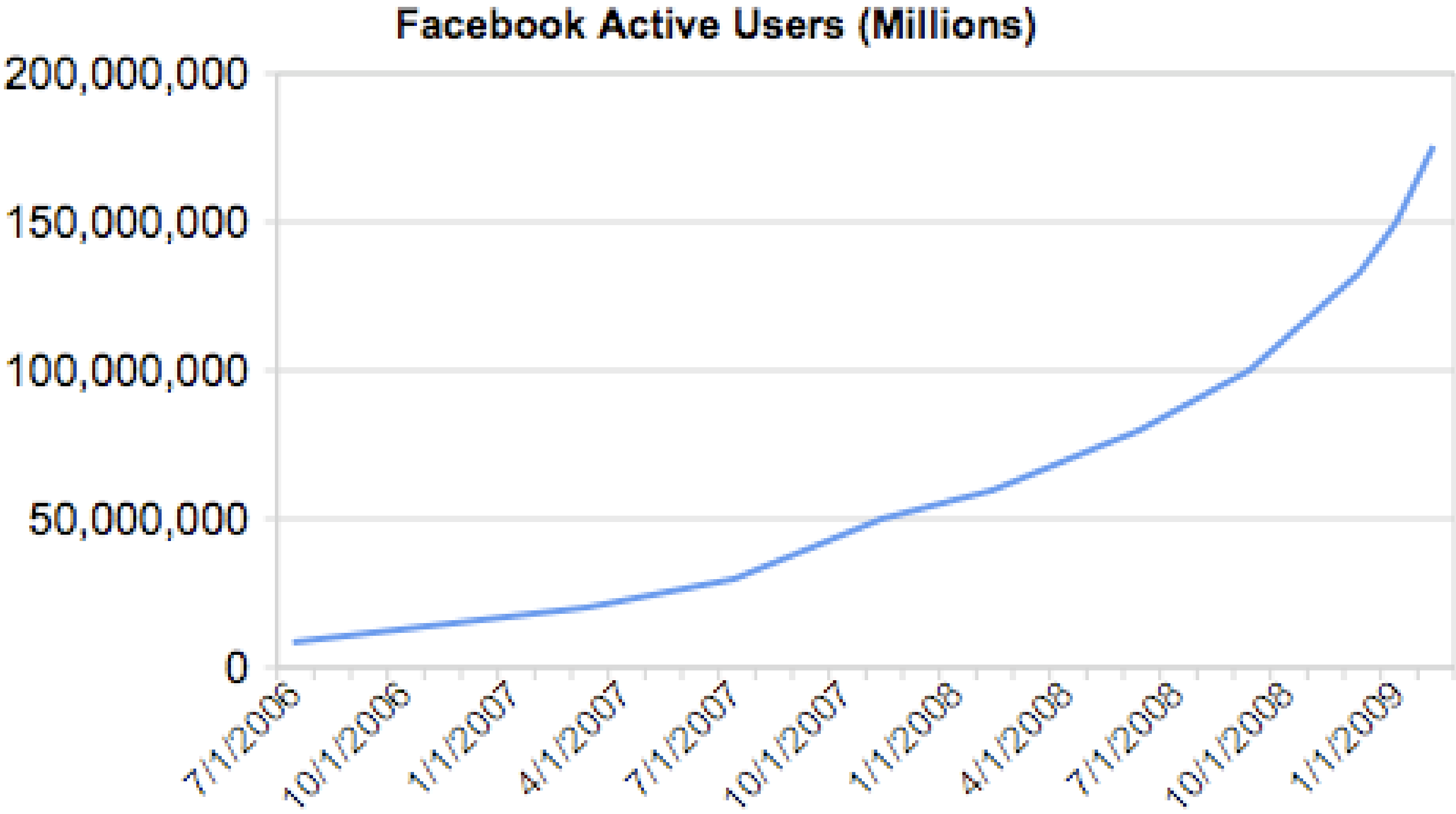
Privacy Implications of Public Listings on Social Networks

Security Seminar
24 March 2009

Joseph Bonneau, Jonathan Anderson,
Frank Stajano, Ross Anderson
Security Research Group
Computer Laboratory
UNIVERSITY OF CAMBRIDGE

Why Facebook Matters



Why Facebook Matters

- Over 190 M users
- 70% outside of the USA and growing
  - Growth rates for 2008 around the world
    - Italy: 2900%
    - Argentina: 2000%
    - Indonesia: 600%
    - France: 400%
- Fast growing age segment: 55+

Why Facebook Is Different

- Most users provide accurate data
- High level of disclosure
- Aggressive policing of false profiles
- Limit on number of friends at 1K
- Rich ACL settings available
- Based on University/Regional networks
- Most users don't consider their profiles “public”

Why Facebook Is Different

- Users feel an intimate connection
- Huge backlashes against changes:
  - News Feed (Sep 2006)
  - Beacon (Nov 2007)
  - “New Facebook” (Sep 2008)
  - Terms of Use (Feb 2009)
  - New Product Pages (Mar 2009)

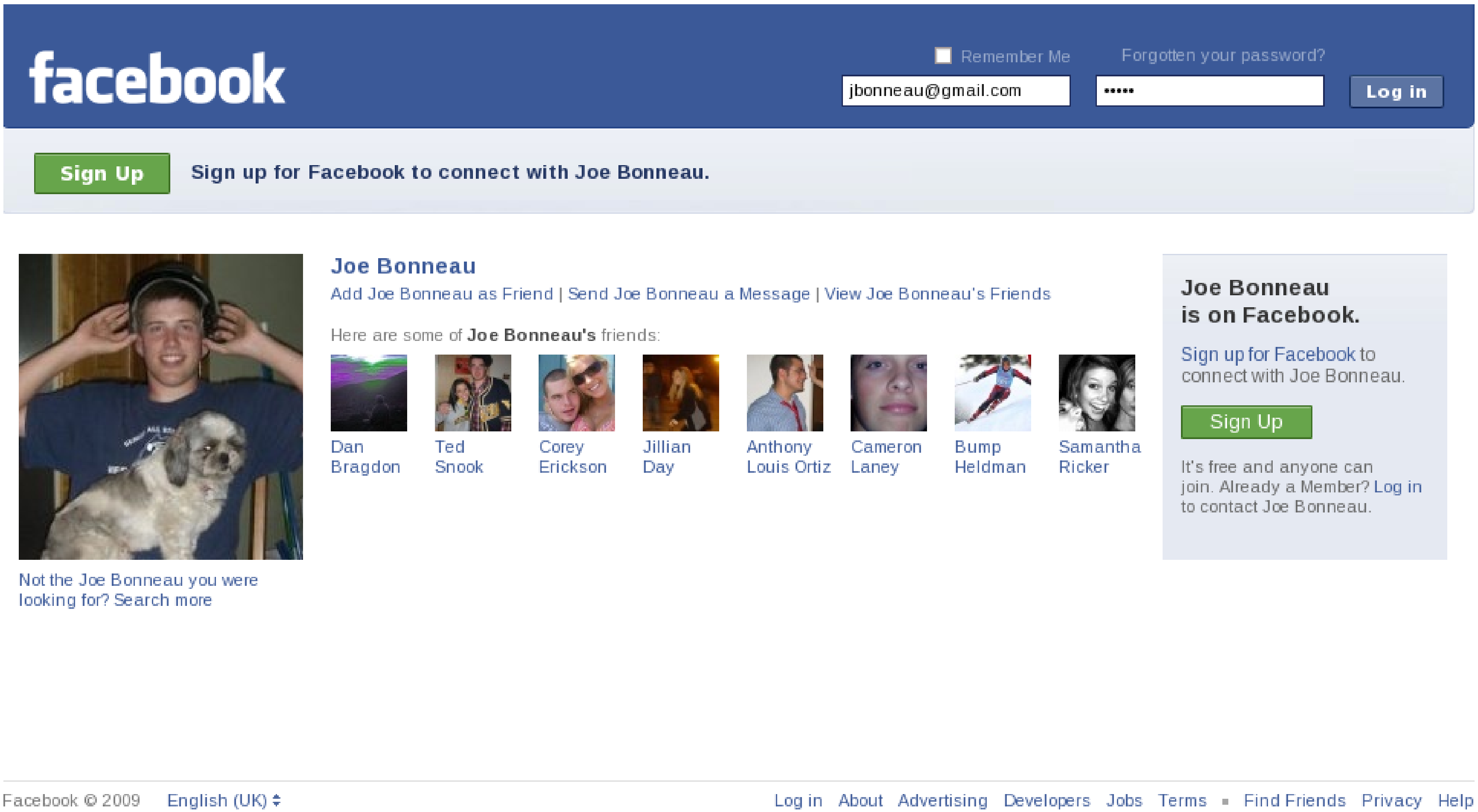
A Quietly Introduced Feature...

Public Search Listings, Sep 2007

Public Search Listings

- Unprotected against crawling
- Contains name, location, 8 friends
- Indexed by search engines
- Opt out—but most users don't know it exists!

Utility



Entity Resolution

Utility

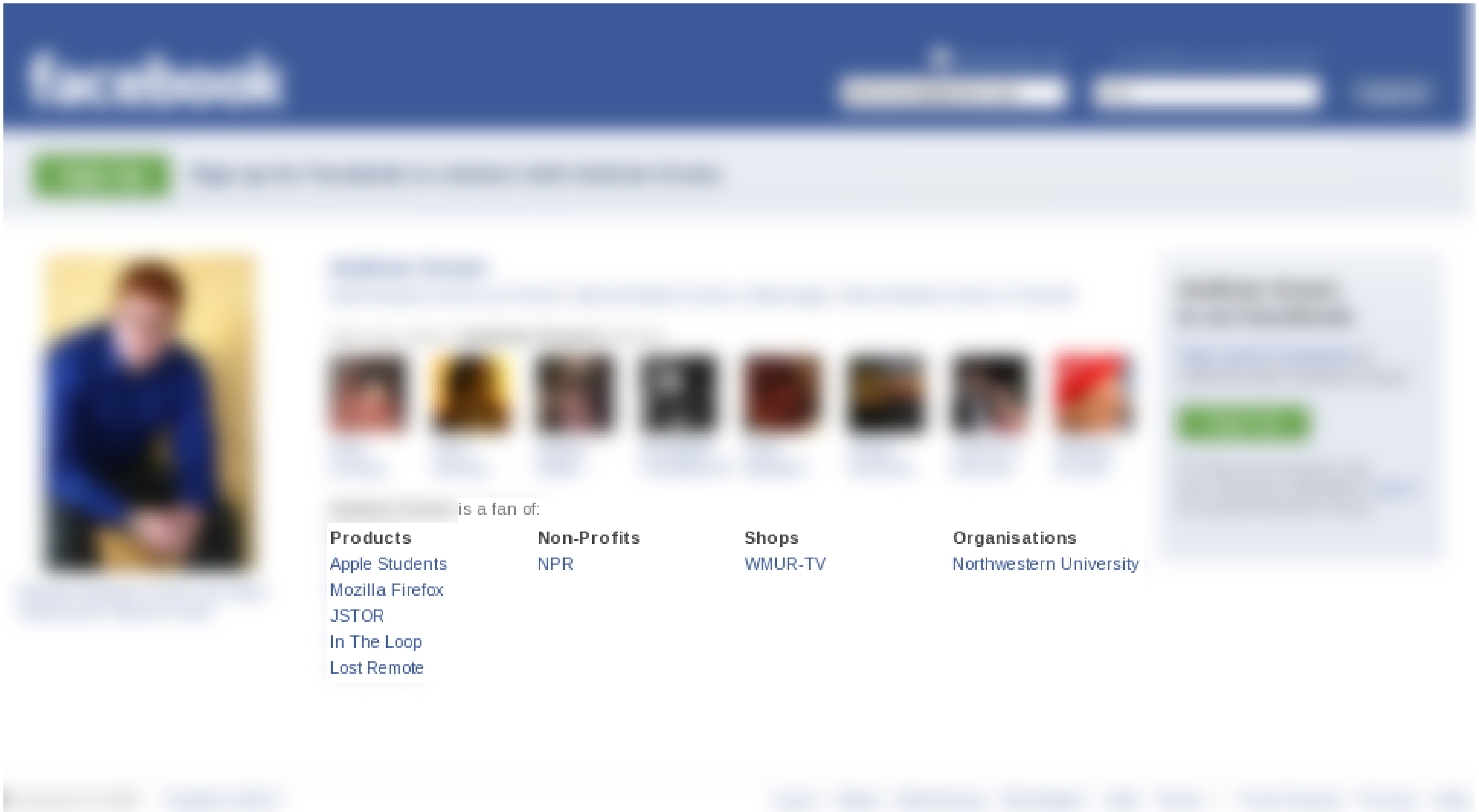
Promotion via Network Effects

Legal Status

“Your name, network names, and profile picture thumbnail will be available in search results across the Facebook network and those limited pieces of information may be made available to third party search engines. This is primarily so your friends can find you and send a friend request.”

-Facebook Privacy Policy

Legal Status



Much More Info Now Included...

Legal Status

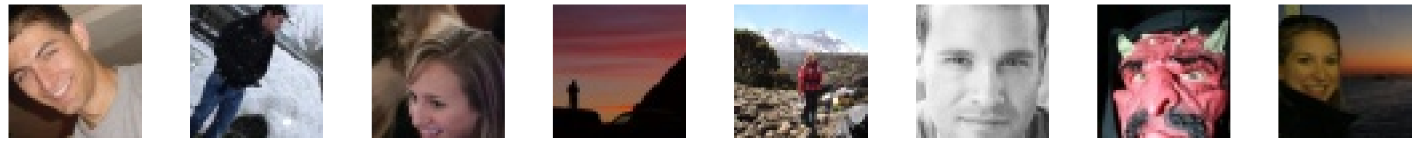
Public Group Pages Recently Added

Obvious Attack

- Initially returned new friend set on refresh
- Can find all *n* friends in $O(n \cdot \log n)$ queries
  - The Coupon Collector's Problem
  - For 100 Friends, need 65 page refreshes
- As of Jan 2009, friends fixed per IP address

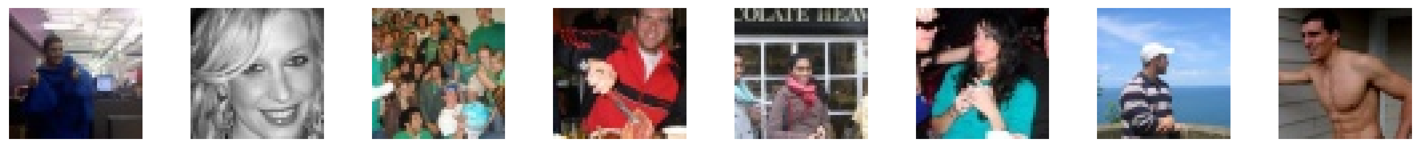
Fun with Tor

UK



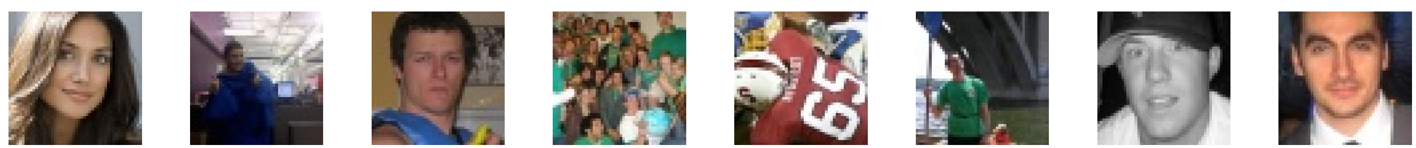
David Cottingham
Eirik George Tsarpalis
Emma Alden
Luke Church
Stella Nordhagen
David J Hornsby
Justin Palfreyman
Jillian Sullivan

Germany



Shoshana Freisinger
Lauren Duffey
Conor Loftus-Sweetland
Will Cordingley
Srilakshmi Raj
Sarita Kristina Sylvester
Brian Brown
Gary Champagne

USA



Melanie Kannokada
Shoshana Freisinger
Russ Heddleston
Conor Loftus-Sweetland
Gustav Rydstedt
Seth Ort
Cameron Lochte
Ben Skolnik

Australia

Shoshana Freisinger
Federico Baradello
Lauren Duffey
Adrian Boscolo-Hightower
Justin David Carl
Katie Gunderson
Ankit Garg
Srilakshmi Raj

Attack Scenario

- Spider all public listings
  - Our experiments crawled 250 k users daily
  - Implies ~800 CPU-days to recover all users
- Compute functions on sampled graph

Abstraction

- Take a graph $G = <V,E>$
- Randomly select k out-edges from each node
- Result is a sampled graph $G_k = <V,E_k>$
- Try to approximate $f(G) \approx f_{\text{approx}}(G_k)$

