Community Detection in Graphs
We often think of networks being organized into **modules, cluster, communities:**
Goal: Find Densely Linked Clusters
Non-overlapping Clusters

Network

Adjacency matrix

Nodes

Nodes
Find micro-markets by partitioning the query-to-advertiser graph:

[Andersen, Lang: Communities from seed sets, 2006]
Clusters in Movies-to-Actors graph:

[Andersen, Lang: Communities from seed sets, 2006]
Discovering social circles, circles of trust:

- friends under the same advisor
- CS department friends
- college friends
- 'ego' $u$
- 'alters' $v_i$
- family members
- highschool friends

[McAuley, Leskovec: Discovering social circles in ego networks, 2012]
Graph is large

- Assume the graph fits in main memory
  - For example, to work with a 200M node and 2B edge graph one needs approx. 16GB RAM
  - But the graph is too big for running anything more than linear time algorithms

- We will cover a PageRank based algorithm for finding dense clusters
  - The runtime of the algorithm will be proportional to the cluster size (not the graph size!)
Discovering clusters based on seed nodes

- **Given**: Seed node \( s \)
- Compute (approximate) Personalized PageRank (PPR) around node \( s \) (teleport set=\( \{s\} \))
- Idea is that if \( s \) belongs to a nice cluster, the random walk will get **trapped** inside the cluster
Algorithm outline:

- Pick a seed node $s$ of interest
- Run PPR with teleport set = $\{s\}$
- Sort the nodes by the decreasing PPR score
- Sweep over the nodes and find good clusters
What makes a good cluster?

- **Undirected graph** $G(V, E)$:

- **Partitioning task:**
  - Divide vertices into 2 disjoint groups $A, B = V \setminus A$

- **Question:**
  - How can we define a “good” cluster in $G$?

![Diagram of a graph with vertices partitioned into two groups A and B]
What makes a good cluster?

- Maximize the number of within-cluster connections
- Minimize the number of between-cluster connections
Express cluster quality as a function of the “edge cut” of the cluster

**Cut:** Set of edges with only one node in the cluster:

\[
cut(A) = \sum_{i \in A, j \notin A} w_{ij}
\]

**Note:** This works for weighted and unweighted (set all \(w_{ij}=1\)) graphs

\[\text{cut}(A) = 2\]
Partition quality: **Cut score**
- Quality of a cluster is the weight of connections pointing outside the cluster

**Degenerate case:**

**Problem:**
- Only considers external cluster connections
- Does not consider internal cluster connectivity
Graph Partitioning Criteria

- **Criterion: Conductance:**
  Connectivity of the group to the rest of the network relative to the density of the group

\[
\phi(A) = \frac{| \{(i, j) \in E; i \in A, j \notin A\} |}{\min(\text{vol}(A), 2m - \text{vol}(A))}
\]

- **Vol(A):** total weight of the edges with at least one endpoint in A: \(\text{vol}(A) = \sum_{i \in A} d_i\)
  - Vol(A) = 2*#edges inside A + #edges pointing out of A

- **Why use this criterion?**
  - Produces more balanced partitions

\[\text{m} \ldots \text{number of edges of the graph}\]
\[\text{d}_i \ldots \text{degree of node } i\]
Example: Conductance Score

\[ \phi = \frac{2}{4} = 0.5 \]

\[ \phi = \frac{6}{92} = 0.065 \]
Algorithm Outline: Sweep

- **Algorithm outline:**
  - Pick a seed node $s$ of interest
  - Run PPR w/ teleport={$s$}
  - Sort the nodes by the decreasing PPR score
  - **Sweep** over the nodes and find good clusters

- **Sweep:**
  - Sort nodes in decreasing PPR score $r_1 > r_2 > \cdots > r_n$
  - For each $i$ compute $\phi(A_i = \{r_1, \ldots, r_i\})$
  - Local minima of $\phi(A_i)$ correspond to good clusters
The whole Sweep curve can be computed in **linear** time:

