying whether the i^{th} agent is connected to the leader or it can be a control gain same as with e_{ij}. Usually g_i is nonzero only for a small fraction of nodes. The following of the leader is then propagated through the graph structure to other vertices.

The total control input is then

$$\dot{x}_i = Ax_i + BK\left[\sum_j e_{ij}(x_j - x_i) + g_i(x_0 - x_i)\right], \tag{89.12}$$

which can again be written in matrix notation

$$\dot{x} = -(L+G)(x - \mathbf{1}_N x_0). \tag{89.13}$$

We can define a synchronization error

$$\delta_i = x_i - x_0 \tag{89.14}$$
$$\dot{\delta} = -(L+G)\delta. \tag{89.15}$$

The matrix $(L+G)$ is nonsingular and all eigenvalues have positive real part, therefore with pinning control, the synchronization will be achieved asymptotically.

90 Write the single-integrator leaderless consensus dynamics in discrete time. How to include a leader?

In discrete time the model of a single agent is

$$x_i(k+1) = u_i(k) \tag{90.1}$$

or in an alternative way

$$x_i(k+1) = x_i(k) + \tilde{u}_i(k). \tag{90.2}$$

In the first case, an uncontrolled system would immediately go to zero. In the latter, the uncontrolled system will retain its value.

The consensus is reached by calculating the weighted average of the current state and the state of all neighbors.

$$u_i(k) = \frac{1}{d_i+1}\left[x_i(k) + \sum_j e_{ij}x_j(k)\right], \tag{90.3}$$

where e_{ij} are elements of the adjacency matrix E. In compact notation the closed loop system is

$$x(k+1) = (I+D)^{-1}(I+E)x(k), \tag{90.4}$$

where $D = diag(d_i)$. If the graph has a spanning tree - if it is strongly connected, it will eventually converge to consensus.

Perhaps a clearer way to write the equation using the second definition of the input

$$x_i(k+1) = x_i(k) + \tilde{u}_i(k) \tag{90.5}$$
$$\tilde{u}_i(k) = -\frac{1}{1+d_i}\sum_j e_{ij}\left[x_i(k) - x_j(k)\right]. \tag{90.6}$$

In matrix notation it can be rewritten using the Laplacian

$$x(k+1) = x(k) + \tilde{u}(k) \tag{90.7}$$
$$\tilde{u}(k) = -(I+D)^{-1}Lx(k). \tag{90.8}$$

90.1 Leader

We add a leader as a term of the weighted average

$$x_i(k+1) = \frac{1}{1+d_i+g_i}\left[x_i(k) + g_i x_0 + \sum_j e_{ij}x_j(k)\right]. \tag{90.9}$$

In matrix notation that would be

$$x(k+1) = (I+D+G)^{-1}\left[(I+E)x(k) + G\mathbf{1}x_0\right]. \tag{90.10}$$

In discrete time, we can also define synchronization error

$$\delta_i(k) = x_i(k) - x(0)\delta(k+1) \qquad = (I+D+G)^{1}(I+E)\delta(k). \tag{90.11}$$

The system is asymptotically stable if the graph of followers is pinned to the root of a spanning tree or more generally to all roots of a spanning forest.

91 Set up the dynamical equations for continuous-time homogeneous LTI agents using local neighborhood error signal for state synchronization.

A single agent has dynamics

$$\dot{x}_i = Ax_i + Bu_i \tag{91.1}$$
$$y_i = Cx_i. \tag{91.2}$$

The agents are homogeneous and therefore the matrices A, B, C, D are the same for every agent. The goal is either $\|x_i - x_j\| \to 0$ in case there is no leader present, or if there is a leader present then the goal is $||x_i - x_0|| \to 0$.