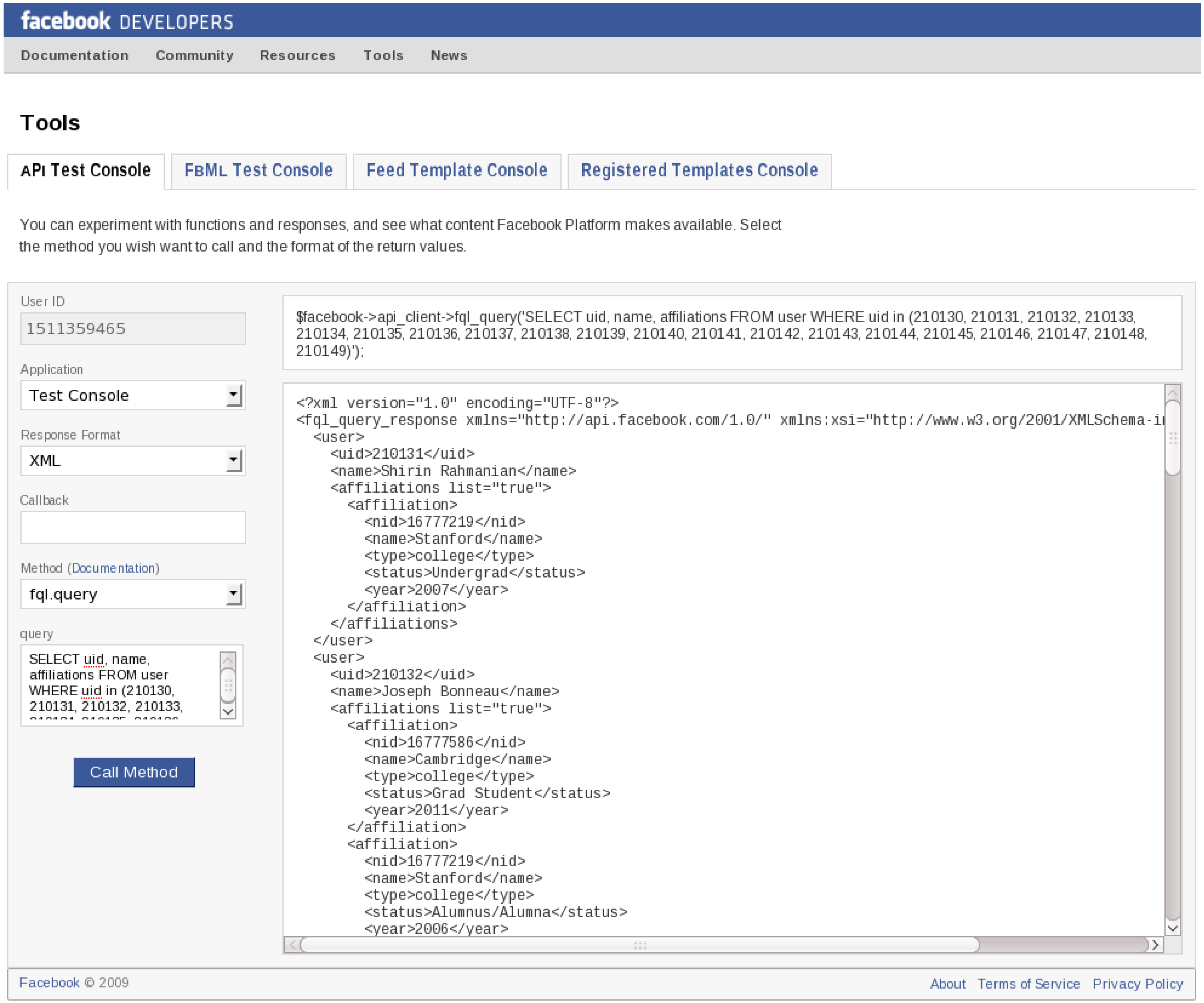
Approximable Functions

- Node Degree
- Dominating Set
- Betweenness Centrality
- Path Length
- Community Structure

Our Data Set

- Only have sampled graph from public crawls
- Need a complete network for testing
- Solution: Facebook Developer's API

Facebook Query Language



Facebook Query Language

- Easy to get (name, UID) pairs:

```
SELECT uid, name FROM user
    WHERE uid IN (0, 1, 2, ... N);
```

- Can query for N ≈ 1k without timeouts

Facebook Query Language

facebook Home Profile Friends Inbox Fitz Renfold Settings Log out Search

ID Query:

210130-210150
SELECT uid, name, affiliations FROM user WHERE uid in (210130, 210131, 210132, 210133, 210134, 210135, 210136, 210137, 210138, 210139, 210140, 210141, 210142, 210143, 210144, 210145, 210146, 210147, 210148, 210149)
Results

Found 13 users
210131 Shirin Rahmanian (Stanford)
210132 Joseph Bonneau (Cambridge Stanford San Francisco, CA)
210137 Jen Cowman (Minneapolis / St. Paul, MN Stanford Northwestern 3M)
210139 Francis Ring (Stanford San Diego, CA)
210140 Robert Negrete (Stanford)
210141 Nicholas Lovell (Stanford Microsoft)
210143 Lisa Feng Yung Chen (Stanford Cornerstone Research)
210145 Weisheng Lee (Stanford)
210147 Pomo Micha (Stanford)
210148 Sheila Dharmarajan (Stanford)
210149 Matt Green (Stanford New York, NY)

11 named users
users

Page built by Stool · Contact · Report About Advertising Developers Jobs Terms · Find Friends Privacy Account Help

Crawled Stanford ID spaces in 1 hour (30 k UIDs)

Facebook Query Language

- Given UID list, extract friendship links:

```
SELECT uid1, uid2 FROM friend
   WHERE uid1 IN (0, 1, 2, ... N);
   AND   uid2 IN (0, 1, 2, ... N);
```

- Can query for N ≈ 1k without timeouts

Facebook Query Language

facebook Home Profile Friends Inbox Fitz Renfold Settings Log out Search

Link Query:

Executed Query

select uid1, uid2 from friend where (uid1=1036 OR uid1=1037 OR uid1=1038 OR uid1=1039 OR uid1=1040 OR uid1=1041 OR uid1=1042 OR uid1=1044 OR uid1=1045 OR uid1=1046 OR uid1=1047 OR uid1=1048 OR uid1=1049) AND (uid2=1050 OR uid2=1052 OR uid2=1053 OR uid2=1054 OR uid2=1056 OR uid2=1057 OR uid2=1058 OR uid2=1059 OR uid2=1060 OR uid2=1061 OR uid2=1062 OR uid2=1063)

Results

Found 8 links

1038 1050
1038 1052
1038 1053
1038 1060
1042 1050
1045 1058
1048 1050
1049 1050
Saved 8 links

Page built by Stool ▪ Contact ▪ Report About Advertising Developers Jobs Terms ▪ Find Friends Privacy Account Help

Extracted Friendship Links in < 6 hours

FQL Advantages

- Extracted all users not opted-out of FB platform

  (~99% of users)

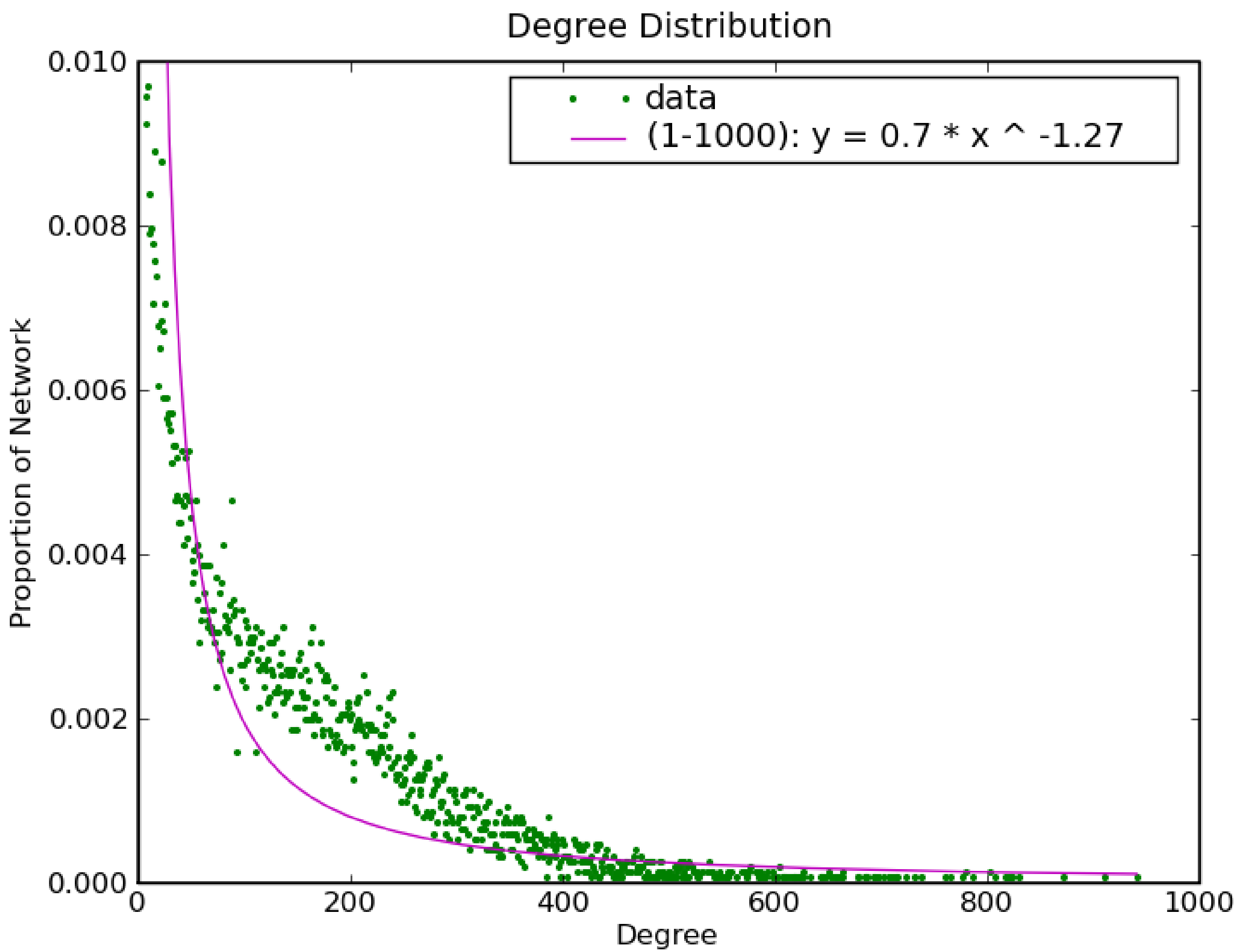
- Crawling method doesn't scale—O(n^2) queries

Experimental Data

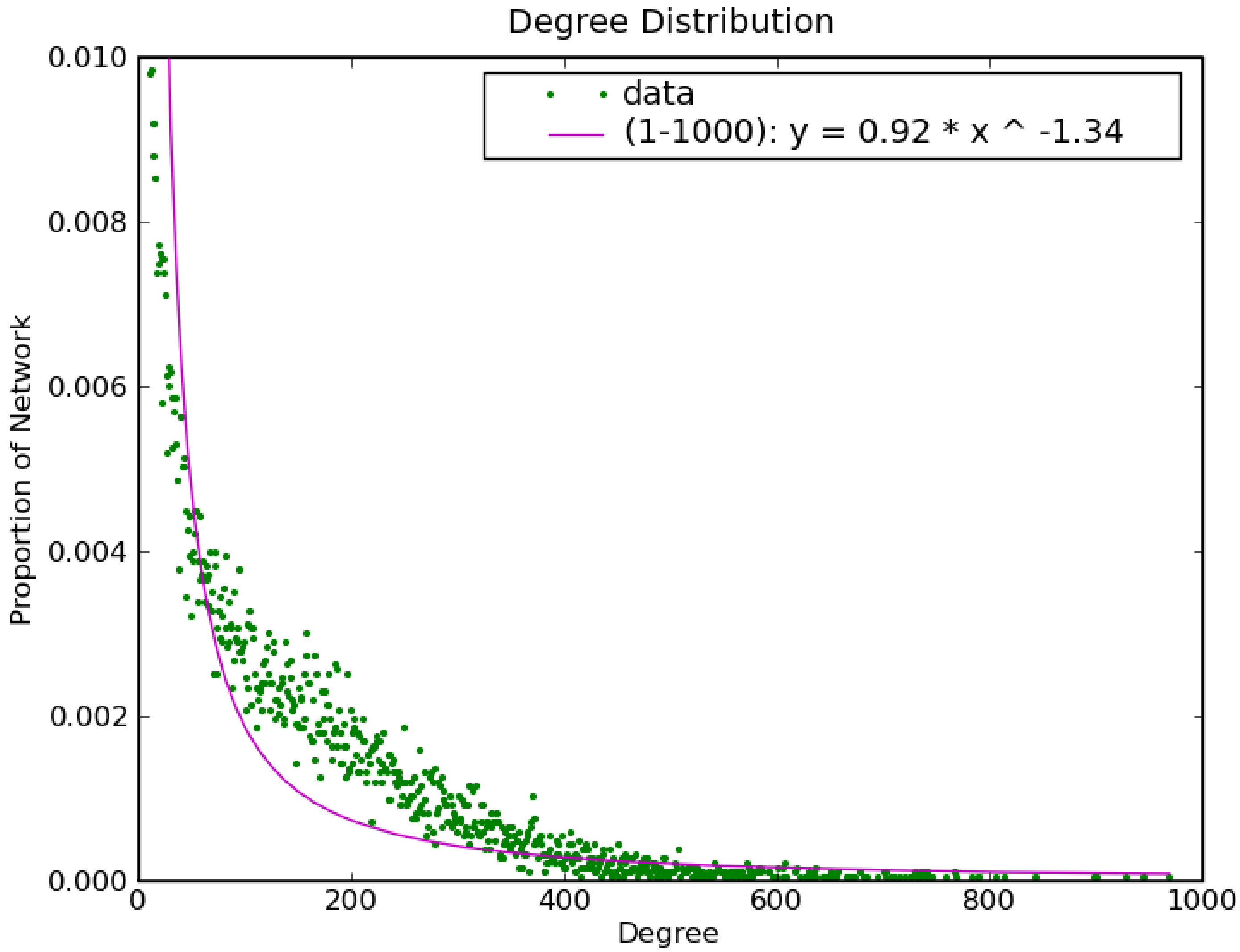
- Crawled original Stanford, Harvard networks
  - From era when UIDs assigned sequentially
- Representative sub-networks

