Social Network Analysis
for Computer Scientists

F.W. Takes

Leiden University, The Netherlands

Lecture 2 — Densest Subgraphs, Centrality & Distance Measures
So . . .

- Did you get my e-mail?
- Have you found a team mate?
- Did you see the course website?
  http://www.liacs.nl/~ftakes/SNACS
- Remember the lab session on September 19 at 9:00 in room 302?
- Have you taken a first look at the homework assignment?
- Any other questions?
Need an exit strategy?

Careers at Facebook

Software Engineering

Data Scientist

Menlo Park, CA

Facebook is seeking a Data Scientist to join our Data Science team. Individuals in this role are expected to be comfortable working as a software engineer and a quantitative researcher. The ideal candidate will have a keen interest in the study of an online social network, and a passion for identifying and answering questions that help us build the best products.
Need an exit strategy?

### Data Scientist Salaries in United States

Updated Oct 7, 2012 – Salaries posted anonymously by employees and employers

#### Change location
- **United States – All Cities**

<table>
<thead>
<tr>
<th>Location</th>
<th>Median</th>
<th>Low to High</th>
</tr>
</thead>
<tbody>
<tr>
<td>National</td>
<td>$117,500</td>
<td>$93k - $148k</td>
</tr>
</tbody>
</table>

#### 61 Salaries: 1–20 of 45 Job Titles

<table>
<thead>
<tr>
<th>Job Title</th>
<th>Avg. Salary</th>
<th>$50k</th>
<th>$100k</th>
<th>$150k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Scientist – Facebook 3 Salaries</td>
<td>$122,257</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Scientist – IMVU 3 Salaries</td>
<td>$120,000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Scientist – LinkedIn 2 Salaries</td>
<td>$108,349</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Scientist – StumbleUpon 2 Salaries</td>
<td>$112,190</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Scientist – Apollo Group 2 Salaries</td>
<td>$123,050</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sort by: Relevance (by Job Title)
Remember . . .

Concept

- Network (graph)
- Objects (nodes/vertices)
- Relations (links/edges)
- Number of nodes — $|V|$
- Number of edges — $|E|$
- Distance between nodes $u, v \in V$
- For now: unweighted unlabeled (un)directed graphs

Symbol

- $G = (V, E)$
- $V$
- $E$
- $n$
- $m$
- $d(u, v)$
Remember . . .

- Small world networks
  - Fat-tailed power-law degree distribution
  - Giant component
  - Low average node-to-node distance
  - Sparse networks (low density)

A social network
Clustering coefficient

- **Clustering coefficient**: extend to which nodes tend to cluster together (form “triangles” of connections)
- Node clustering coefficient for a node \( v \in V \):

\[
C(v) = \frac{2 \cdot |\{(u, w) \in E : (u, v) \in E \land (v, w) \in E\}|}{\deg(v) \cdot (\deg(v) - 1)}
\]
Clustering coefficient

- **Clustering coefficient**: extend to which nodes tend to cluster together (form “triangles” of connections)

- Node clustering coefficient for a node \( v \in V \):

\[
C(v) = \frac{2 \cdot |\{(u, w) \in E : (u, v) \in E \land (v, w) \in E\}|}{\deg(v) \ast (\deg(v) - 1)}
\]

\[
C(v) = \frac{2 \cdot \text{edges between neighbors of } v}{\text{maximum number of such edges}}
\]
Clustering coefficient

- **Clustering coefficient**: extend to which nodes tend to cluster together (form “triangles” of connections)
- Node clustering coefficient for a node \( v \in V \):

\[
C(v) = \frac{2 \cdot |\{(u, w) \in E : (u, v) \in E \land (v, w) \in E\}|}{\deg(v) \cdot (\deg(v) - 1)}
\]

\[
C(v) = \frac{2 \cdot \text{edges between neighbors of } v}{\text{maximum number of such edges}}
\]

- Average graph clustering coefficient for a graph \( G \):

\[
C(G) = \frac{1}{n} \cdot \sum_{v \in V} C(v)
\]
Situation A: $v$ has a clustering coefficient of 0

Situation B: $v$ has a clustering coefficient of $\frac{7}{10} = 0.7$

Small world networks: high average clustering coefficient compared to a random graph with the same number of vertices

Small world networks

- Fat-tailed power-law degree distribution
- Giant component
- Low average node-to-node distance
- Sparse networks (low density)
Small world networks

- Fat-tailed power-law degree distribution
- Giant component
- Low average node-to-node distance
- Sparse networks (low density)
- High average clustering coefficient
Subgraphs and Cliques

- Subgraph $G' = (V', E')$ of a graph $G = (V, E)$ with $V' \subseteq V$ and $E' \subseteq E$

- Complete graph: graph with all edges present

- **Clique**: complete subgraph

- Maximal clique: complete subgraph of maximal size (cannot be extended with another node)

- Maximum clique: largest possible clique in the graph

- **Clique problem**: find the maximum clique(s) of a graph (one of Karp’s 21 NP-complete problems introduced in 1972)
The subgraph with $V' = \{t, v, y\}$ is not complete.
The subgraph with $V' = \{t, v, y\}$ is not complete.