- For loop over the nodes
- Keep hash-table of nodes in $A_i$
- To compute $\phi(A_{i+1}) = \text{Cut}(A_{i+1})/\text{Vol}(A_{i+1})$
  - $\text{Vol}(A_{i+1}) = \text{Vol}(A_i) + d_{i+1}$
  - $\text{Cut}(A_{i+1}) = \text{Cut}(A_i) + d_{i+1} - 2\#(\text{edges of } u_{i+1} \text{ to } A_i)$
How to compute Personalized PageRank (PPR) without touching the whole graph?

- Power method won’t work since each single iteration accesses all nodes of the graph:
  \[ \mathbf{r}^{(t+1)} = \beta \mathbf{M} \cdot \mathbf{r}^{(t)} + (1 - \beta) \mathbf{a} \]
  - \( \mathbf{a} \) is a teleport vector: \( \mathbf{a} = [0 \cdots 0 \ 1 \ 0 \cdots 0]^T \)
  - \( \mathbf{r} \) is the personalized PageRank vector

Approximate PageRank [Andersen, Chung, Lang, ‘07]

- A fast method for computing approximate Personalized PageRank (PPR) with teleport set =\{s\}
- ApproxPageRank(s, \( \beta \), \( \varepsilon \))
  - s ... seed node
  - \( \beta \) ... teleportation parameter
  - \( \varepsilon \) ... approximation error parameter
Overview of the approximate PPR

- Lazy random walk, which is a variant of a random walk that stays put with probability $1/2$ at each time step, and walks to a random neighbor the other half of the time:

$$r_u^{(t+1)} = \frac{1}{2} r_u^{(t)} + \frac{1}{2} \sum_{i \rightarrow u} \frac{1}{d_i} r_i^{(t)}$$

- Keep track of residual PPR score $q_u = p_u - r_u^{(t)}$
  - Residual tells us how well PPR score of $u$ is approximated
  - $p_u$... is the “true” PageRank of node $u$
  - $r_u^{(t)}$... is PageRank estimate at around $t$

If residual $q_u$ of node $u$ is too big $\frac{q_u}{d_u} \geq \varepsilon$ then push the walk further (for each $v$ such that $(u, v) \in E$: $q_v' = q_v + \frac{1}{2} \beta \frac{q_u}{d_u}$), else don’t touch the node.
Towards approximate PPR

- **A different way to look at PageRank:**


  \[ p_\beta(\alpha) = (1 - \beta)\alpha + \beta p_\beta(M \cdot \alpha) \]

  - \( p_\beta(\alpha) \) is the true PageRank vector with teleport parameter \( \beta \), and teleport vector \( \alpha \)
  - \( p_\beta(M \cdot \alpha) \) is the PageRank vector with teleportation vector \( M \cdot \alpha \), and teleportation parameter \( \beta \)
  - where \( M \) is the stochastic PageRank transition matrix
  - Notice: \( M \cdot \alpha \) is one step of a random walk
Towards approximate PPR

- Proving: \( p_\beta(a) = (1 - \beta)a + \beta p_\beta(M \cdot a) \)
  - We can break this probability into two cases:
    - Walks of length 0, and
    - Walks of length longer than 0
  - The probability of length 0 walk is \( 1 - \beta \), and the walk ends where it started, with walker distribution \( a \).
  - The probability of length >0 is \( \beta \), and then the walk starts at distribution \( a \), takes a step, (so it has distribution \( Ma \)), then takes the rest of the random walk to with distribution \( p_\beta(Ma) \)
    - Note that we used the memoriless nature of the walk: After we know the second node of the walk has distribution \( Ma \), the rest of the walk can forget where it started and behave as if it started at \( Ma \). This proves the equation.
“Push” Operation