	# Users	Mean *d*	Median *d*
Stanford	15043	125	90
Harvard	18273	116	76

Stanford Histogram

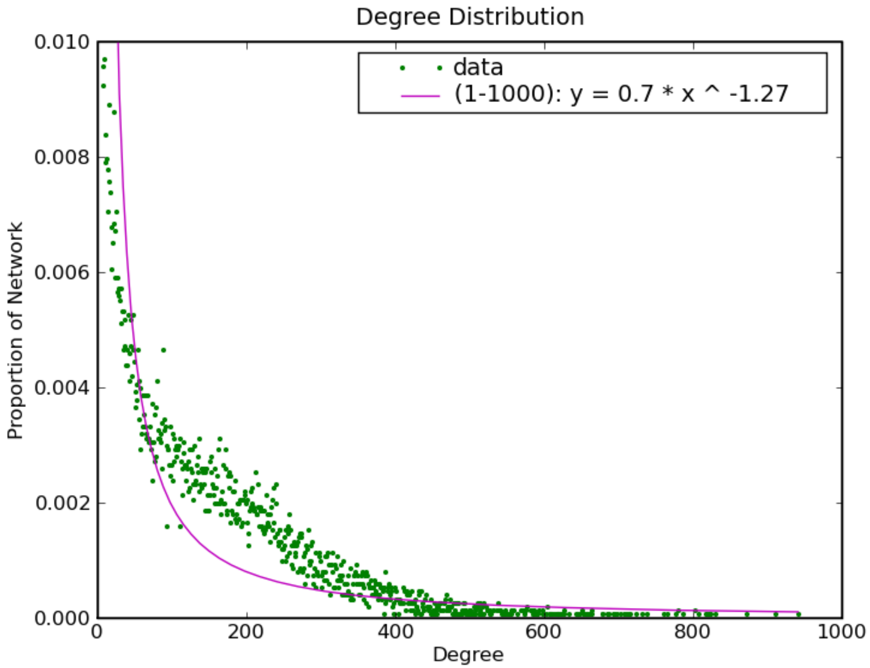


Harvard Histogram

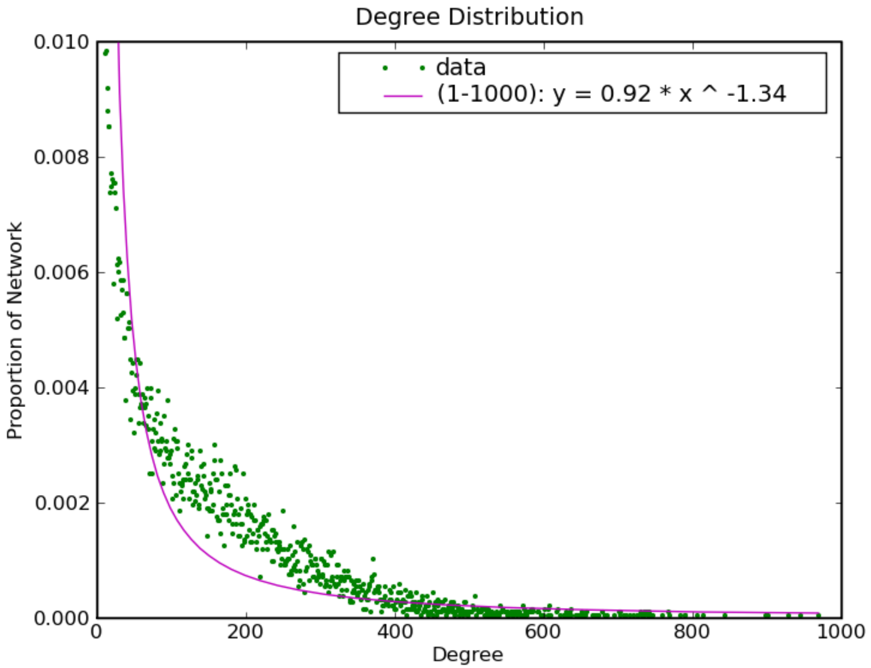


Comparison

Stanford

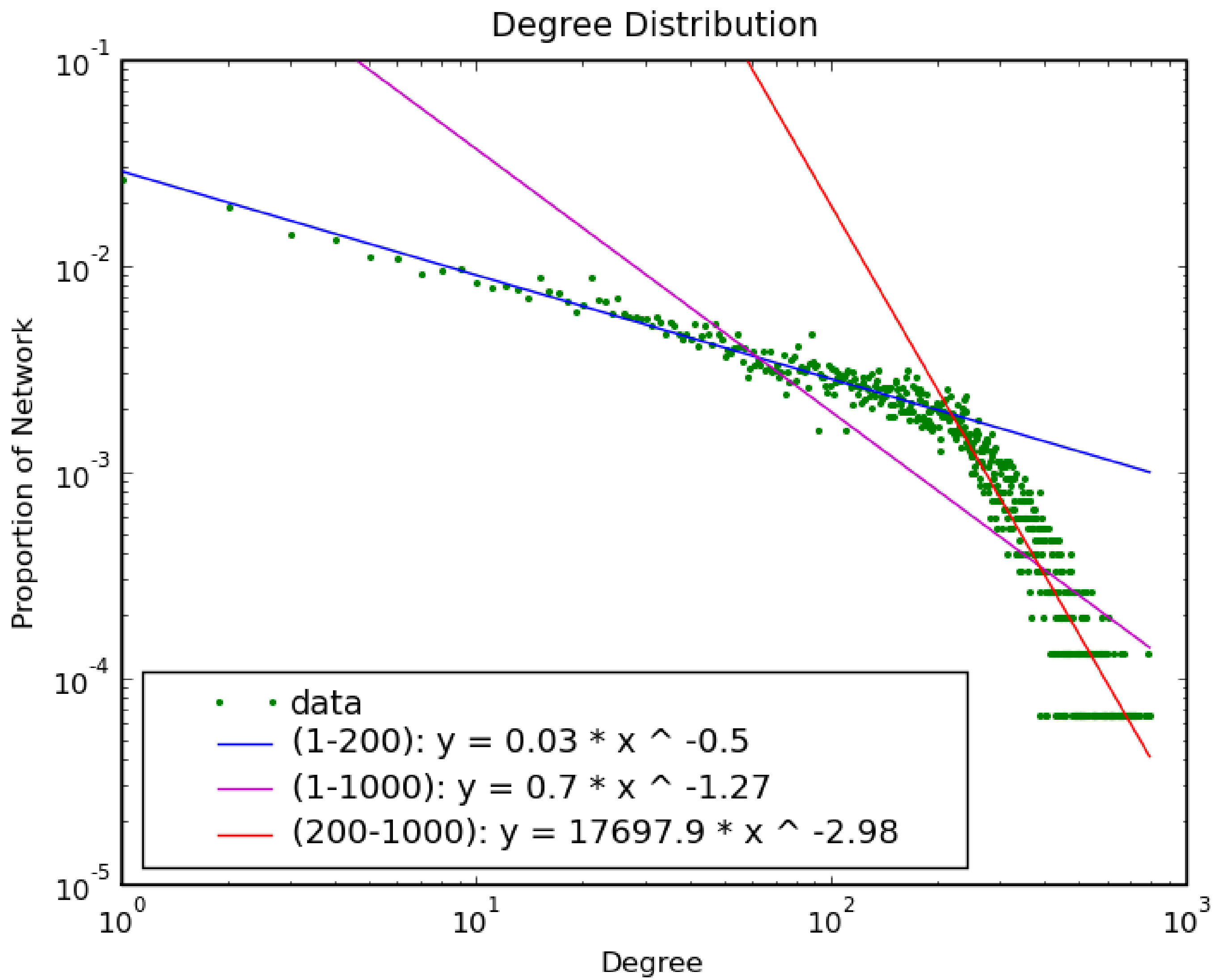


Harvard



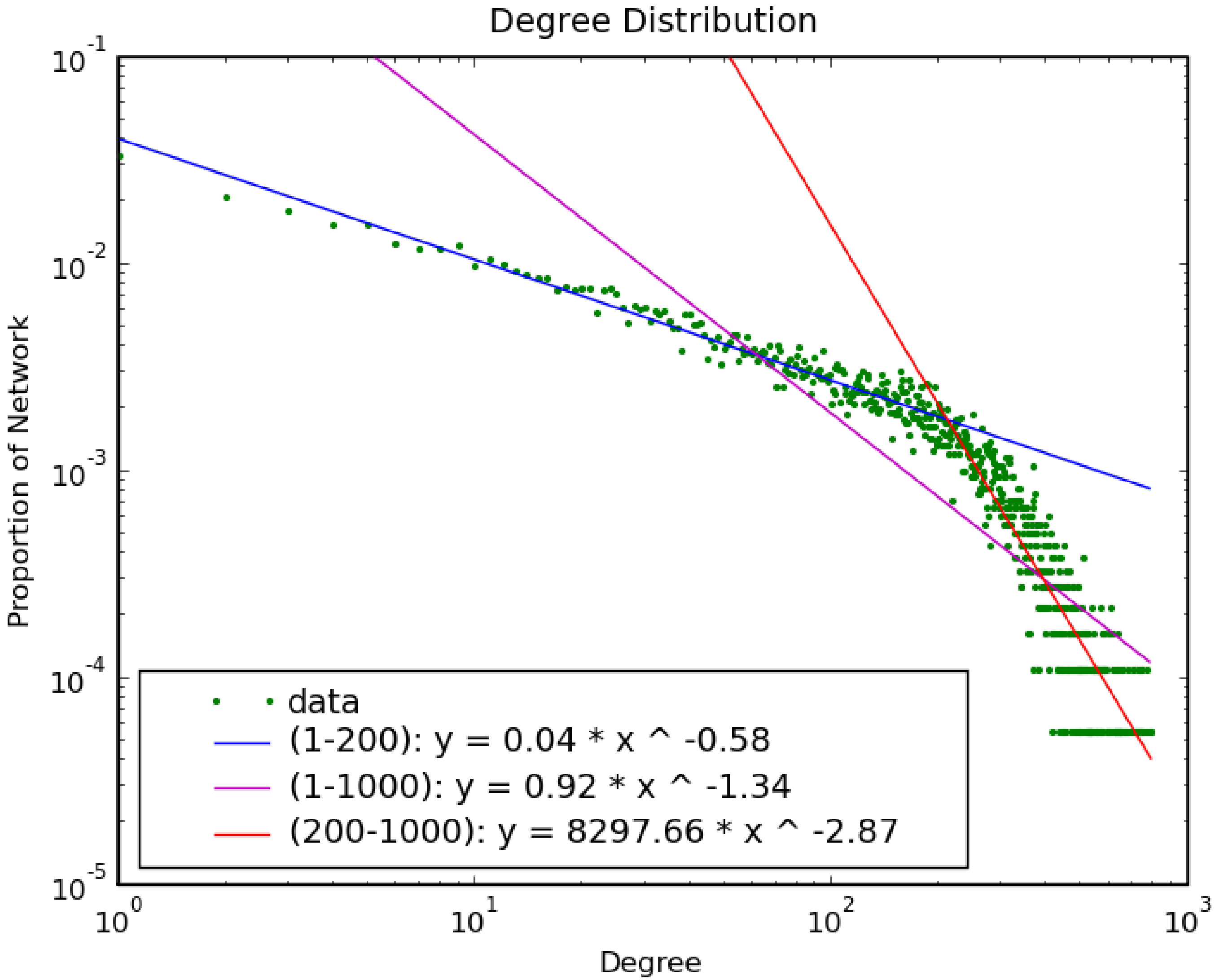
Networks have very similar structure

Stanford Log-Log plot



Apparent discontinuity at *d* = 200. Dunbar's number?

Harvard Log-Log plot



Apparent discontinuity at d = 200. Dunbar's number?

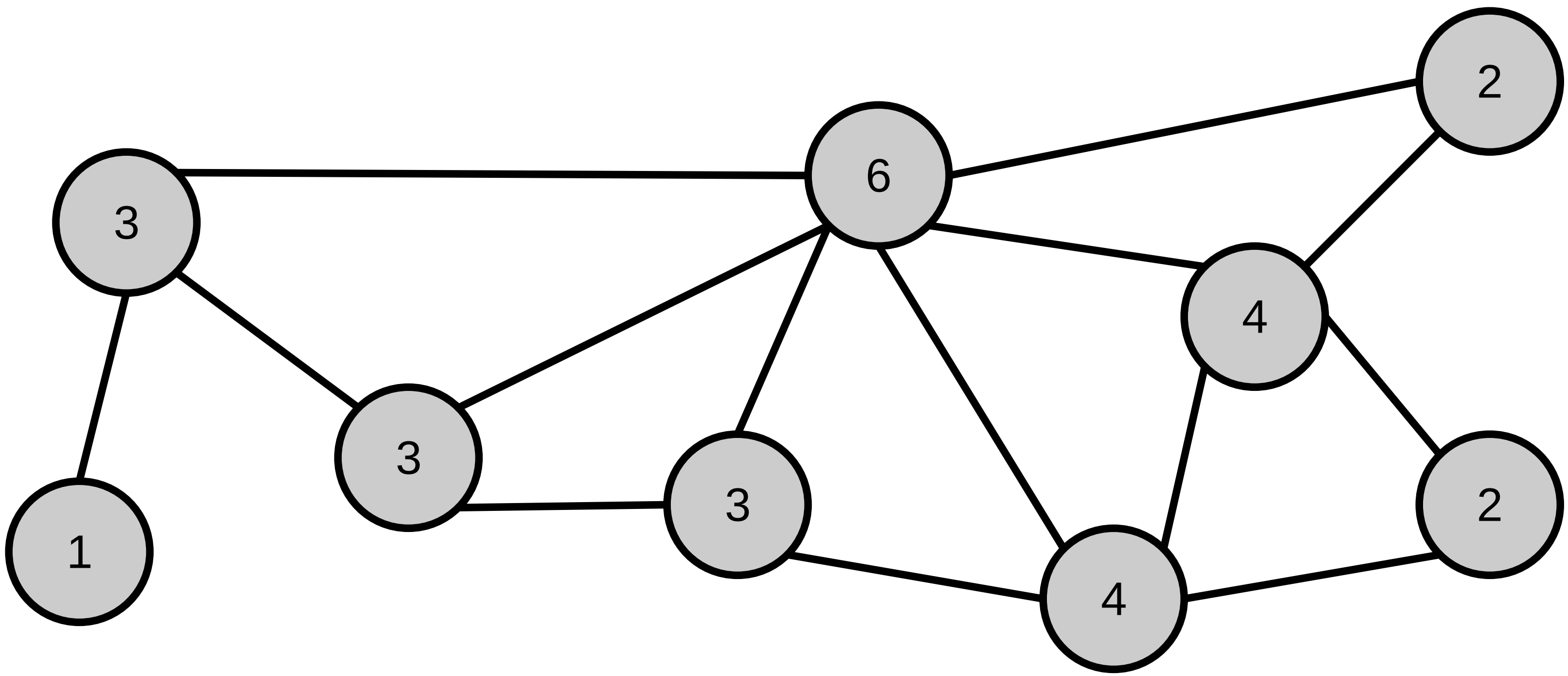
Back To Our Abstraction

- Take a graph $G = <V,E>$
- Randomly select k out-edges from each node
- Result is a sampled graph $G_k = <V,E_k>$
- Try to approximate $f(G) \approx f_{\text{approx}}(G_k)$

Estimating Degrees

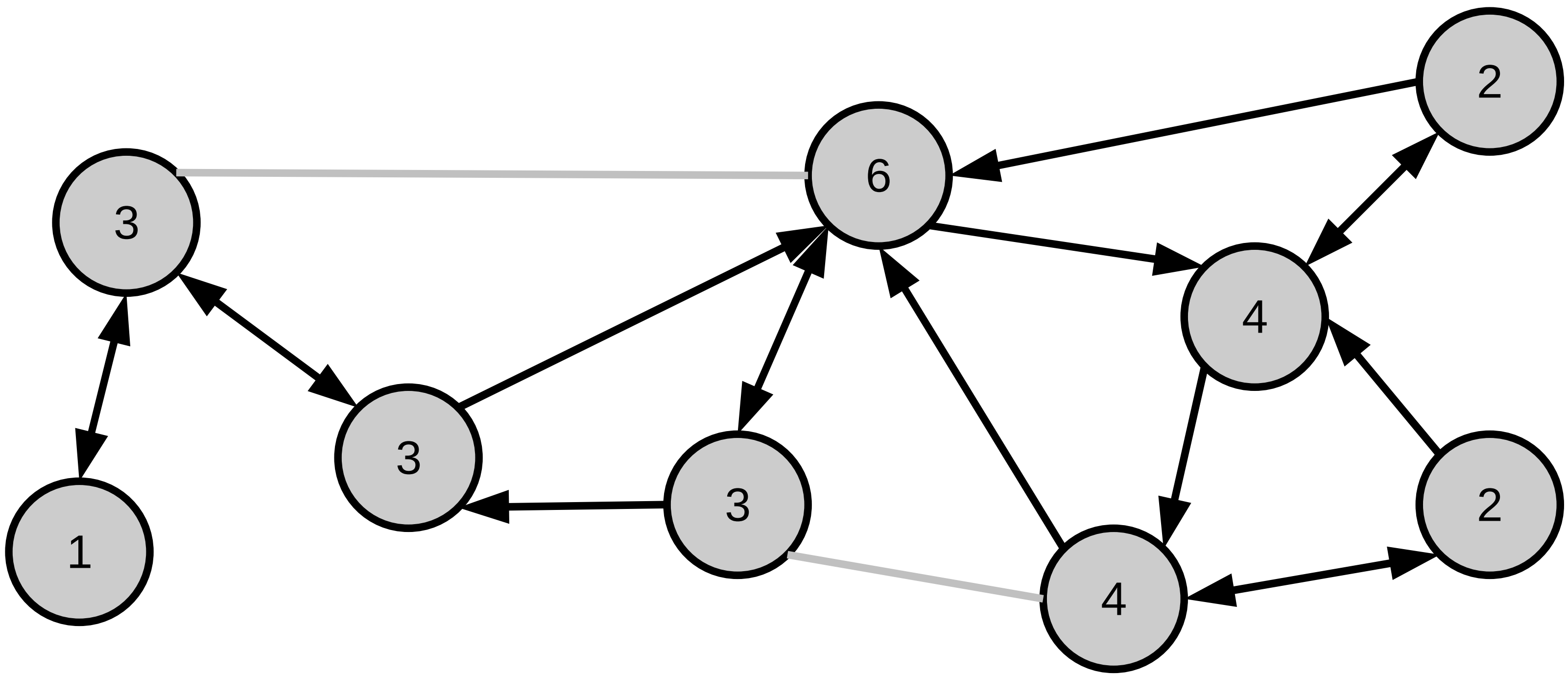
- Convert sampled graph into a directed graph
  - Edges originate at the node where they were seen
- Learn exact degree for nodes with degree < *k*
  - Less than k out-edges
- Get random sample for nodes with degree ≥ *k*
  - Many have more than *k* in-edges