We assume that the leader is uncontrolled

$$\dot{x}_0 = Ax_0 \tag{91.3}$$
$$y_0 = Cx_0. \tag{91.4}$$

For leaderless configuration the control input is

$$u_i = K\left[\sum_j e_{ij}(x_j - x_i)\right] \tag{91.5}$$

$$\dot{x}_i = Ax_i + BK\left[\sum_j e_{ij}(x_j - x_i)\right], \tag{91.6}$$

where K is a control gain. Unlike in single-integrator systems, here $x_i \in \mathbb{R}^n$, where n is the order of a single agent.

With leader following there is an addition of a pinning control

$$u_i = K\left[\sum_j e_{ij}(x_j - x_i) + g_i(x_0 - x_i)\right] \tag{91.7}$$

$$\dot{x}_i = Ax_i + BK\left[\sum_j e_{ij}(x_j - x_i) + g_i(x_0 - x_i)\right], \tag{91.8}$$

In vector notation that is

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \in \mathbb{R}^{nN},\ x_i \in R^n \tag{91.9}$$

$$\boldsymbol{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_N \end{bmatrix} \in \mathbb{R}^{mN},\ u_i \in R^m. \tag{91.10}$$

With the use of Kronecker multiplication

$$I_n \in \mathbb{R}^{n\times n},\ I_p \in \mathbb{R}^{p\times p},\ A \in \mathbb{R}^{p\times p},\ B \in \mathbb{R}^{n\times n} \tag{91.11}$$

$$I \otimes A = \begin{bmatrix} A & & \\ & \ddots & \\ & & A \end{bmatrix} \in \mathbb{R}^{np\times np} \tag{91.12}$$

$$A \otimes I = \begin{bmatrix} a_{11}I & \dots & a_{1p}I \\ \vdots & \ddots & \vdots \\ a_{p1}I & \dots & a_{pp}I \end{bmatrix} \in \mathbb{R}^{pn\times pn} \tag{91.13}$$

$$(I_n \otimes A)(B \otimes I_p) = B \otimes A \tag{91.14}$$

The local neighborhood error can be written as one of the two following options

$$L \otimes I\boldsymbol{x} \tag{91.15}$$

$$(L + G) \otimes I\boldsymbol{x}. \tag{91.16}$$

We can then write the control input u as

$$\boldsymbol{u} = \begin{cases} -(I \otimes K)(L \otimes I)\boldsymbol{x} = -L \otimes K\boldsymbol{x} \text{ for swarming} \\ -(I \otimes K)((L+G) \otimes I)\boldsymbol{x} = -(L+G) \otimes K(\boldsymbol{x} - \mathbf{1} \otimes x_0) \text{ for leader following} \end{cases} \tag{91.17}$$

$$\dot{\boldsymbol{x}} = \left[(I_N \otimes A) - cL \otimes BK\right]\boldsymbol{x} \tag{91.18}$$

$$\dot{\boldsymbol{\delta}} = \left[(I_N \otimes A) - c(L+G) \otimes BK\right]\boldsymbol{\delta}, \tag{91.19}$$

where $c > 0$ is a scalar gain. We design the gain K based on the dynamics of a single agent and the scalar gain c based on the graph topology. The total system dynamic matrix has size $\mathbb{R}^{mN\times nN}$.

92 Show how to use complex matrix pencils for investigating state synchronization of homogeneous agents.

Warning

Proof/Derivation is required as a part of the exam

We are looking for a transformation T

$$T^{-1}LT = \Lambda \tag{92.1}$$
$$T^{-1}(L+G)T = \Lambda, \tag{92.2}$$

where Λ is triangular. The analysis is the same for both cases, so I will continue with swarming.

If we apply such transformation on the system dynamic matrix we get

$$(T^{-1} \otimes I_n)(I_N \otimes A - cL \otimes BK)(T \otimes I_n). \tag{92.3}$$

The Kronecker multiplication properties give

$$T^{-1}T \otimes A - cT^{-1}LT \otimes BK \tag{92.4}$$
$$I_n \otimes A - c\Lambda \otimes BK \tag{92.5}$$

The resulting matrix has triangular blocks on the diagonal. The blocks are

$$A - c\Lambda_{ii}BK \in \mathbb{R}^{n \times n}. \tag{92.6}$$

Now we can analyze N matrices of lower dimensionality then before. The eigenvalues of the blocks are the eigenvalues of the whole matrix. If the eigenvalues of theses blocks are stable, we can conclude that the whole system is stable.