The subgraph with $V'' = \{t, u\}$ is complete and thus a clique of size 2.
The subgraph with $V' = \{t, v, y\}$ is not complete.

The subgraph with $V'' = \{t, u\}$ is complete and thus a clique of size 2.

The subgraph with $V''' = \{t, u, v\}$ is a maximal clique of size 3.
Subgraphs and Cliques

The subgraph with $V' = \{t, v, y\}$ is not complete.

The subgraph with $V'' = \{t, u\}$ is complete and thus a clique of size 2.

The subgraph with $V''' = \{t, u, v\}$ is a maximal clique of size 3.

The subgraph with $V'''' = \{v, w, x, z\}$ is a maximum clique of size 4.
Dense subgraphs

- Density: \( m/n(n - 1) \) (value \( \in [0; 1] \))
- Complete graphs have density 1
- Density (alternative): \( m/n \) (value \( \in [0; n - 1] \))
- Dense subgraph: subgraph with a high density (almost a clique)
- **Densest subgraph problem**: find a densest subgraph in an undirected graph (NP-hard; reduction from the clique problem)
- Applications in community detection (see later lecture)
- Many (exponential) algorithms have been proposed

Densest subgraph

- $S$ is a subgraph of $G$
- $S$ is not a clique, but it is dense (8 out of 10 edges present)

Iterative Algorithm

- Density (alternative): $m/n$ (value $\in [0; n - 1]$)
- Simple iterative algorithm to find the densest subgraph by Charikar et al.:
  1. Compute the average degree of the graph
  2. Delete all nodes whose degree is below the average
  3. Keep track of the density at each step
  4. Go to step 1 if nodes were deleted in this iteration
  5. Output the densest graph seen over all iterations

Charikar et al., "Greedy approximation algorithms for finding dense components in a graph", in LNCS 1913, pp. 84–95, 2000.
Iterative Algorithm

- Iteration 1: Current density $16/11 = 1.45$, avg. degree = 2.9
- Best density (iteration) = \ldots

Iterative Algorithm

- **Iteration 1**: Current density $16/11 = 1.45$, avg. degree $= 2.9$
- **Best density (iteration) = 1.45 (1)**

Iterative Algorithm

- **Iteration 2:** Current density \( \frac{9}{5} = 1.8 \), avg. degree = 3.6
- Best density (iteration) = 1.45 (1)

Iterative Algorithm

- Iteration 2: Current density $9/5 = 1.8$, avg. degree = 3.6
- Best density (iteration) = 1.8 (2)

Iterative Algorithm

- **Iteration 3**: Current density $\frac{3}{3} = 1$, avg. degree $= 2$
- Best density (iteration) $= 1.8$ (2) (unchanged)

Iterative Algorithm

- No node deleted in previous iteration, algorithm terminates
- Best density (iteration) = 1.8 (2)

Iterative Algorithm Performance

FLICKR: Remaining graph vs passes

IM: Remaining graph vs passes
Dense subgraphs for community detection
Centrality

- Given a social network, which person is most important or most influential?
- What is the most important page on the web?
- Which protein is most vital in a biological network?
- Who is the most respected author in a scientific citation network?
- What is the most crucial router in an internet topology network?
Centrality

- **Node centrality**: the importance of a node with respect to the other nodes based on the structure of the network
- **Centrality measure**: computes the centrality value of all nodes in the graph
- For all $v \in V$ a measure $M$ returns a value $C_M(v) \in [0; 1]$
- But what is the ground truth to verify these measures?
Centrality

- **Node centrality**: the importance of a node with respect to the other nodes based on the structure of the network
- **Centrality measure**: computes the centrality value of all nodes in the graph
- For all $v \in V$ a measure $M$ returns a value $C_M(v) \in [0; 1]$
- But what is the ground truth to verify these measures?
  - Hard to say!
Centrality

- **Node centrality**: the importance of a node with respect to the other nodes based on the structure of the network

- **Centrality measure**: computes the centrality value of all nodes in the graph

  For all $v \in V$ a measure $M$ returns a value $C_M(v) \in [0; 1]$

- But what is the ground truth to verify these measures?
  - Hard to say!