Estimating Degrees



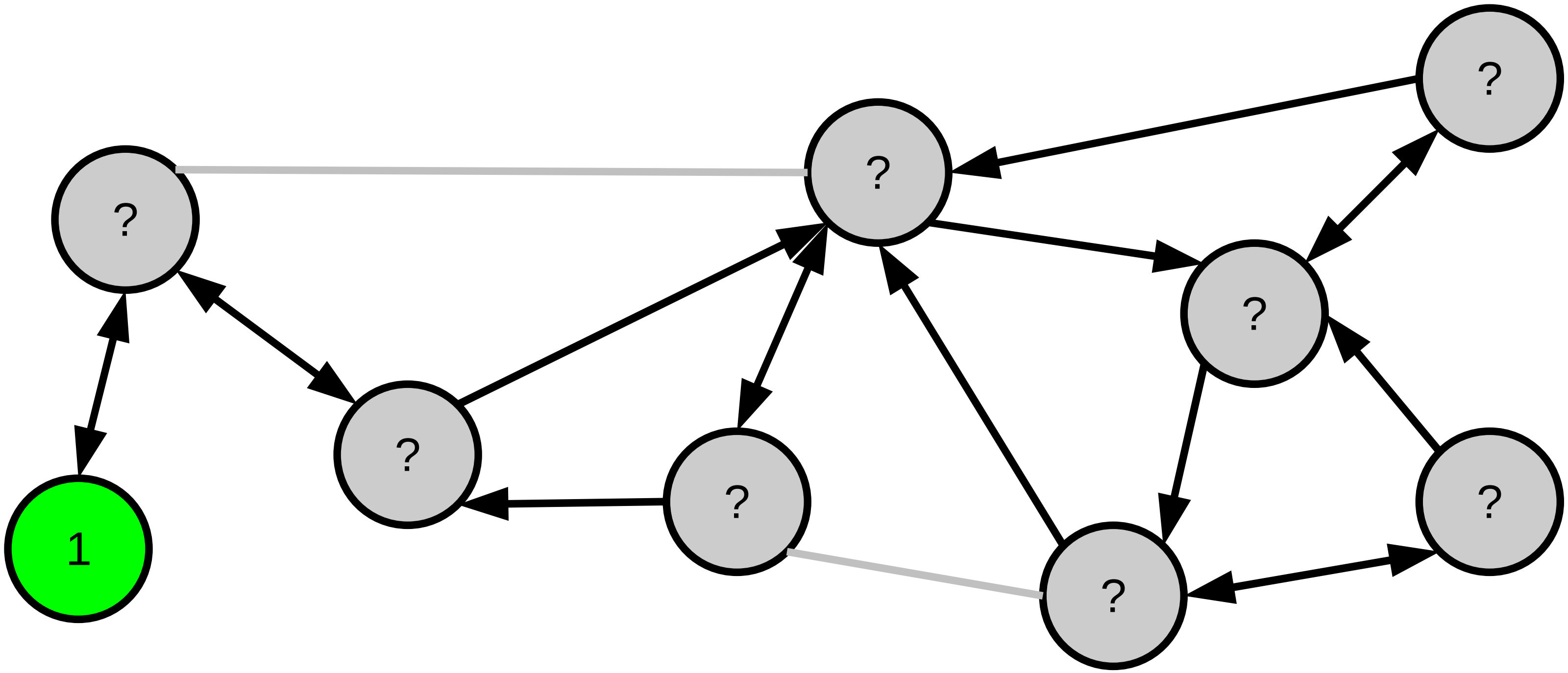
Average Degree: 3.5

Estimating Degrees



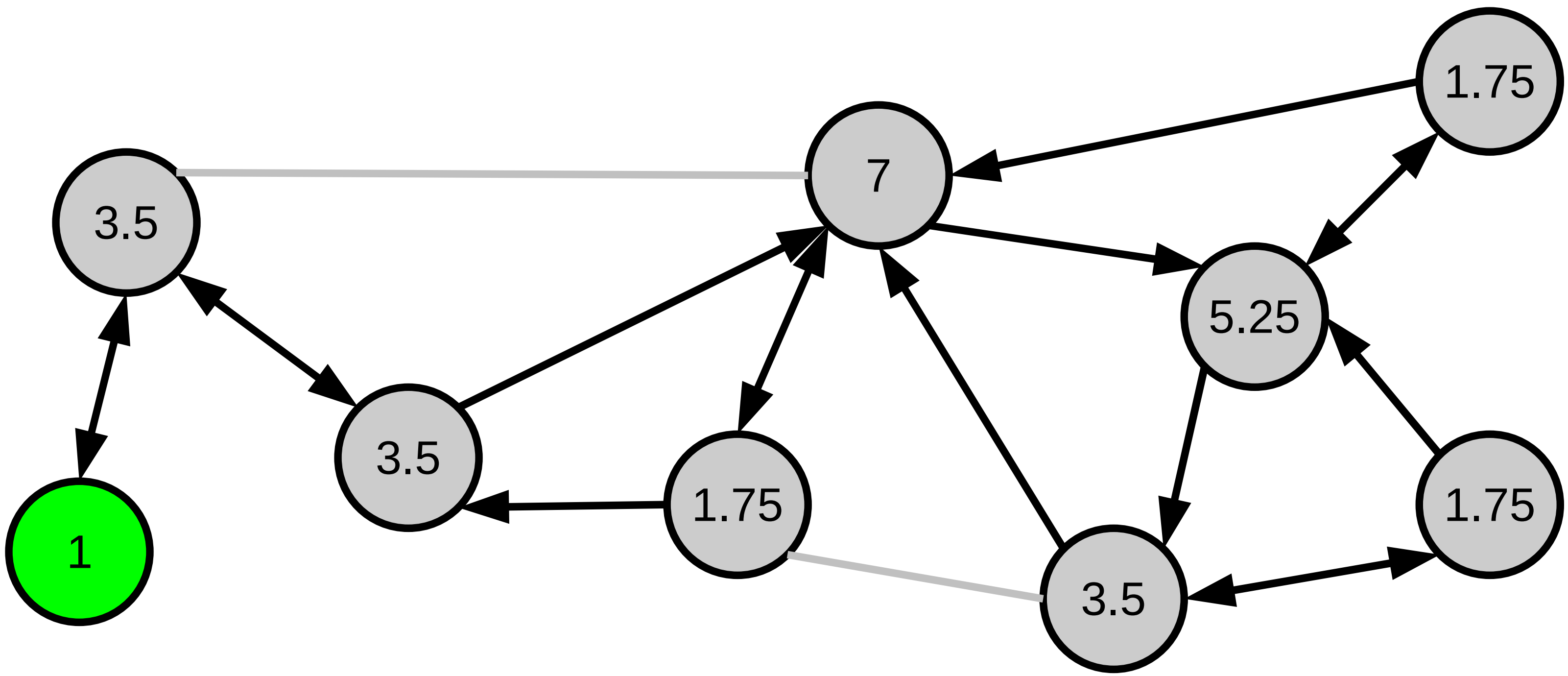
Sampled with *k*=2

Estimating Degrees



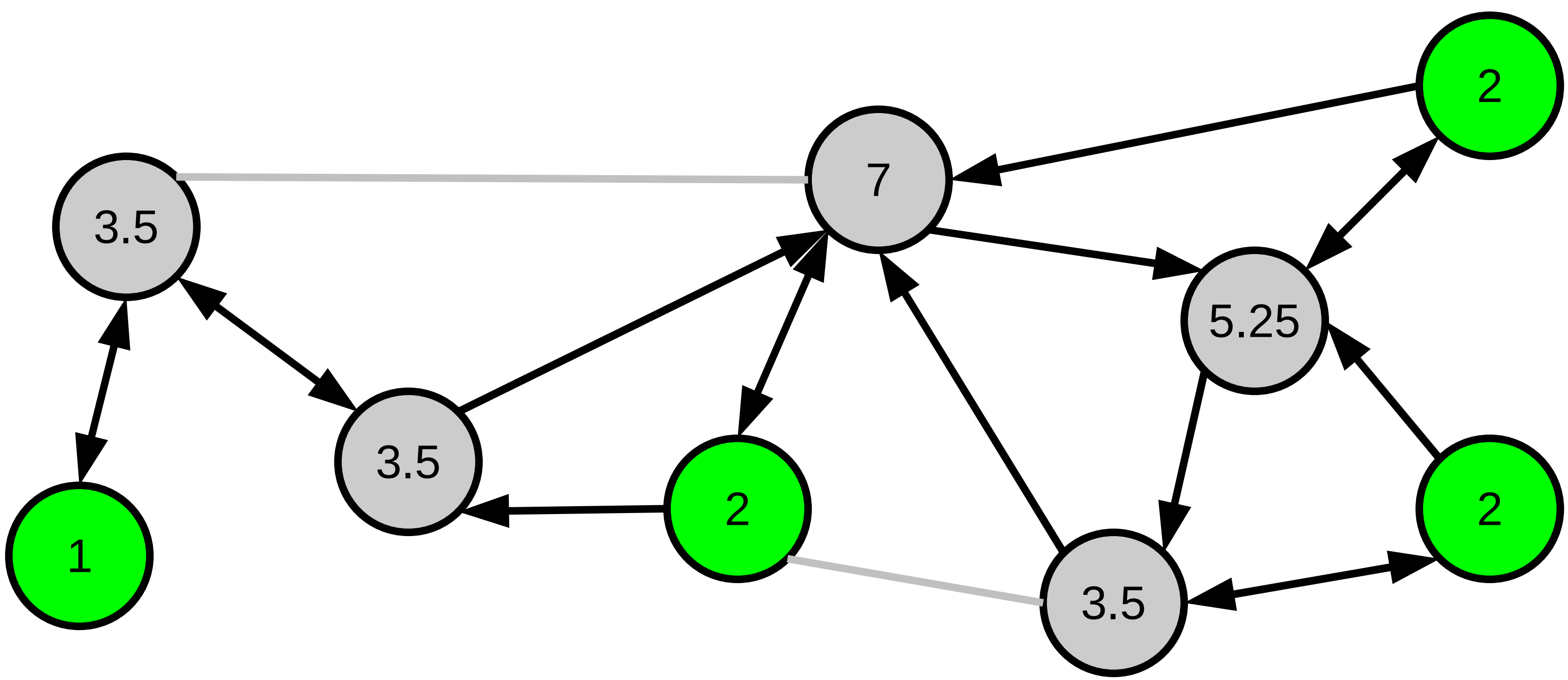
Degree known exactly for one node

Estimating Degrees



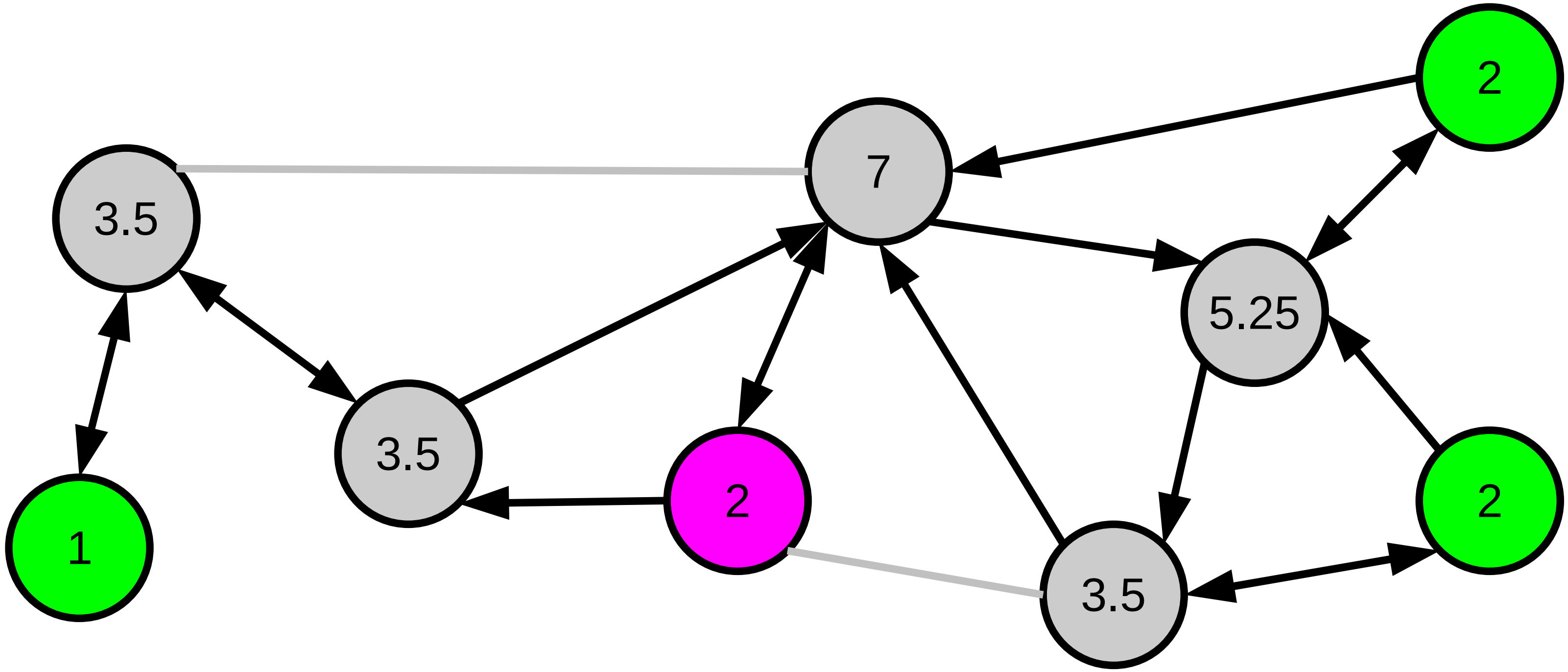
Naïve approach: Multiply in-degree by average degree / *k*

Estimating Degrees



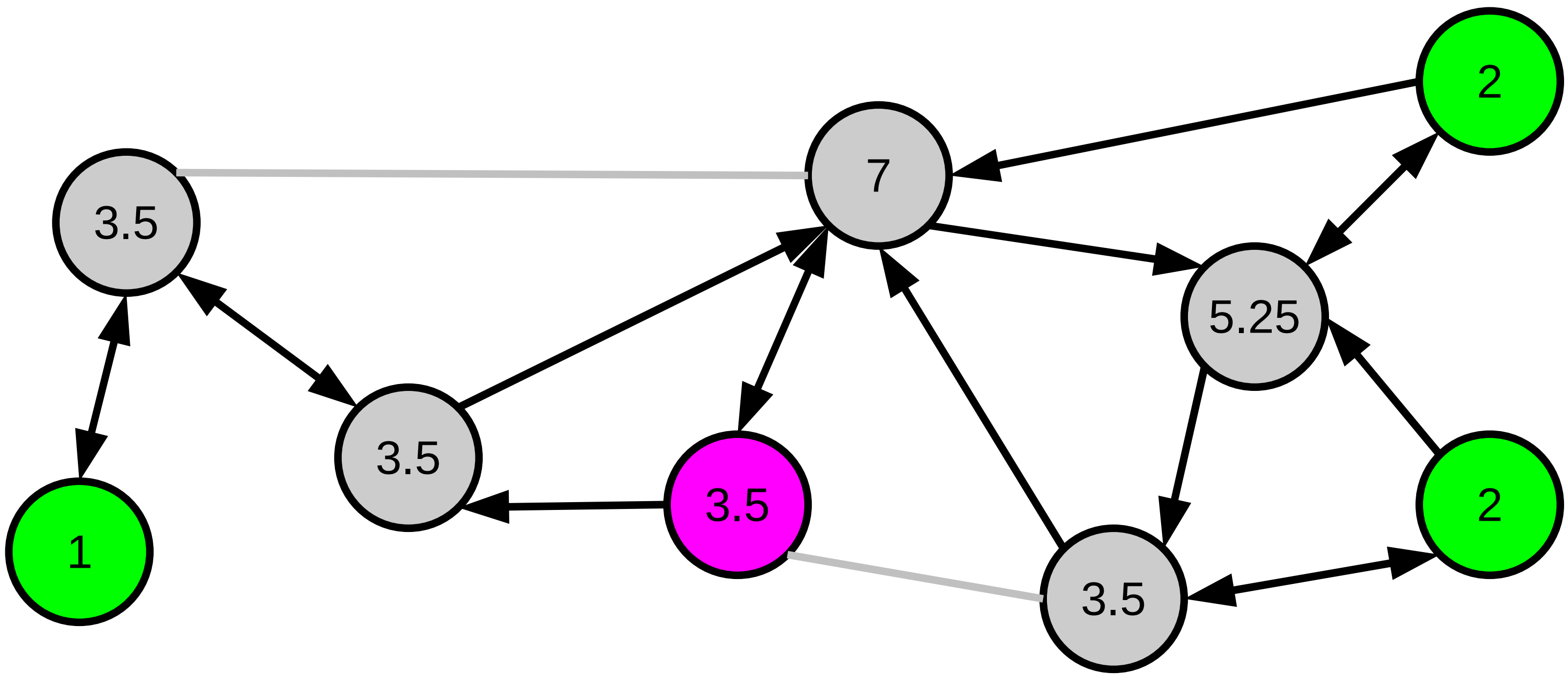
Raise estimates which are less than *k*

Estimating Degrees



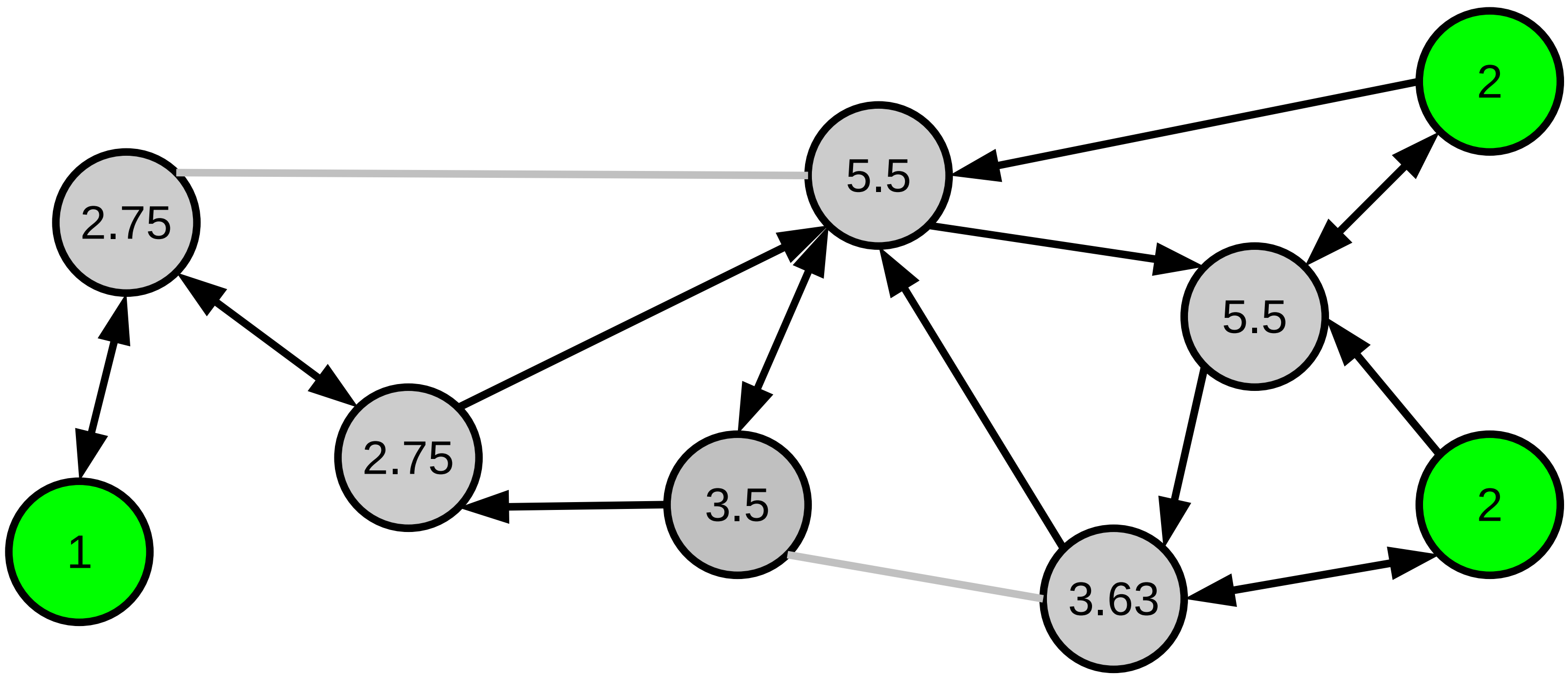
- Nodes with high-degree neighbors underestimated
- Iterative approach: Scale by current estimate / k in each step
- Basically, running PageRank