- **Idea:**
  - \( r \) ... approx. PageRank, \( q \) ... its residual PageRank
  - Start with trivial approximation: \( r = 0 \) and \( q = 0 \)
  - Iteratively push PageRank from \( q \) to \( r \) until \( q \) is small

- **Push: 1 step of a lazy random walk from node \( u \):**

  \[
  \text{Push}(u, r, q):
  \]
  \[
  r' = r, \quad q' = q
  \]
  \[
  r'_u = r_u + (1 - \beta)q_u
  \]
  \[
  q'_u = \frac{1}{2} \beta q_u
  \]
  for each \( v \) such that \( (u, v) \in E \):
  \[
  q'_v = q_v + \frac{1}{2} \beta \frac{q_u}{d_u}
  \]

Update \( r \)

Do 1 step of a walk:
Stay at \( u \) with prob. \( \frac{1}{2} \)
Spread remaining \( \frac{1}{2} \)
fraction of \( q_u \) as if a single step of random walk were applied to \( u \)

residual PPR score \( q_u = p_u - r_u \)
If $q_u$ is large, this means that we have underestimated the importance of node $u$.

Then we want to take some of that residual ($q_u$) and give it away, since we know that we have too much of it.

So, we keep $\frac{1}{2} \beta q_u$ and then give away the rest to our neighbors, so that we can get rid of it.

This corresponds to the spreading of $\frac{1}{2} \beta q_u/d_u$ term.

Each node wants to keep giving away this excess PageRank until we have settled close to the all nodes have no or a very small gap.

---

**Push($u$, $r$, $q$):**

- $r' = r$, $q' = q$
- $r'_u = r_u + (1 - \beta)q_u$
- $q'_u = \frac{1}{2} \beta q_u$

for each $v$ such that $(u, v) \in E$:

- $q'_v = q_v + \frac{1}{2} \beta \frac{q_u}{d_u}$

return $r'$, $q'$
Approximate PPR

- **ApproxPageRank(S, β, ε):**

  Set $r = \vec{0}$, $q = [0 \ldots 0 1 0 \ldots 0]$

  While $\max_{u \in V} \frac{q_u}{d_u} \geq \varepsilon$:
  
  Choose any vertex $u$ where $\frac{q_u}{d_u} \geq \varepsilon$

  **Push(u, r, q):**

  $r' = r$, $q' = q$

  $r'_u = r_u + (1 - \beta)q_u$

  $q'_u = \frac{1}{2} \beta q_u$

  For each $v$ such that $(u, v) \in E$:

  $q'_v = q_v + \frac{1}{2} \beta q_u / d_u$

  Update $r = r'$, $q = q'$

  Return $r$
Observations (1)

- **Runtime:**
  - PageRank-Nibble computes PPR in time $\left( \frac{1}{\varepsilon(1-\beta)} \right)$ with residual error $\leq \varepsilon$
    - Power method would take time $O\left( \frac{\log n}{\varepsilon(1-\beta)} \right)$

- **Graph cut approximation guarantee:**
  - If there exists a cut of conductance $\phi$ and volume $k$ then the method finds a cut of conductance $O\left( \sqrt{\phi / \log k} \right)$

The smaller the $\varepsilon$ the farther the random walk will spread!
Observations (3)

[Andersen, Lang: Communities from seed sets, 2006]
Algorithm summary:

- Pick a seed node $s$ of interest
- Run PPR with teleport set = $\{s\}$
- Sort the nodes by the decreasing PPR score
- Sweep over the nodes and find good clusters
Motif-Based Local Spectral Clustering
What if we want our clustering based on other patterns (not edges)?

