graph partitioning
community detection

SYSM 6302

CLASS 15

Divide network into groups/clusters

Heuristic Algorithms

→ **Graph Partitioning**
- fixed number/size of groups
- goal is to divide the graph into more managable pieces, ie., the graph will be divided
- reduce the number of edges between groups.

→ **Community detection**
- goal is to analyze a network to determine groups that may exist
- reduce the number of edges compared to number expected
- size & number of groups not fixed

Partitioning

in fact the division of groups is called a cut set

→ Fundamentally similar to finding a cut set

↳ But would need to consider not only all-pairs of cut sets, but somehow aggregate the vertex pair cutsets into a "network cut set"

→ Specify the number and approximate relative sizes of groups

↳ Evenly distribute a network calculation to minimize inter-group communication

Community Detection

→ Goal is not to minimize absolute number of edges (cut set)
  ↳ Relative to the number expected (modularity)

→ Many popular methods focus on Modularity maximization:

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

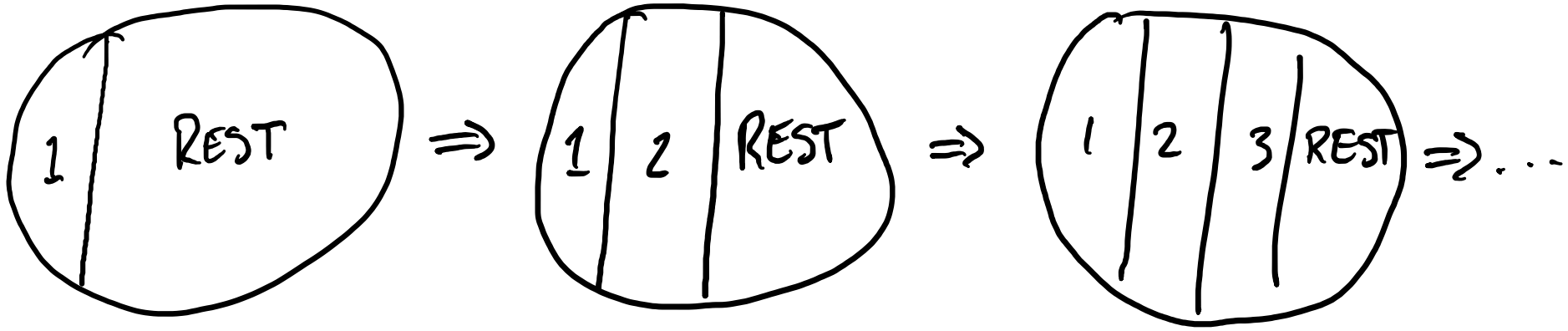
→ Aim to find the groups in the network - if there is only one group, then don't need to divide the network ← more subjective than partitioning

→ Any black box optimization method can be used to max Q
  → Genetic algorithm, simulated annealing, particle swarm, or greedy

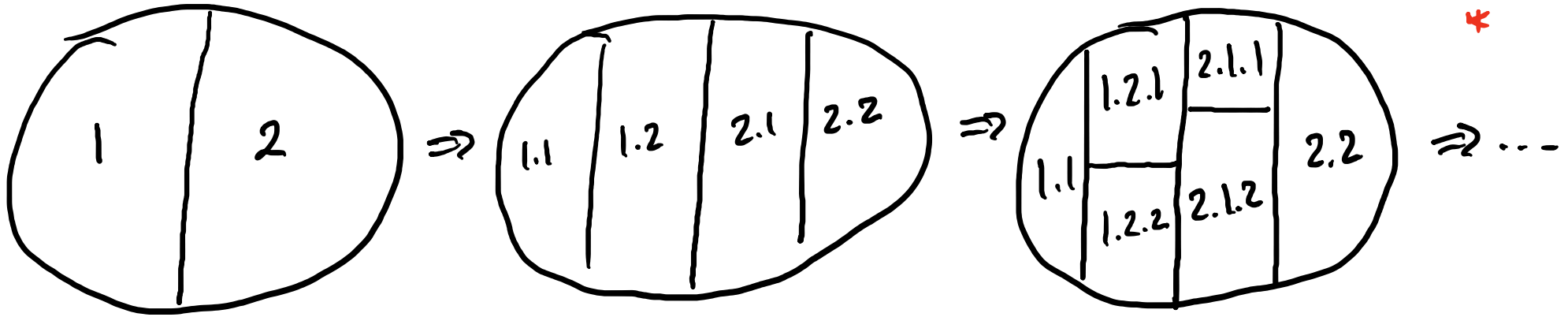
Bisection methods ← split something into 2 pieces (samuri style)