Estimating Degrees



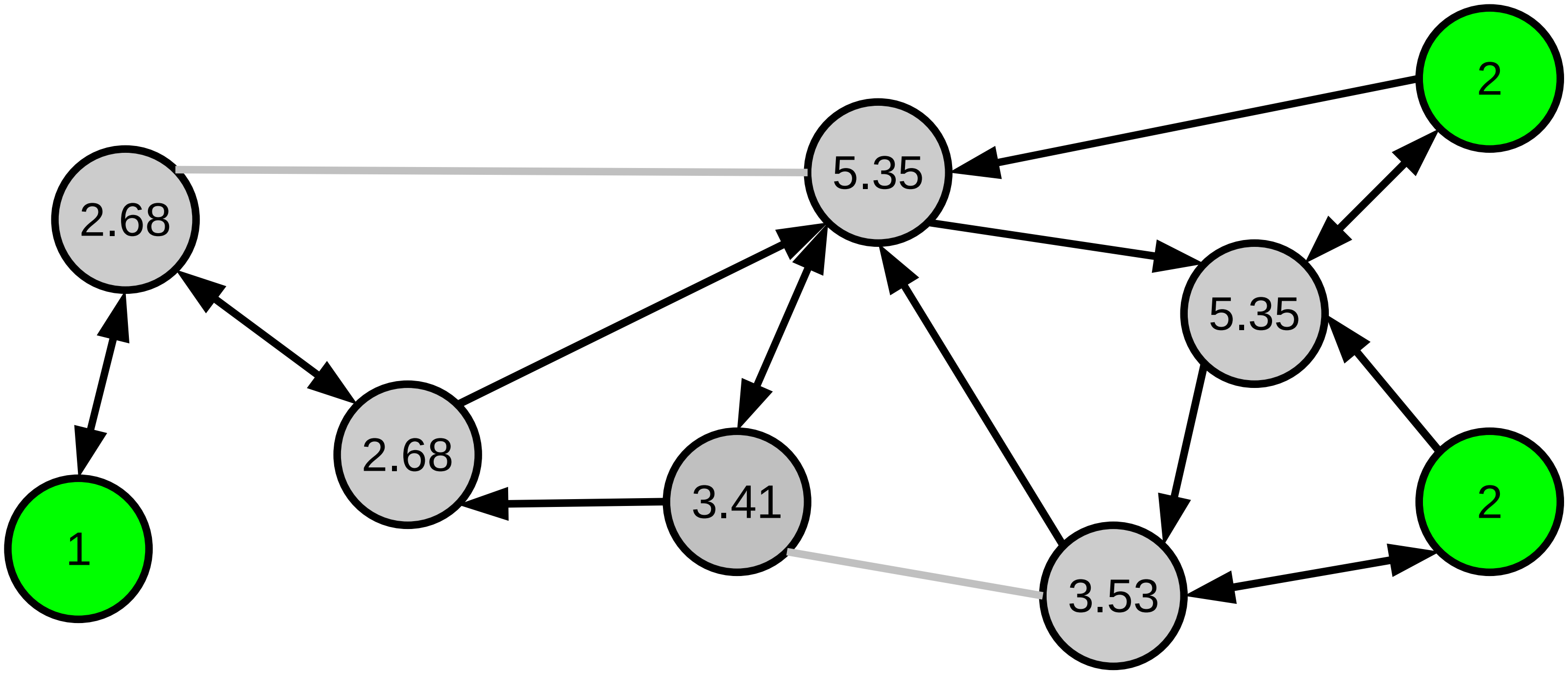
Refined estimate

Estimating Degrees



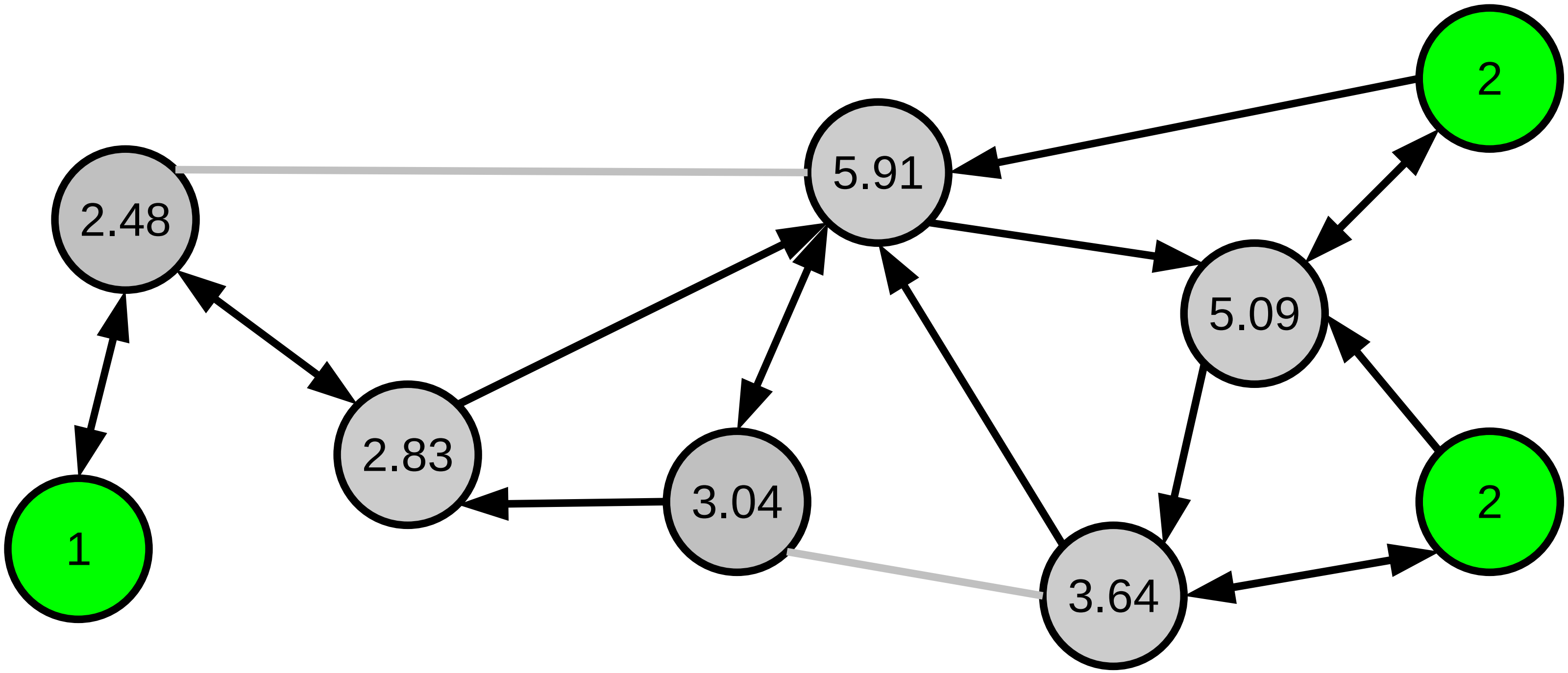
After 1 iteration

Estimating Degrees



Normalise to estimated total degree

Estimating Degrees

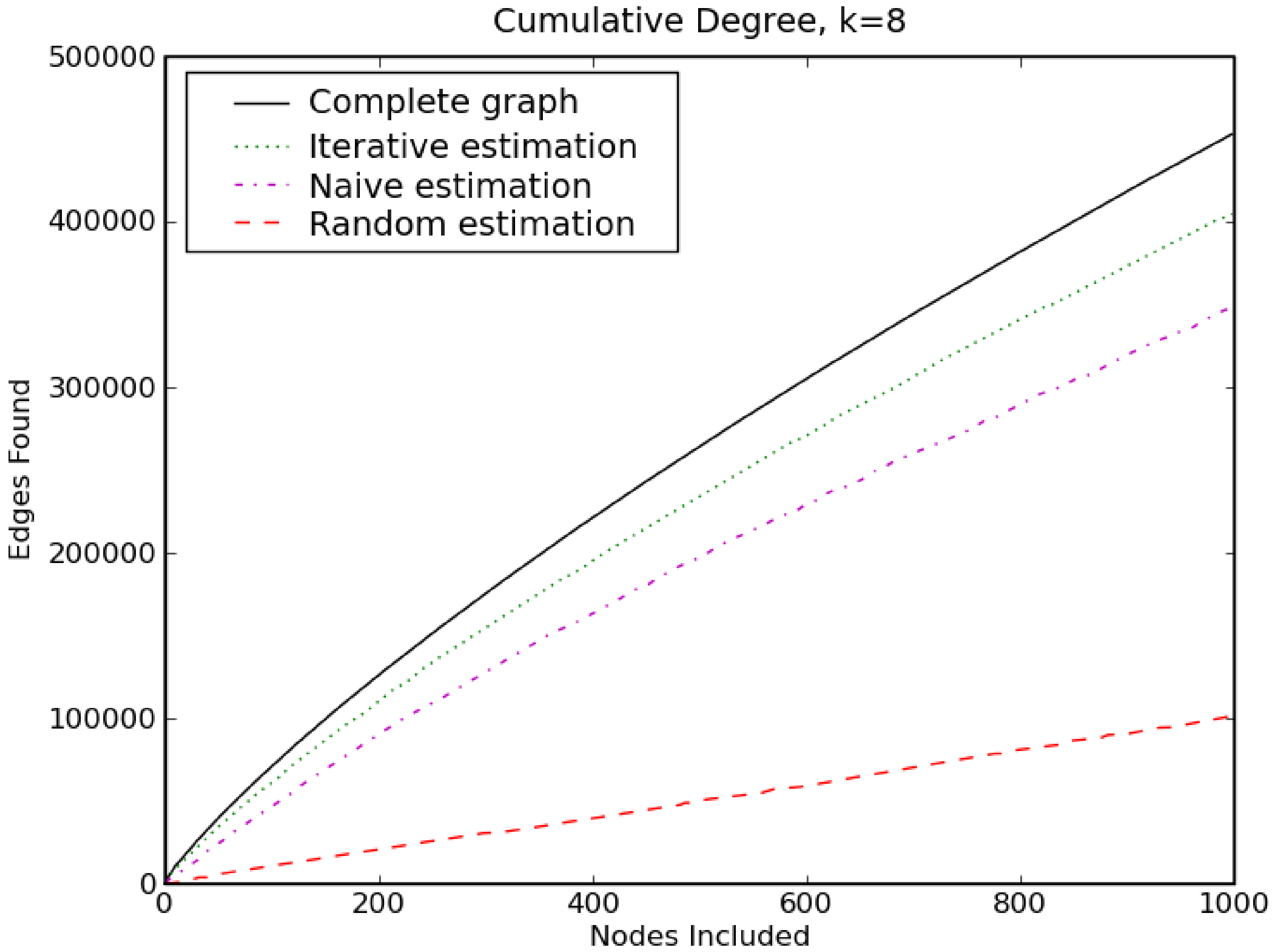


Convergence after n > 10 iterations

Estimating Degrees

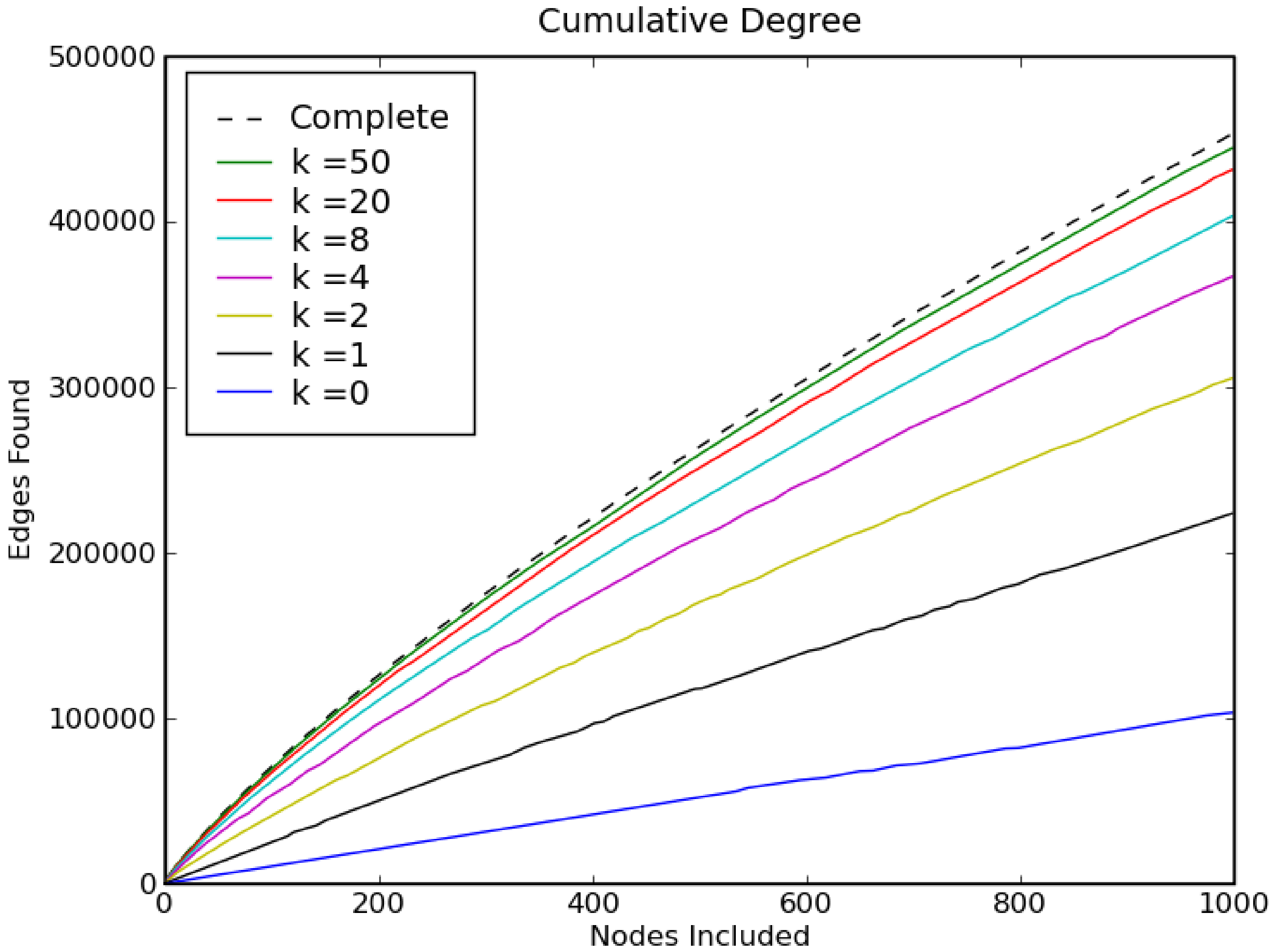
- Converges fast, typically after 10 iterations
- Absolute error is high—38% average
  - Reduced to 23% for nodes with $d \geq 50$
- Still accurately can pick high degree nodes

Estimating Degrees



D(x) = Aggregate degree of x highest-degree nodes

Estimating Degrees



D(x) = Aggregate degree of x highest-degree nodes

Dominating Sets

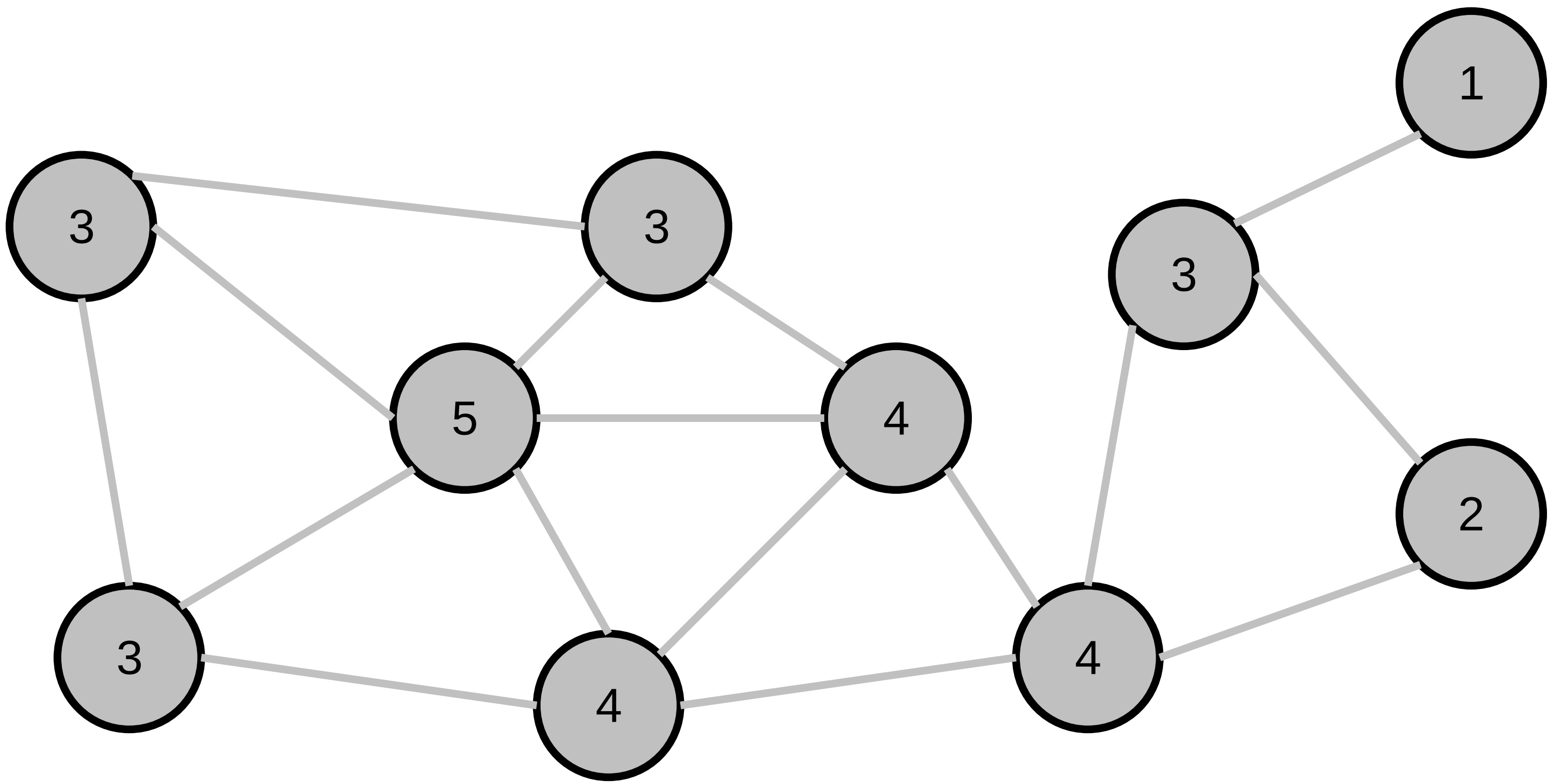
- Set of Nodes $D \subseteq V$ such that

  $D \cup \text{Neighbors}(D) = V$

- Set which allows viewing entire network

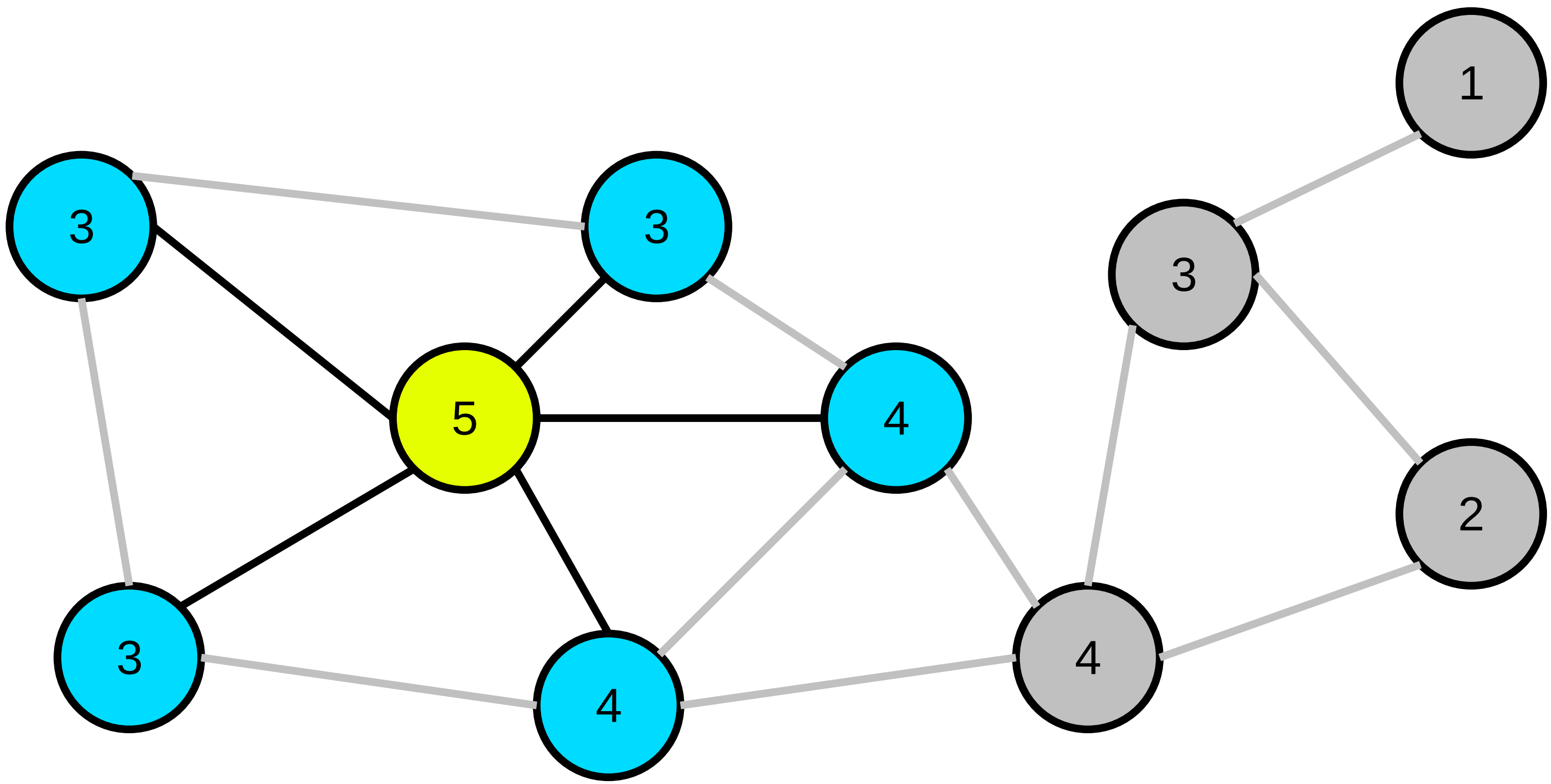
- Also useful for maximal marketing coverage