Small subgraphs (motifs, graphlets) are building blocks of networks [Milo et al., ’02]
Motif-based spectral clustering

Network:

Motif:
Re-define Conductance for Motifs

- **Generalize cuts and volumes to motifs**

  $\text{edges cut}$ \quad $\rightarrow$ \quad $\text{motifs cut}$

  $\text{vol}(S) = \#(\text{edge end-points in } S)$ \quad $\rightarrow$ \quad $\text{vol}_M(S) = \#(\text{motif end-points in } S)$

  $\phi(S) = \frac{\#(\text{edges cut})}{\text{vol}(S)}$ \quad $\rightarrow$ \quad $\phi_M(S) = \frac{\#(\text{motifs cut})}{\text{vol}_M(S)}$

Optimize motif conductance

[Benson et al., ’16]
Motif-based Clustering

- **Three basic stages:**
  1) **Pre-processing**
     - $W_{ij}^{(M)} = \# \text{times } (i, j) \text{ participates in the motif}$
  2) **PageRank Nibble**
     - Same as before but on weighted $W^{(M)}$
  3) **Sweep**
     - Same as before

**Figure 1:**
- Higher-order network structures and the higher-order network clustering framework. A: Higher-order structures are captured by network motifs. For example, all 13 connected three-node directed motifs are shown here.
- B: Clustering of a network based on motif $M^7$. For a given motif $M$, our framework aims to find a set of nodes $S$ that minimizes motif conductance, $M(S)$, which we define as the ratio of the number of motifs cut (filled triangles cut) to the minimum number of nodes in instances of the motif in either $S$ or $\bar{S}$.
- In this case, there is one motif cut.
- C: The higher-order network clustering framework. Given a graph and a motif of interest (in this case, $M^7$), the framework forms a motif adjacency matrix $(W^{(M)})$ by counting the number of times two nodes co-occur in an instance of the motif. An eigenvector of a Laplacian transformation of the motif adjacency matrix is then computed. The ordering of the nodes provided by the components of the eigenvector produces nested sets $S_r = \{1, \ldots, r\}$ of increasing size $r$. We prove that the set $S_r$ with the smallest motif-based conductance, $M(S_r)$, is a near-optimal higher-order cluster (13).
Motif-based Clustering of a Food Web

Use multiple eigenvectors or recursive bi-partitioning to get multiple clusters
Motif Clustering of a Neural Network

Neuron locations

“Bi-fan” motif known to be important in neural networks [Milo et al., ’02]

- Ring motor (RME*) neurons act as inputs
- Inner labial sensory (IL2*) neurons are the destinations
- URA neurons act as intermediaries
Analysis of Large Graphs: Trawling
Search for small communities in a Web graph

What is the signature of a community/discussion in a Web graph?

Intuition: Many people all talking about the same things

Dense 2-layer graph

Use this to define “topics”: What the same people on the left talk about on the right
Remember HITS!
A more well-defined problem:
Enumerate complete bipartite subgraphs $K_{s,t}$
- Where $K_{s,t}$: $s$ nodes on the “left” where each links to the same $t$ other nodes on the “right”

$|X| = s = 3$
$|Y| = t = 4$

Fully connected
Market basket analysis:

- What items are bought together in a store?

Setting:

- Market: Universe \( U \) of \( n \) items
- Baskets: \( m \) subsets of \( U: S_1, S_2, \ldots, S_m \subseteq U \)
  \((S_i \text{ is a set of items one person bought})\)
- Support: Frequency threshold \( s \)

Goal:

- Find all sets \( T \) s.t. \( T \subseteq S_i \) of \( \geq s \) sets \( S_i \)
  \((\text{items in } T \text{ were bought together at least } s \text{ times})\)
The Apriori Algorithm

- For $i = 1, \ldots, k$
  - Generate all sets of size $i$ by composing sets of size $i - 1$ that differ in 1 element
  - Prune the sets of size $i$ with support $< s$

- What’s the connection between the itemsets and complete bipartite graphs?
Freq. Itemsets finds Complete bipartite graphs

Set frequency threshold $s=3$

Suppose $\{a, b, c\}$ is a frequent itemset

Then there exist $\geq s$ nodes that all link to each of $\{a, b, c\}$!
Freq. Itemsets finds Complete bipartite graphs