→ Most algorithms will perform bisection, but can be used to do further division

PARTITIONING:



COMMUNITY DETECTION:



*important to evaluate Q based on the entire graph not only the subgraph being split

Kernighan-Lin Algorithm for Partitioning

1. Randomly split network (nodes) into the groups (respecting number & size)
2. Find the pair $i \in$ Group1, $j \in$ Group2 that, if swapped, reduces the cut set size the most. Must test all pairs i, j. If no pair reduces the cut set size, pick the pair that increases it the least. ← $O(n^2)$
3. Make this swap.
4. Continue this (without picking the same nodes again) until you run out of nodes that haven't been picked before. ← $O(\frac{n}{2})$
5. Look back over all these swaps & pick the partitioning that has the smallest cut
6. Start with this partition and begin again at (2). Do this "several" times.

Simple Modularity Maximization for Community Detection

1. Split the network into two groups
2. Determine the modularity improvement for moving each node to the other group.
3. Make the move for the node that most increases (or least decreases) modularity
4. After each node has been moved once, pick the grouping with max modularity
5. Use these groups and start again at 2. Do this "several" times.

Spectral Modularity Maximization

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

Let $s \in \mathbb{R}^n$ such that $s_i = \begin{cases} +1 & \text{if node } i \text{ belongs to group 1} \\ -1 & \text{if node } i \text{ belongs to group 2} \end{cases}$

$$\delta(c_i, c_j) = \frac{1}{2}(s_i s_j + 1)$$

in lab show: $\sum_j B_{ij} = 0$

$$Q = \frac{1}{4m} \sum_{i,j} B_{ij}(s_i s_j + 1) = \frac{1}{4m} \sum_{i,j} B_{ij} s_i s_j = \frac{1}{4m} \sum_{i=1}^{n} s_i \sum_{j=1}^{n} B_{ij} s_j$$

$$= \frac{1}{4m} s^T B s$$

$$\max_{s_i \in \{-1,1\}} \frac{1}{4m} s^T B s \approx \max_{s_i \in \mathbb{R}} \frac{1}{4m} s^T B s$$

$$s^T s = n$$

Called a relaxation of the integer optimization problem, leading to a new constraint

$$s_1^2 + s_2^2 + \cdots + s_n^2 = n$$

using the convexity of the quadratic form

$$\frac{\partial}{\partial s}\left(s^T B s\right) = 0 \Rightarrow \frac{\partial}{\partial s}\left[s^T B s + \beta(n - s^T s)\right] = 0$$

Lagrange multiplier method for adding constraints

see next slide

$$\Rightarrow 2Bs - 2\beta s = 0 \longrightarrow Bs = \beta s \leftarrow \text{Eigenvector!!}$$

To maximize $\frac{1}{4m} s^T B s = \frac{1}{4m} s^T(\beta s) = \frac{\beta}{4m} \overbrace{s^T s}^{=n}$ ← pick largest β, i.e., largest eigenvalue

Select $s = v_1$ ← eigenvector corresponding to largest eigenvalue

EXCEPT $s_i \in \{\pm 1\}$

$$\Rightarrow s_i = \begin{cases} +1 & [v_1]_i > 0 \\ -1 & [v_1]_i < 0 \end{cases} \Rightarrow s = \text{sign}(v_1)$$

Consider $a = x^T A x = \sum_{i=1}^{n} x_i \sum_{j=1}^{n} A_{ij} x_j = \sum_{i,j} x_i A_{ij} x_j$ notice $a \in \mathbb{R}$ (scalar)

$$\frac{\partial a}{\partial x_k} = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial}{\partial x_k} (x_i A_{ij} x_j) = \sum_{j=1}^{n} 1 A_{kj} x_j + \sum_{i=1}^{n} x_i A_{ik} 1 = [Ax]_k + [A^T x]_k$$

$$\frac{\partial}{\partial x_k} (x_i A_{ij} x_j) = \begin{cases} A_{ij} x_j & \text{if } i = k \\ x_i A_{ij} & \text{if } j = k \\ 0 & \text{otherwise} \end{cases}$$

$$\Rightarrow \frac{\partial a}{\partial x} = \begin{bmatrix} [Ax]_1 + [A^T x]_1 \\ [Ax]_2 + [A^T x]_2 \\ \vdots \\ [Ax]_n + [A^T x]_n \end{bmatrix} = Ax + A^T x = (A + A^T) x$$