Dominating Sets



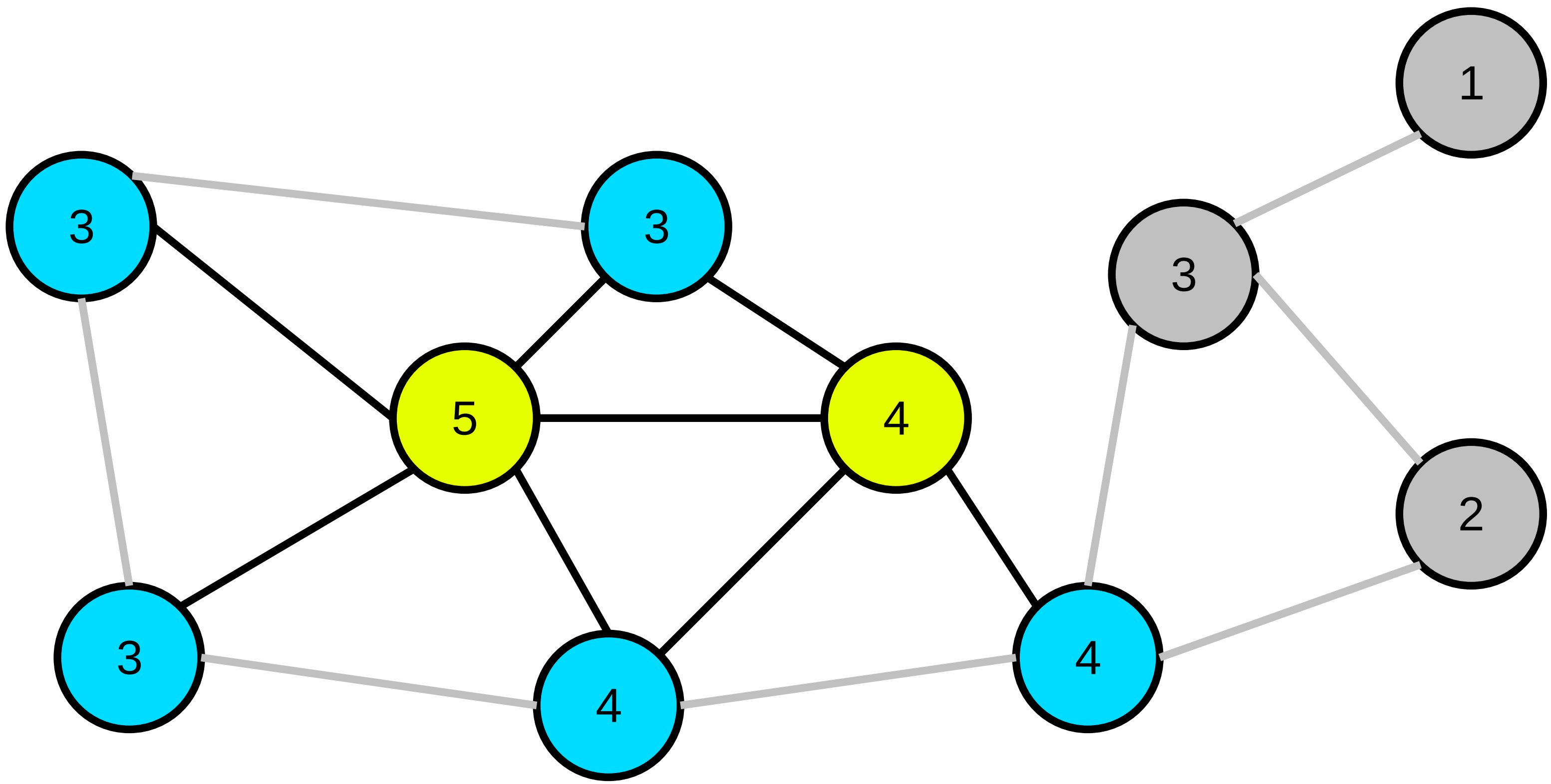
Trivial Algorithm: Select High-Degree Nodes in Order

Dominating Sets



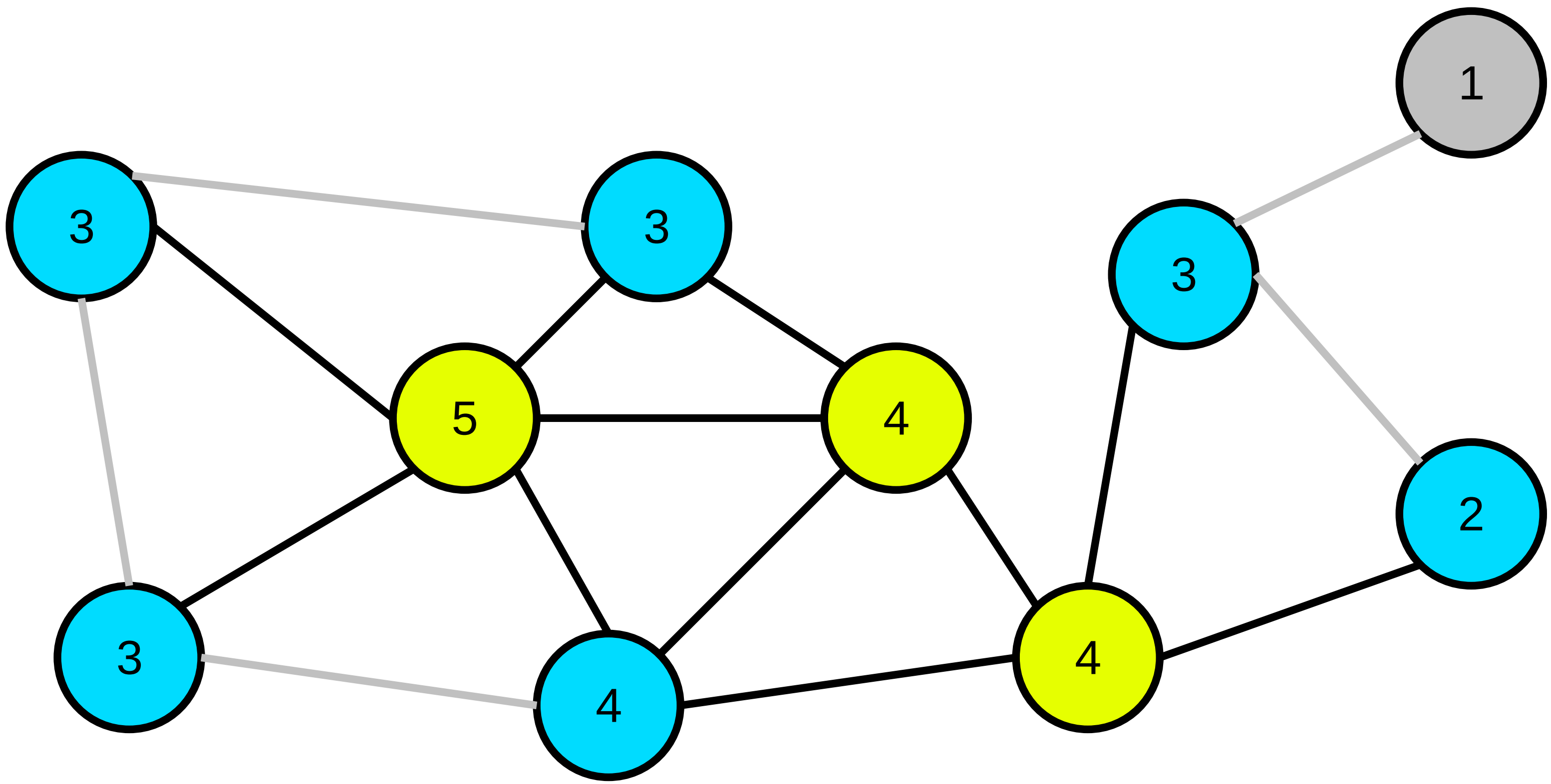
Trivial Algorithm: Select High-Degree Nodes in Order

Dominating Sets



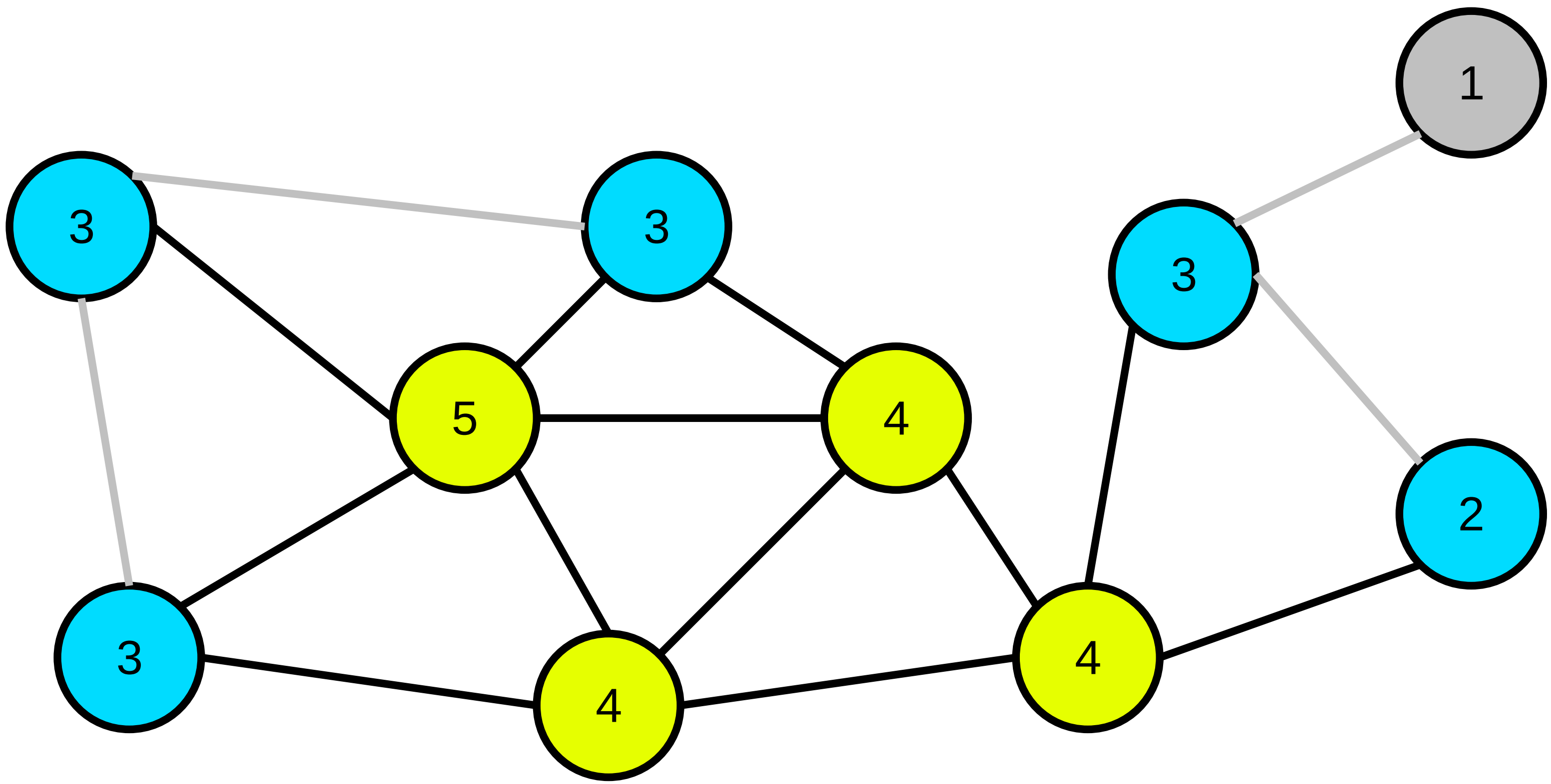
Trivial Algorithm: Select High-Degree Nodes in Order

Dominating Sets



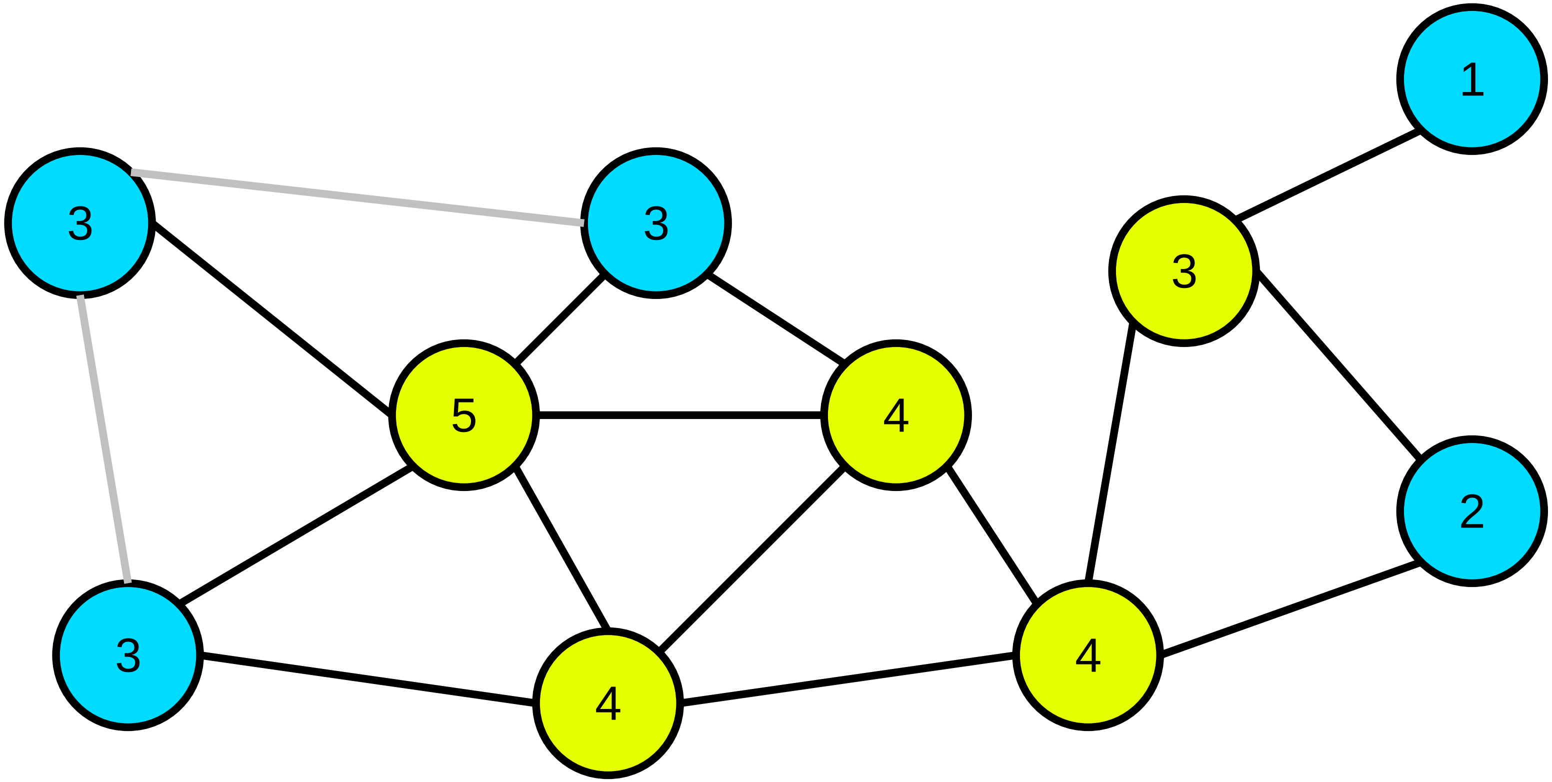
Trivial Algorithm: Select High-Degree Nodes in Order