How?
- View each node \( i \) as a set \( S_i \) of nodes \( i \) points to
- \( K_{s,t} = \) a set \( Y \) of size \( t \) that occurs in \( s \) sets \( S_i \)
- Finding \( K_{s,t} \) is equivalent to finding itemsets of frequency threshold \( s \) and then look at layer \( t \)

\( S_i = \{ a, b, c, d \} \)

\( s \ldots \) minimum support (\( |X| = s \))
\( t \ldots \) frequent itemset size

[Kumar et al. ‘99]
View each node $i$ as a set $S_i$ of nodes $i$ points to

$S_i = \{a, b, c, d\}$

Find frequent itemsets:

- $s$ … minimum support
- $t$ … itemset size

We found $K_{s,t}$!

$K_{s,t} = \text{a set } Y \text{ of size } t \text{ that occurs in } s \text{ sets } S_i$
Example (1)

Support threshold $s=2$
- $\{b,d\}$: support 3
- $\{e,f\}$: support 2

And we just found 2 bipartite subgraphs:

Itemsets:
- $a = \{b,c,d\}$
- $b = \{d\}$
- $c = \{b,d,e,f\}$
- $d = \{e,f\}$
- $e = \{b,d\}$
- $f = \{}$
Example of a community from a web graph

A community of Australian fire brigades

<table>
<thead>
<tr>
<th>Nodes on the right</th>
<th>Nodes on the left</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSW Rural Fire Service Internet Site</td>
<td>New South Wales Fir...ial Australian Links</td>
</tr>
<tr>
<td>NSW Fire Brigades</td>
<td>Feuerwehrlinks Australien</td>
</tr>
<tr>
<td>Sutherland Rural Fire Service</td>
<td>FireNet Information Network</td>
</tr>
<tr>
<td>CFA: County Fire Authority</td>
<td>The Cherrybrook Rur...re Brigade Home Page</td>
</tr>
<tr>
<td>“The National Cente...ted Children’s Ho...</td>
<td>New South Wales Fir...ial Australian Links</td>
</tr>
<tr>
<td>CRAFTI Internet Connexions-INFO</td>
<td>Fire Departments, F... Information Network</td>
</tr>
<tr>
<td>Welcome to Blackwoo... Fire Safety Serv...</td>
<td>The Australian Firefighter Page</td>
</tr>
<tr>
<td>The World Famous Guestbook Server</td>
<td>Kristiansand brannv...dens brannvesener...</td>
</tr>
<tr>
<td>Wilberforce County Fire Brigade</td>
<td>Australian Fire Services Links</td>
</tr>
<tr>
<td>NEW SOUTH WALES FIR...ES 377 STATION</td>
<td>The 911 F,P,M., Fir...mp; Canada A Section</td>
</tr>
<tr>
<td>Woronora Bushfire Brigade</td>
<td>Feuerwehrlinks Australien</td>
</tr>
<tr>
<td>Mongarlowe Bush Fire – Home Page</td>
<td>Sanctuary Point Rural Fire Brigade</td>
</tr>
<tr>
<td>Golden Square Fire Brigade</td>
<td>Fire Trails “I...ghters around the...</td>
</tr>
<tr>
<td>FIREBREAK Home Page</td>
<td>FireSafe – Fire and Safety Directory</td>
</tr>
<tr>
<td>Guises Creek Volunt...official Home Page</td>
<td>Kristiansand Firede...departments of th...</td>
</tr>
</tbody>
</table>

[Kumar, Raghavan, Rajagopalan, Tomkins: Trawling the Web for emerging cyber-communities 1999]
Algorithmic result:
- Frequent itemset extraction and dynamic programming find graphs $K_{s,t}$
- Method is very scalable

Further improvements: Given $s$ and $t$
- (Repeatedly) prune out all nodes with out-degree $< t$ and in-degree $< s$