↖ if $A = A^T$ (symmetric)

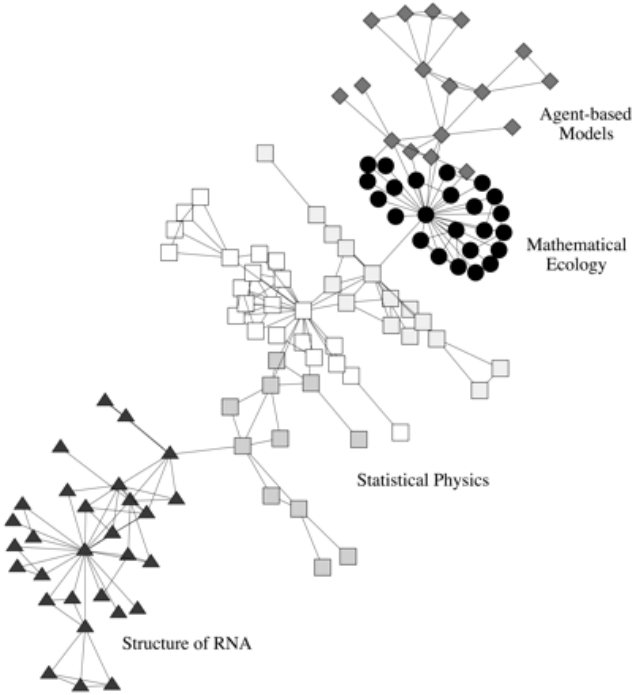
$$\frac{\partial a}{\partial x} = \frac{\partial}{\partial x} (x^T A x) = 2Ax$$

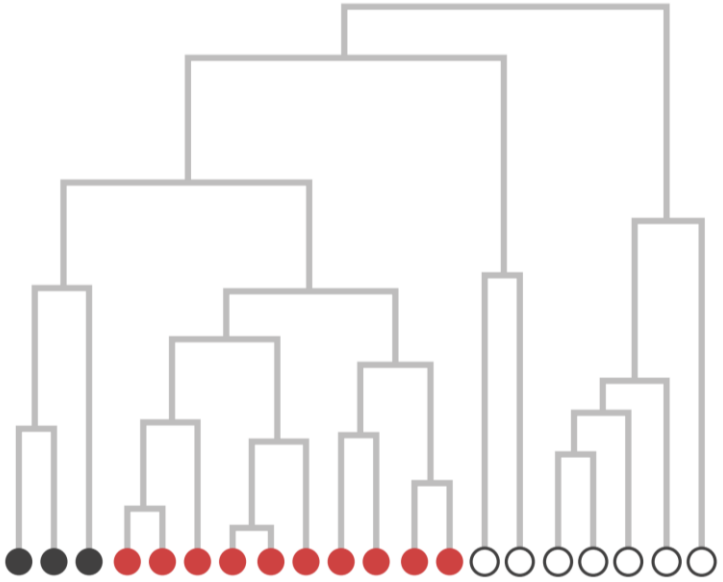
Betweenness-based Community Detection

→ Edges that lie between communities have high betw

① Remove the edge with highest betweenness

② Continue ① until there are no edges





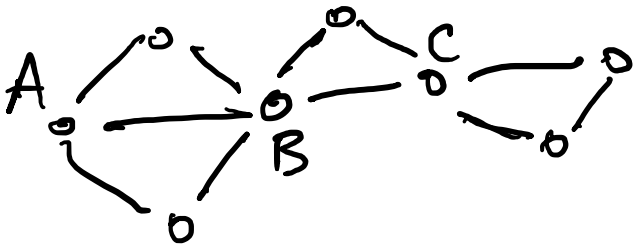
Yields a **dendrogram** which captures the process of decomposing nodes down to n individual components.

↳ Identify a meaningful cutoff

Louvain Community Detection

→ Modularity Maximization requires testing all possible groupings

↳ Beyond ~100 nodes this combinatorial explosion is ungainly



if A and C belong to the same community, how likely is it that B does too?

⇒ Adjacent nodes are more likely to belong to the same community

Louvain Community Detection

⓪ Assign every node to its own community

① Compute modularity of the network

② For each node: assign it the same community as each of its neighbors
if modularity increases keep the assignment

③ If modularity has increased since ①, go back to ①

④ Otherwise contract all nodes of the same type into a single meta node
with self loops representing edges between the community
weighted edges aggregating edges between communities

⑤ If no contraction occurs: stop. Otherwise go back to ①