Dominating Sets



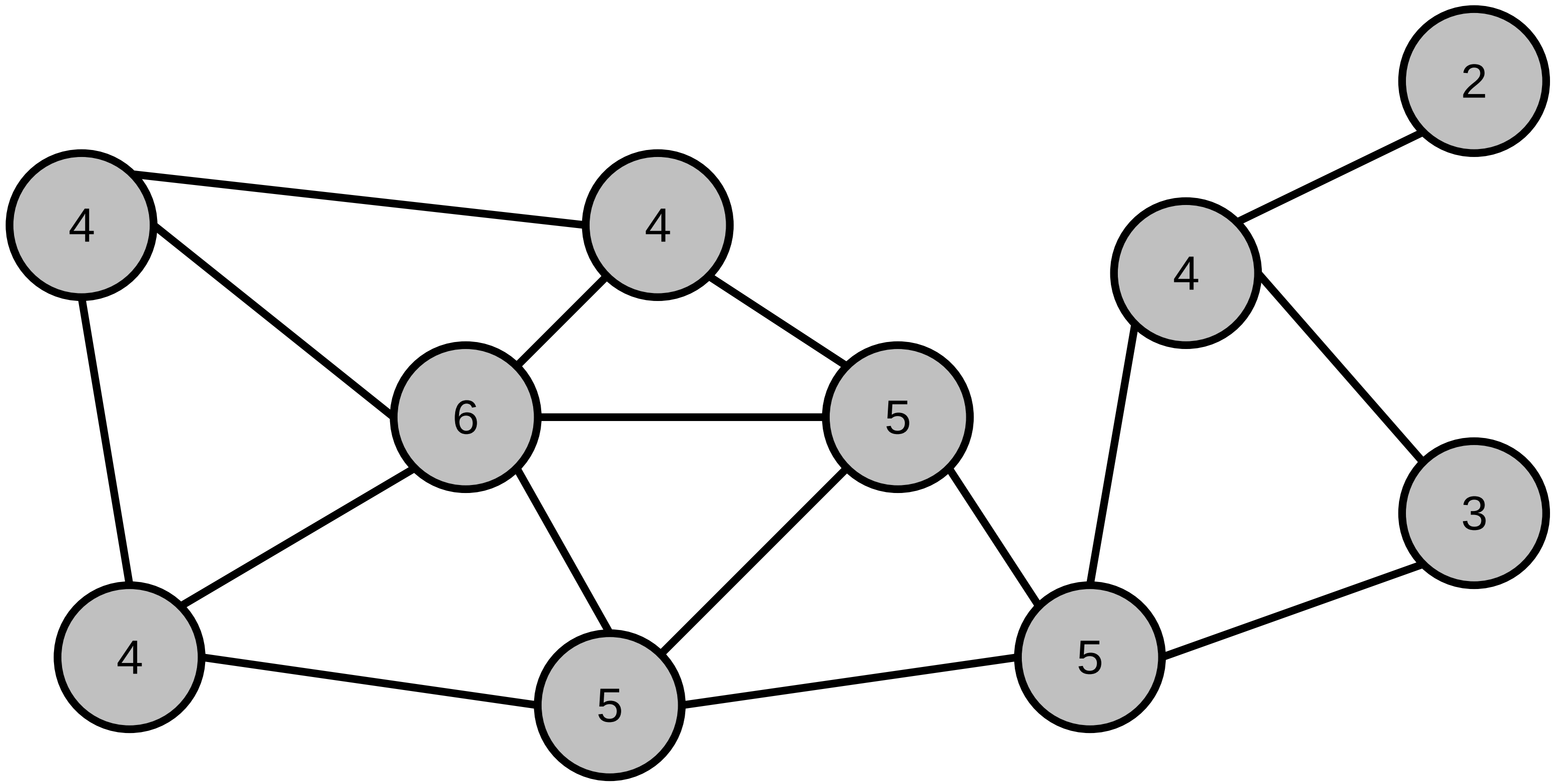
Trivial Algorithm: Select High-Degree Nodes in Order

Dominating Sets



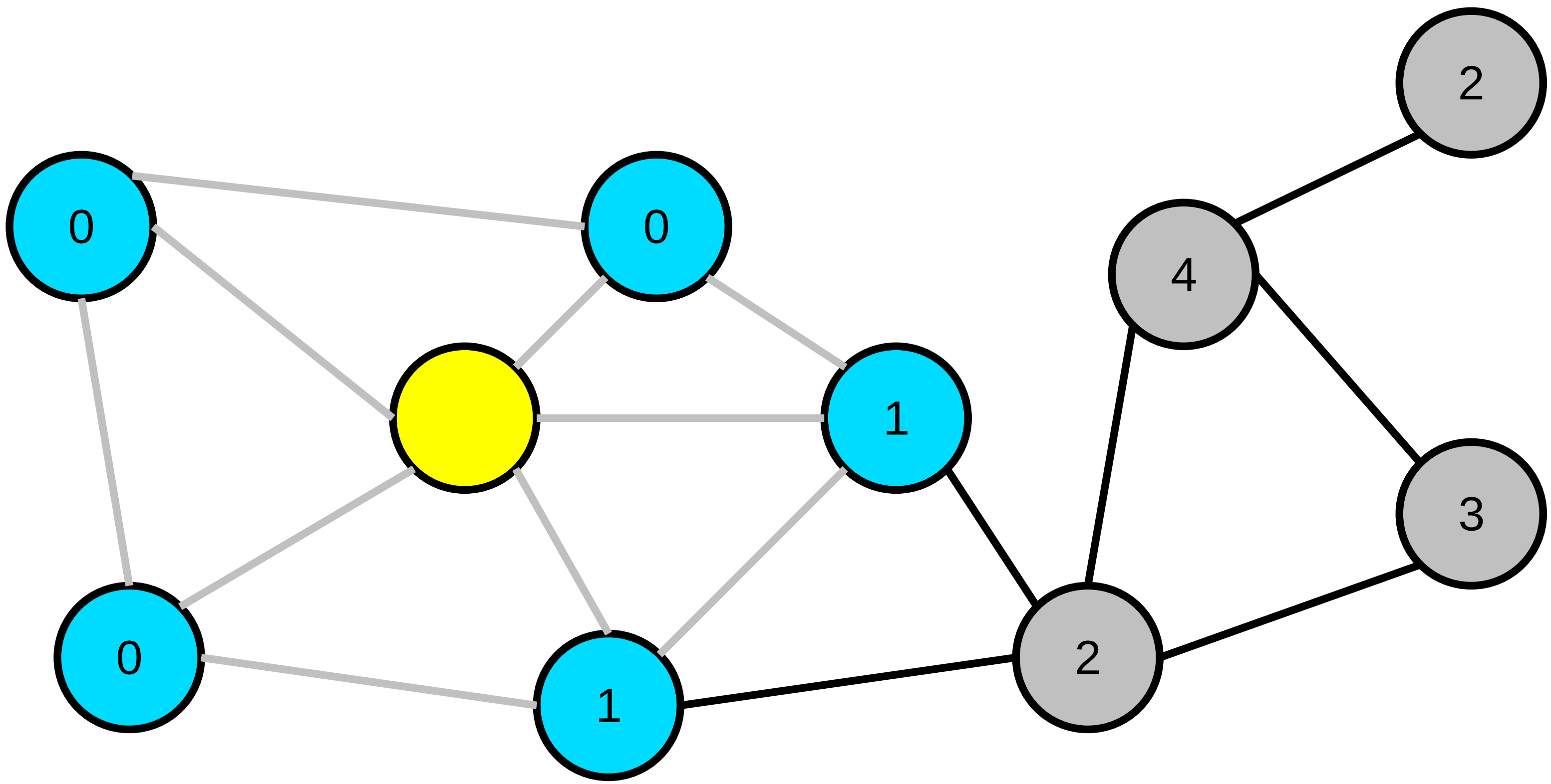
In fact, finding minimal dominating set is NP-complete

Dominating Sets



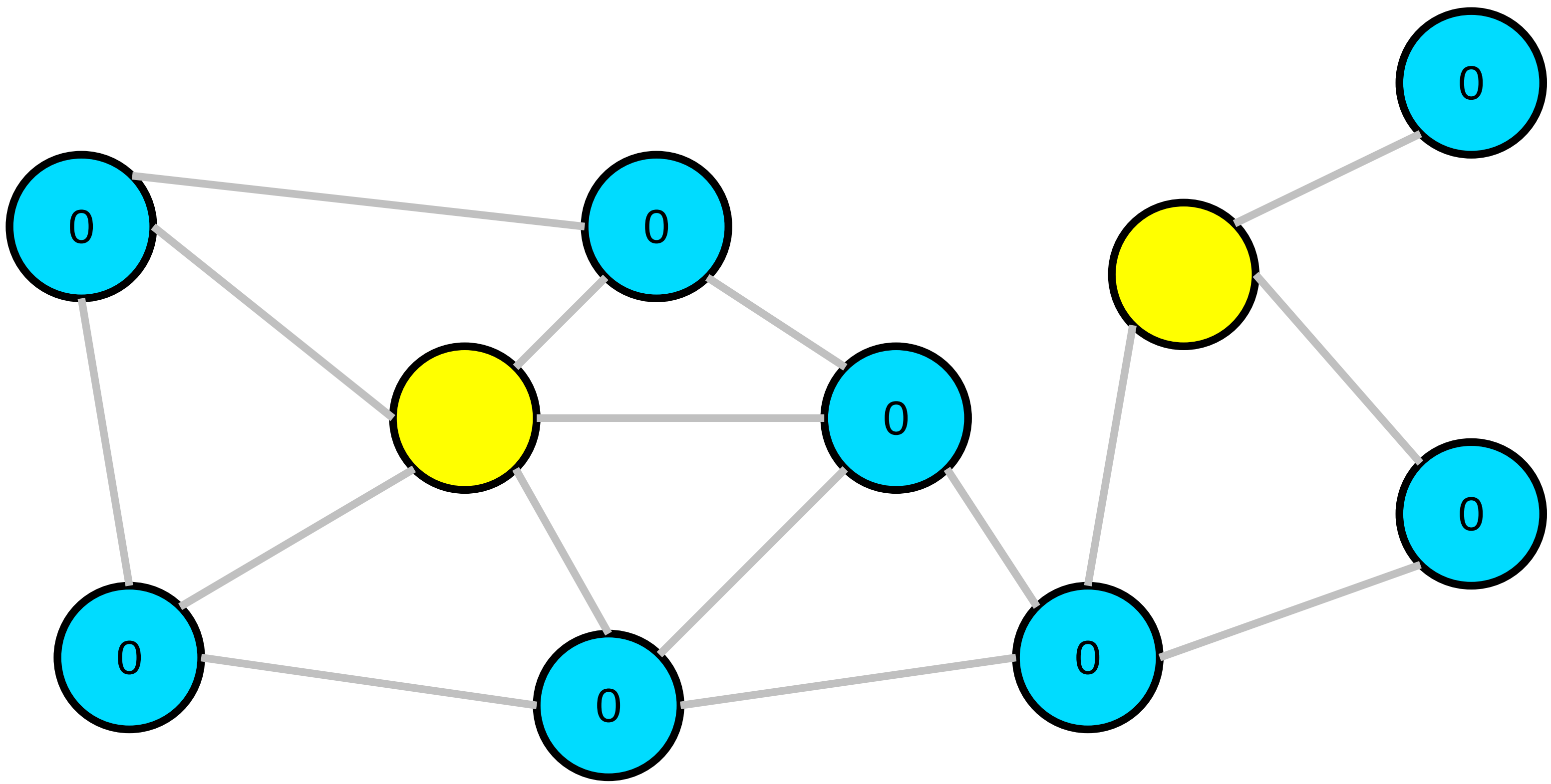
Greedy Algorithm: select node which adds maximal coverage

Dominating Sets



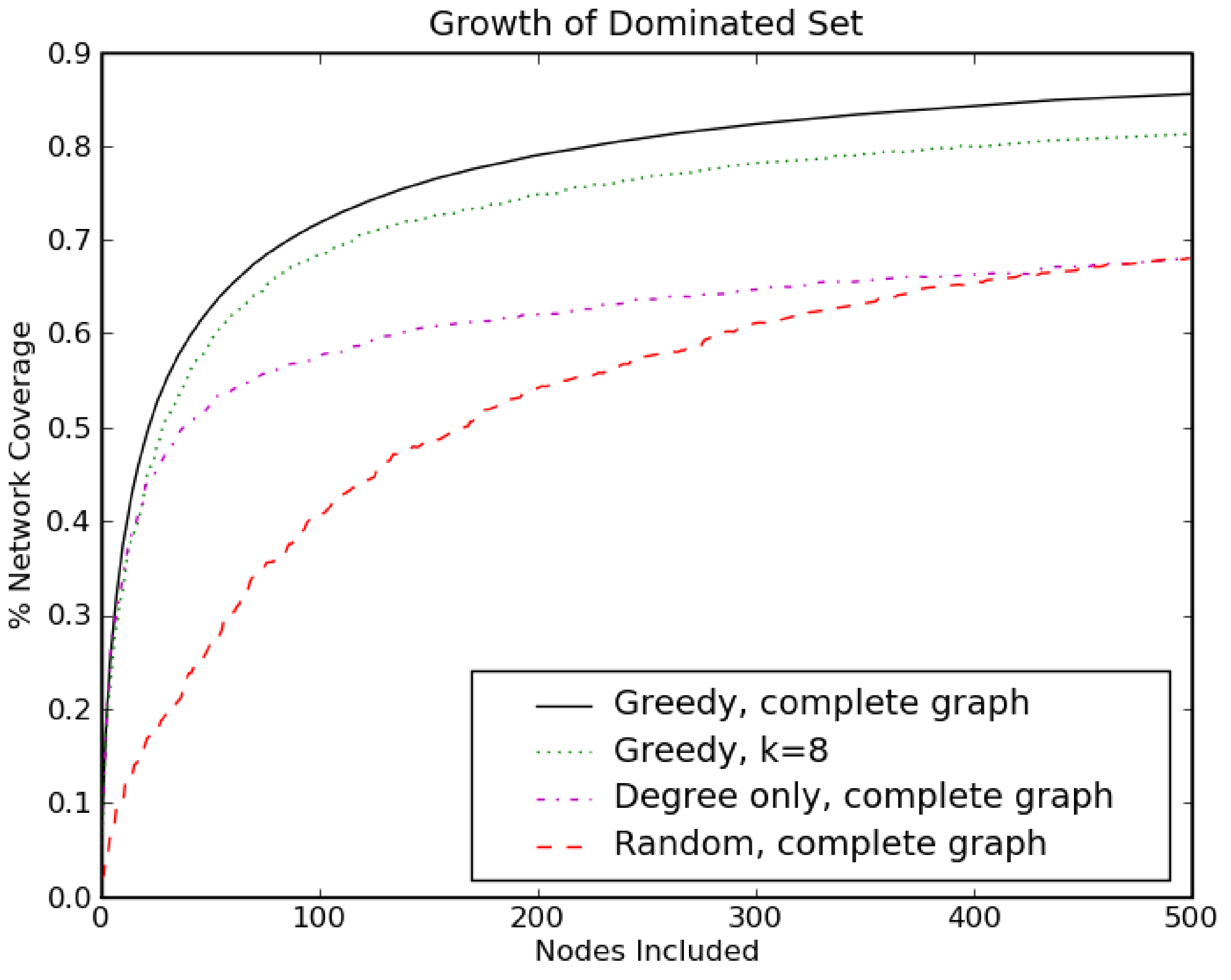
Greedy Algorithm: select node which adds maximal coverage

Dominating Sets



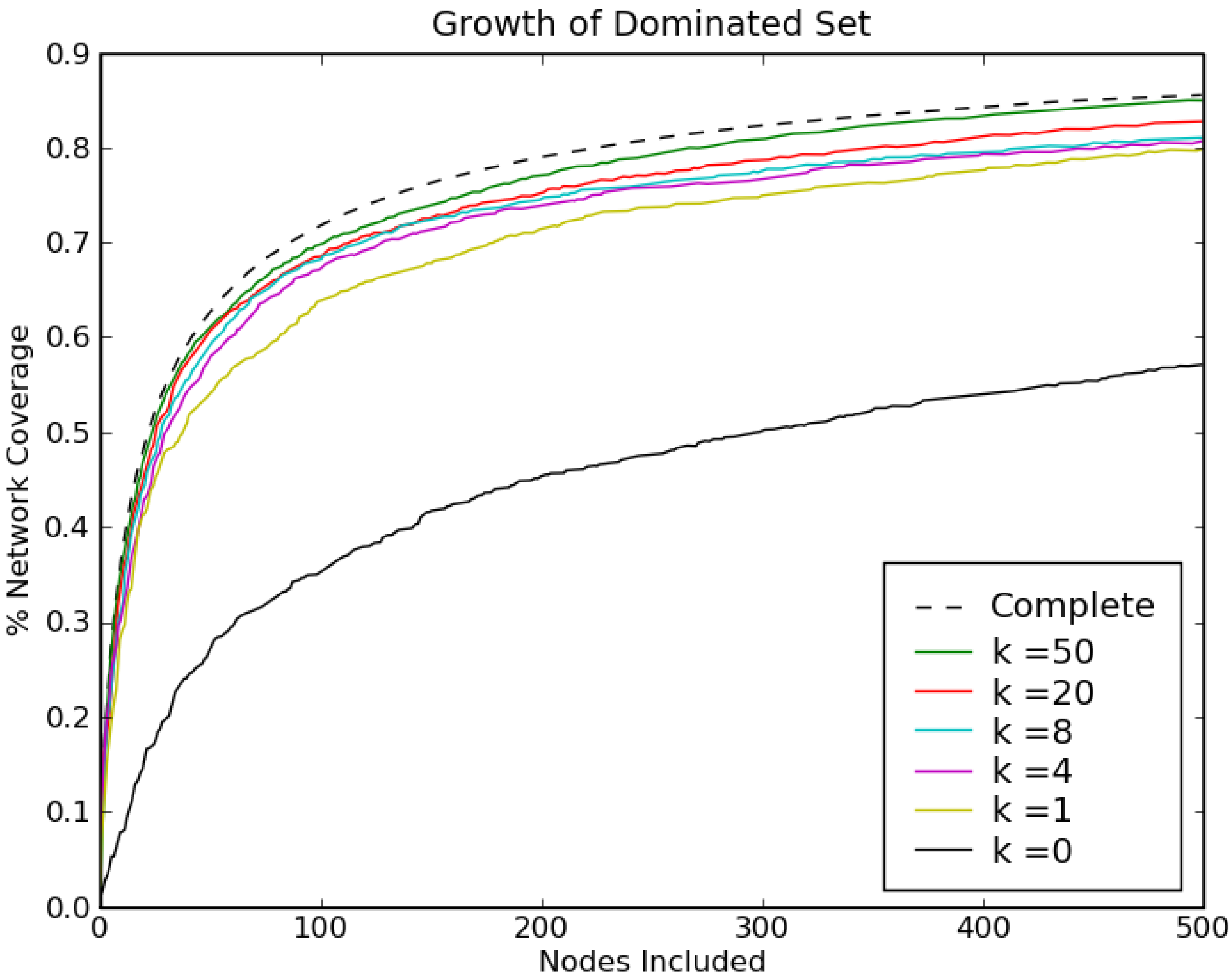
Shown to perform adequately in practice

Dominating Sets



Works well on sampled graph with no modification!