- Related problem — Edge centrality: which edge/connection between nodes is most important?
Centrality

- **Node centrality**: the importance of a node with respect to the other nodes based on the structure of the network
- **Centrality measure**: computes the centrality value of all nodes in the graph
- For all $v \in V$ a measure $M$ returns a value $C_M(v) \in [0; 1]$
- But what is the ground truth to verify these measures?
  - Hard to say!
- Related problem — Edge centrality: which edge/connection between nodes is most important?
  - Often the edges between nodes with a high node centrality value
Degree centrality

- Undirected graphs – **degree centrality**: measure the number of adjacent nodes
  \[ C_d(v) = \frac{\text{deg}(v)}{n - 1} \]
- Directed graphs — indegree centrality and outdegree centrality
- Local measure
- \( O(1) \) time to compute
Degree distribution

- Not so many distinct values in the lower ranges
Degree centrality
Closeness centrality

- **Closeness centrality**: the average distance to each other node in the graph

\[
C_c(v) = \frac{1}{n-1} \sum_{w \in V} d(v, w)
\]

- Global distance-based measure
- Connected component(s)...
- \(O(mn)\) to compute: one BFS in \(O(m)\) for each of the \(n\) nodes
Breadth First Search (BFS)

- Traverse the graph as a tree in level-order, skipping nodes that have already been visited
- Consider each of the $m$ edges once, so $O(m)$ time
Closeness centrality
Degree vs. Closeness

Figure: Node size based on degree, color based on closeness
Betweenness centrality

- **Betweenness centrality**: measure the number of shortest paths that run through a node

\[
C_b(u) = \sum_{v, w \in V \setminus \{v, w\}} \frac{\sigma_u(v, w)}{\sigma(v, w)}
\]

- \(\sigma(v, w)\) is the number of shortest paths from \(v\) to \(w\)
- \(\sigma_u(v, w)\) is the number of such shortest paths that run through \(u\)
- Divide by largest value to normalize to \([0; 1]\)
- Global path-based measure
- \(O(2mn)\) time to compute (two “BFSes” for each node)

Counting shortest paths

- Bellman criterion: $v$ lies on a shortest path from $u$ to $w$ if $d(u, v) + d(v, w) = d(u, w)$.

- Predecessors: $P_u(w)$ is the set of predecessors of node $w$ on a shortest path from $u$, formally:

$$P_u(w) = \{ v \in V : (v, w) \in E, d(u, w) = d(u, v) + 1 \}$$

- Counting the number of shortest paths $\sigma(u, w)$ for $u \neq w$:

$$\sigma(u, w) = \sum_{v \in P_u(w)} \sigma(u, v)$$

with $\sigma(u, u) = 1$
Counting shortest paths

Task: compute $\sigma(u, w)$, the number of shortest paths from $u$ to $w$ ...
Counting shortest paths

Task: compute $\sigma(u, w)$, the number of shortest paths from $u$ to $w$
First a BFS from $u$ to find $w$
Counting shortest paths

Task: compute $\sigma(u, w)$, the number of shortest paths from $u$ to $w$
First a BFS from $u$ to find $w$
Counting shortest paths

Task: compute $\sigma(u, w)$, the number of shortest paths from $u$ to $w$
First a BFS from $u$ to find $w$
Counting shortest paths

Task: compute $\sigma(u, w)$, the number of shortest paths from $u$ to $w$
First a BFS from $u$ to find $w$
Counting shortest paths

Task: compute $\sigma(u, w)$, the number of shortest paths from $u$ to $w$

First a BFS from $u$ to find $w$
Counting shortest paths

Task: compute $\sigma(u, w)$, the number of shortest paths from $u$ to $w$

$\sigma(u, w)$? Ask predecessors $P_u(w) = \{ h \}$ for its value
Counting shortest paths

Task: compute $\sigma(u, w)$, the number of shortest paths from $u$ to $w$

$\sigma(u, h)$? Ask predecessors $P_u(h) = \{d, e, f\}$ for its value
Counting shortest paths

Task: compute $\sigma(u, w)$, the number of shortest paths from $u$ to $w$

Ask predecessors . . .
Counting shortest paths

Task: compute $\sigma(u, w)$, the number of shortest paths from $u$ to $w$
Ask predecessors . . .
Counting shortest paths

Task: compute $\sigma(u, w)$, the number of shortest paths from $u$ to $w$.