The diagonal elements of Λ_{ii} are the eigenvalues of the graph Laplacian. One is zero and the other ones have non-negative real part. If the graph has a spanning tree, then the other eigenvalues have strictly positive real part.

The matrices on the block diagonal are $A - c\lambda_i BK \in \mathbb{C}^{n \times n}$. We can now ignore the graph topology and only analyze the stability of the matrix pencil. A metric pencil is a matrix polynomial of the first degree. In our case a polynomial of variable $\sigma \in \mathbb{C}$ with matrix coefficients. The matrix pencil is

$$A - \sigma BK. \tag{92.7}$$

For which σ is the matrix asymptotically stable. These create the synchronizing region.

We want to find a feedback gain K which stabilizes a single agent - we want to maximize the synchronizing region of a single agent. Then we want to find a scalar gain $c > 0$ which scales the eigenvalues of the graph Laplacian into the synchronizing region.

We require (A, B) stabilizeable. Also that $(A - BK)$ asymptotically stable for $\sigma = 1$ - that is there exists a K such that $\sigma = 1$ is in the synchronizing region.

93 Show that with the distributed feedback gain designed from the single-agent Algebraic Riccati Equation the resulting synchronizing region is an unbounded left-hand half-plane in the complex plane.

Warning

Proof/Derivation is required as a part of the exam

93.1 Lyapunov analysis

For continuous time the Lyapunov function is

$$V(z) = z^\dagger P z,\ z \in \mathbb{C},\ P = P^T \succ 0,\ P \in \mathbb{R}^{n\times n}. \tag{93.1}$$

Here $z^\dagger$ stands for Hermitian adjoint - transposed and complex conjugate.

$$z = \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix},\ z^\dagger = \begin{bmatrix} \bar{z}_1 & \dots & \bar{z}_n \end{bmatrix},\ \text{where} \tag{93.2}$$

$$z_i = a + bi,\ \bar{z}_i = a - bi \tag{93.3}$$

$$z^\dagger z = \sum_i \bar{z}_i z_i = \sum_i |z_i|^2 = \|z\|^2. \tag{93.4}$$

The time derivative of the Lyapunov function is

$$\dot{V} = z^\dagger \left[(A - \sigma BK)^\dagger P + P(A - \sigma BK) \right] z. \tag{93.5}$$

The matrix can be modified to obtain

$$(A - \sigma BK)^\dagger P + P(A - \sigma BK) = A^T P + PA - (\sigma BK)^\dagger P - P(\sigma BK) \tag{93.6}$$

$$= A^T P + PA - K^T B^T \bar{\sigma}^T P - P\sigma BK. \tag{93.7}$$

We chose the feedback gain

$$K = R^{-1} B^T P \tag{93.8}$$

And substitute into (93.7)

$$(A - \sigma BK)^\dagger P + P(A - \sigma BK) = A^T P + PA - (R^{-1}B^T P)^T B^T \bar{\sigma}^T P - P\sigma B R^{-1} B^T P \tag{93.9}$$

$$= A^T P + PA - PBR^{-1}B^T \bar{\sigma}^T P - \sigma PBR^{-1}B^T P \tag{93.10}$$

$$= A^T P + PA - (\bar{\sigma}^T + \sigma) PBR^{-1}B^T P. \tag{93.11}$$

Since $\sigma \in \mathbb{C}$ is a scalar complex number, the transpose is equal to itself.

$$(A - \sigma BK)^\dagger P + P(A - \sigma BK) = A^T P + PA - (\bar{\sigma} + \sigma) PBR^{-1}B^T P \tag{93.12}$$

$$= A^T P + PA - 2\Re(\sigma) PBR^{-1}B^T P \tag{93.13}$$

$$= A^T P + PA - PBR^{-1}B^T P + (1 - 2\Re(\sigma)) PBR^{-1}B^T P. \tag{93.14}$$

Now we substitute from ARE

$$A^T P + PA + Q - PB^{-1}RB^T P = 0 \tag{93.15}$$
$$A^T P + PA - PB^{-1}RB^T P = -Q. \tag{93.16}$$