Ask predecessors ...
Counting shortest paths

Task: compute $\sigma(u, w)$, the number of shortest paths from $u$ to $w$

$\sigma(u, u) = 1$, now propagate back...
Counting shortest paths

Task: compute $\sigma(u, w)$, the number of shortest paths from $u$ to $w$

$$\sigma(u, a) = \sigma(u, c) = 1$$
Counting shortest paths

Task: compute $\sigma(u, w)$, the number of shortest paths from $u$ to $w$

$$\sigma(u, b) = \sigma(u, a) + \sigma(u, c) = 2$$
Counting shortest paths

Task: compute $\sigma(u, w)$, the number of shortest paths from $u$ to $w$

$\sigma(u, d) = \sigma(u, e) = \sigma(u, f) = \sigma(u, c) = 2$
Counting shortest paths

Task: compute $\sigma(u, w)$, the number of shortest paths from $u$ to $w$

$$\sigma(u, h) = \sigma(u, d) + \sigma(u, e) + \sigma(u, f) = 6$$
Counting shortest paths

Done!

\[ \sigma(u, w) = \sigma(u, h) = 6 \]
Betweenness centrality (1)
Betweenness centrality (2)
Degree vs. Betweenness

Figure: Node size based on degree, color based on betweenness
Centrality measures compared

Figure: Degree, closeness and betweenness centrality

Source: "Centrality" by Claudio Rocchini, Wikipedia File:Centrality.svg
Eccentricity centrality

- **Node eccentricity**: length of a longest shortest path (distance to a node furthest away)

\[ \varepsilon(v) = \max_{w \in V} d(v, w) \]

- **Eccentricity centrality**:

\[ C_e(v) = \frac{1}{\varepsilon(v)} \]

- Worst-case variant of closeness centrality
- \( O(mn) \) to compute: one BFS in \( O(m) \) for each of the \( n \) nodes
Eccentricity centrality

- **Node eccentricity**: length of a longest shortest path (distance to a node furthest away)
  \[
  \varepsilon(v) = \max_{w \in V} d(v, w)
  \]

- **Eccentricity centrality**:
  \[
  C_e(v) = \frac{1}{\varepsilon(v)}
  \]

- Worst-case variant of closeness centrality
- \(O(mn)\) to compute: one BFS in \(O(m)\) for each of the \(n\) nodes
Eccentricity centrality
Centrality measures so far

- Distance/path-based measures:
  - Degree centrality $O(1)$
  - Closeness centrality $O(mn)$
  - Betweenness centrality $O(mn)$
  - Eccentricity centrality $O(mn)$

- Approximation of these measures is possible

- Next week: propagation-based centrality measures
Course project

- In teams
- Presentation (30% of your grade) on a course-related CS paper
  - 30 minutes for your talk
  - 15 minutes for questions and discussion
- Paper (50% of your grade) comparing algorithms from three to five course-related CS papers
  - Write in two parts with deadlines
  - Peer reviewing for first part
Presentation

- Present the paper that you studied
- Show a nice demo, pictures, movies or visualization
- Have a clearly structured presentation
- Briefly discuss to which other techniques you are going to compare this method in your project
Course project

- Read your papers
- Define and confine the exact problem
- Determine which techniques you are going to compare
- Program (or obtain code of) the different algorithms and techniques
- Get some applicable datasets for comparing the algorithms
- Perform experiments to compare the algorithms
- Determine and discuss results
- Write a sensible conclusion
Course project paper

- Scientific paper
- \LaTeX
- 8 to 12 pages
- Images, figures, graphs, diagrams, tables
- Between 5 and 9 sections
- Peer review after first 3 sections
- Option for “intermediary paper check” before final hand-in
Course project schedule

- Sep 22: deadline for e-mailing your team’s course project topic
- From Oct 10 onwards: presentations by students
- Oct. 24: deadline for handing in the first three sections to other team for peer review
- Nov. 1: deadline for giving the other team your review
- Nov. 21: optional deadline for “intermediary paper check”
- Dec. 5: deadline for full course project paper
Possible talk structure

- Introduction and motivation of the problem
- Related and previous work
- Formal definition of the problem
- Solution of the problem, algorithms, techniques
- Experimental setup: datasets used, verification measures, etc.
- Results: how well and in which cases does the proposed technique work well?
- Conclusion and future work
For next week . . .

- Remember the lab session on September 19 at 9.00 in room 302
- Make sure that you have access to the student workstations with your ULCN-account
- E-mail your preferred topic for the course project (presentation and paper) before September 22
- Think about presenting on October 10
- Make serious progress with the homework assignment
End

BRUTE-FORCE SOLUTION: $O(n!)$

DYNAMIC PROGRAMMING ALGORITHMS: $O(n^22^n)$

SELLING ON EBAY: $O(1)$

STILL WORKING ON YOUR ROUTE?

SHUT THE HELL UP.

http://xkcd.com/399