We obtain

$$(A - \sigma BK)^\dagger P + P(A - \sigma BK) = -Q + (1 - 2\Re(\sigma))PBR^{-1}B^T P. \tag{93.17}$$

The resulting matrix is negative definite for $\Re(\sigma) > \frac{1}{2}$ meaning the synchronizing region is a right half-plane.

Every positive eigenvalue of the graph Laplacian can be multiplied by c such that

$$c\,\Re(\lambda_i) \;>\; \Re(\sigma) \;>\; \frac{1}{2}. \tag{93.18}$$

This gives us that the scalar gain c needs to satisfy

$$c > \frac{1}{2\Re(\min \lambda_i(L)_{>0})}. \tag{93.19}$$

In case of leader following

$$c > \frac{1}{2\Re(\min \lambda_i(L+G))}. \tag{93.20}$$

The condition is only a sufficient condition, in fact the stabilizing region is a bit larger. Fig. 2 show an example of the synchronizing region.

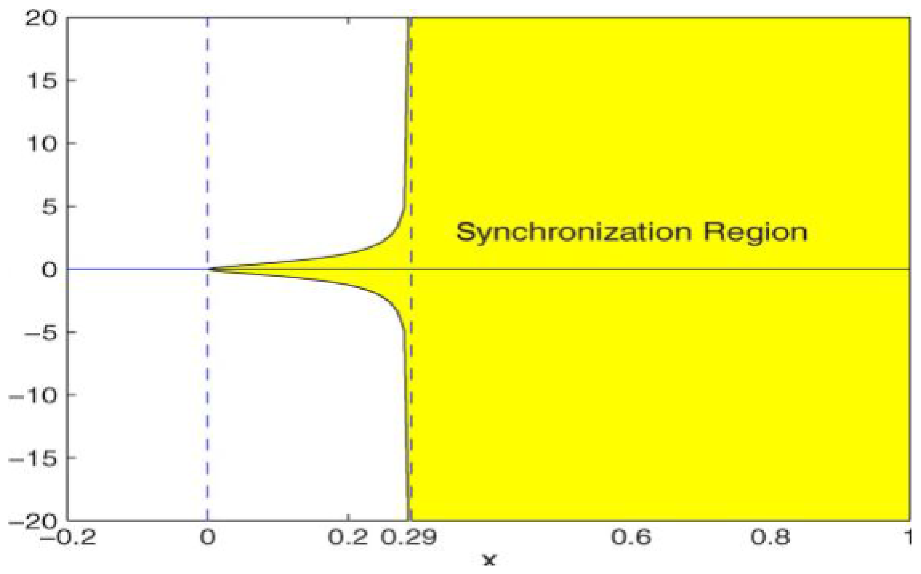

Figure 2: Synchronizing region

94 What are the necessary topological conditions on the communication graph for consensus or synchronization? Explain the dynamical role of the Fiedler eigenvalue in continuous time single integrator consensus.

The control input of a single integrator system for each vertex is $\dot{x} = -Lx$. The dynamics of the system is given by the graph Laplacian - if every eigenvalue (other then the 0) of the Laplacian has positive real part (the eigenvalues of -L need to be negative), the system will reach consensus. The zero eigenvalue corresponds to the consensus vector $L\mathbf{1} = 0$. If more than one eigenvalues are zero, the graph does not have a spanning tree and therefore there are at least two connected components which are not connected together. Each of the components will reach its own consensus but there will not be a common consensus. If the graph is weakly connected, then each of the strongly connected components will reach its own consensus and then slowly all vertices will reach a common consensus.

The rate of convergence is given by the second smallest eigenvalue - the Fiedler eigenvalue.