Dominating Sets



Surprising: Even $k = 1$ performs quite well

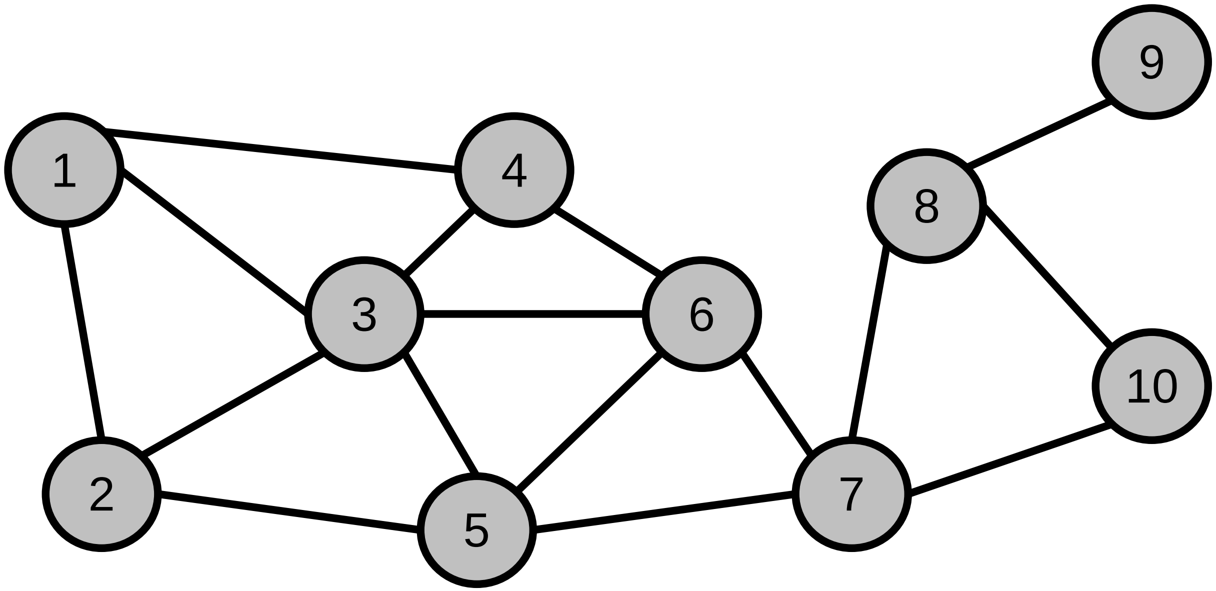
Shortest Paths

- Social networks shown to be “small world”
- Short paths should exist, even for large graphs
- Short paths can be used for social engineering

Floyd-Warshall Algorithm

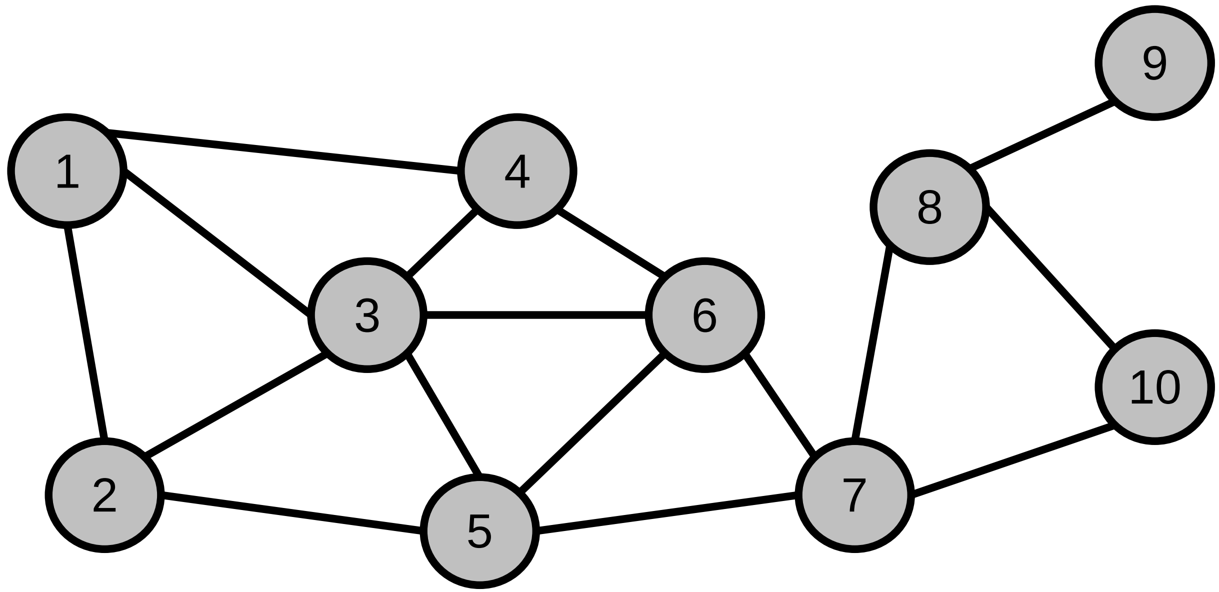
- Finds shortest distance between all (V,V) pairs
- Dynamic programming – $O(V^3)$ over V^2 nodes
- Think Dijkstra, but for all vertices

Floyd-Warshall Algorithm



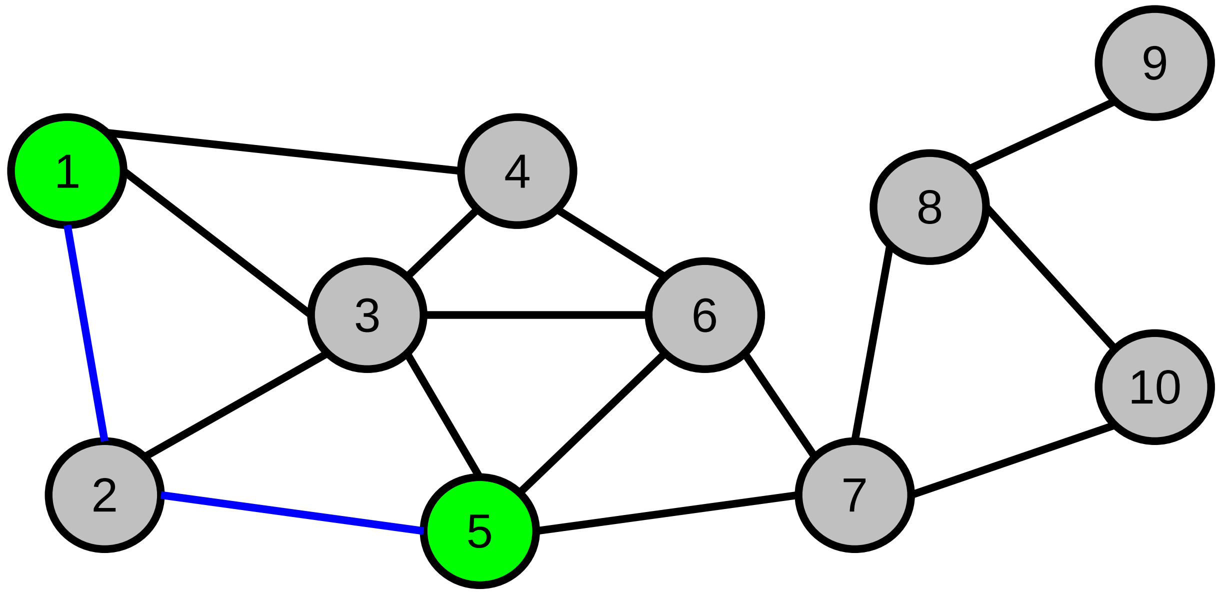
	1	2	3	4	5	6	7	8	9	10
1	0	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	0	∞	∞	∞	∞	∞	∞	∞	∞
3	∞	∞	0	∞	∞	∞	∞	∞	∞	∞
4	∞	∞	∞	0	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	0	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	∞	0	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	0	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	0	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	0	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	0

Floyd-Warshall Algorithm



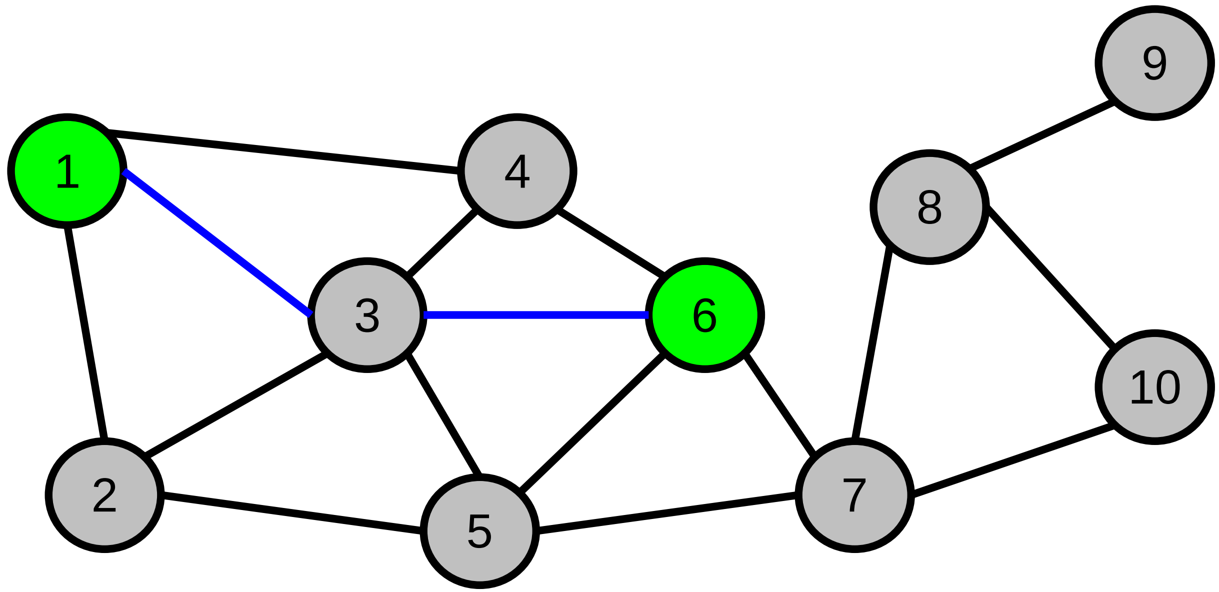
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0	1	1	1	∞	∞	∞	∞	∞	∞
2	1	0	1	∞	1	∞	∞	∞	∞	∞
3	1	1	0	1	1	1	∞	∞	∞	∞
4	1	∞	1	0	∞	1	∞	∞	∞	∞
5	∞	1	1	∞	0	1	1	∞	∞	∞
6	∞	∞	1	1	1	0	1	∞	∞	∞
7	∞	∞	∞	∞	1	1	0	1	∞	1
8	∞	∞	∞	∞	∞	∞	1	0	1	1
9	∞	∞	∞	∞	∞	∞	∞	1	0	∞
10	∞	∞	∞	∞	∞	∞	1	1	∞	0

Floyd-Warshall Algorithm



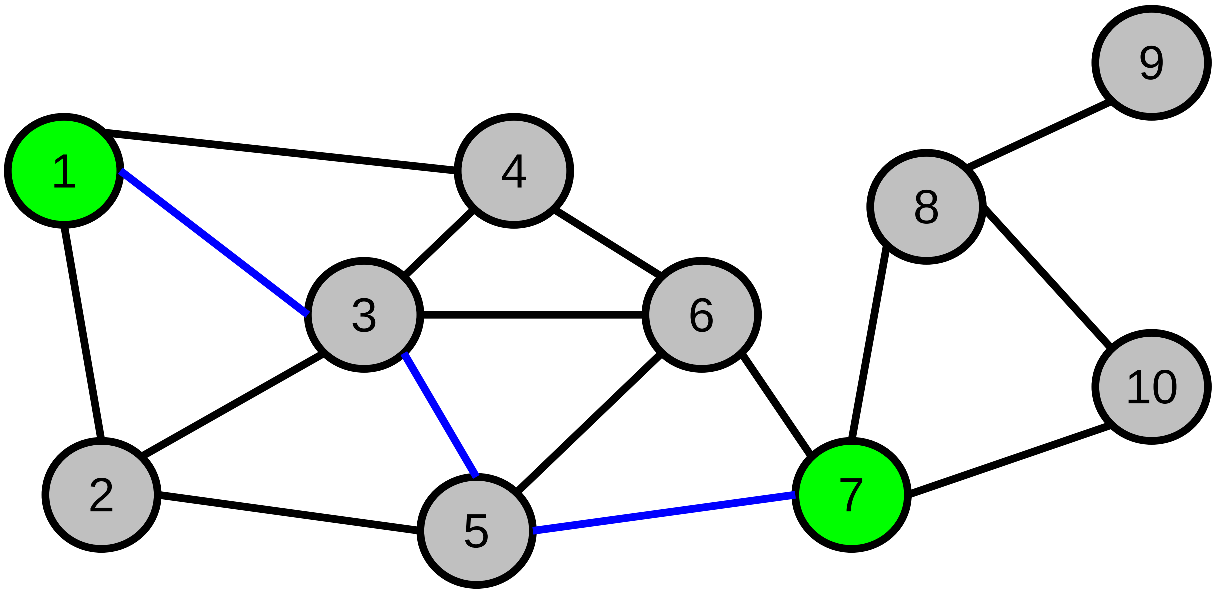
	1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	2	∞	∞	∞	∞	∞
2	1	0	1	∞	1	∞	∞	∞	∞	∞
3	1	1	0	1	1	1	∞	∞	∞	∞
4	1	∞	1	0	∞	1	∞	∞	∞	∞
5	2	1	1	∞	0	1	1	∞	∞	∞
6	∞	∞	1	1	1	0	1	∞	∞	∞
7	∞	∞	∞	∞	1	1	0	1	∞	1
8	∞	∞	∞	∞	∞	∞	1	0	1	1
9	∞	∞	∞	∞	∞	∞	∞	1	0	∞
10	∞	∞	∞	∞	∞	∞	1	1	∞	0

Floyd-Warshall Algorithm



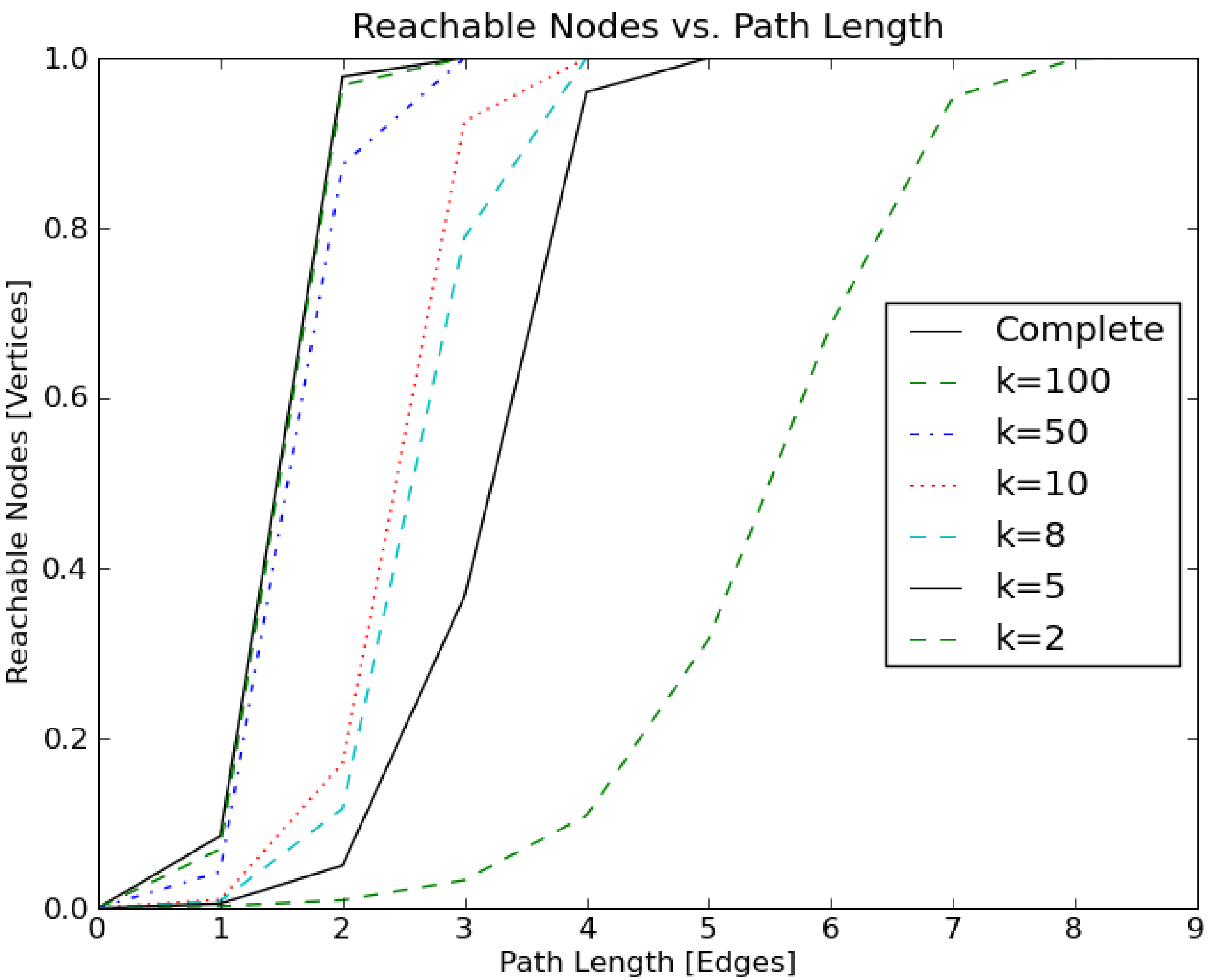
	1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	2	2	∞	∞	∞	∞
2	1	0	1	∞	1	∞	∞	∞	∞	∞
3	1	1	0	1	1	1	∞	∞	∞	∞
4	1	∞	1	0	∞	1	∞	∞	∞	∞
5	2	1	1	∞	0	1	1	∞	∞	∞
6	2	∞	1	1	1	0	1	∞	∞	∞
7	∞	∞	∞	∞	1	1	0	1	∞	1
8	∞	∞	∞	∞	∞	∞	1	0	1	1
9	∞	∞	∞	∞	∞	∞	∞	1	0	∞
10	∞	∞	∞	∞	∞	∞	1	1	∞	0

Floyd-Warshall Algorithm



	1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	2	2	3	∞	∞	∞
2	1	0	1	∞	1	∞	∞	∞	∞	∞
3	1	1	0	1	1	1	∞	∞	∞	∞
4	1	∞	1	0	∞	1	∞	∞	∞	∞
5	2	1	1	∞	0	1	1	∞	∞	∞
6	2	∞	1	1	1	0	1	∞	∞	∞
7	3	∞	∞	∞	1	1	0	1	∞	1
8	∞	∞	∞	∞	∞	∞	1	0	1	1
9	∞	∞	∞	∞	∞	∞	∞	1	0	∞
10	∞	∞	∞	∞	∞	∞	1	1	∞	0

Shortest Paths



[1] For k=8, paths are ~1 hop longer

[2] All nodes reachable with k=2

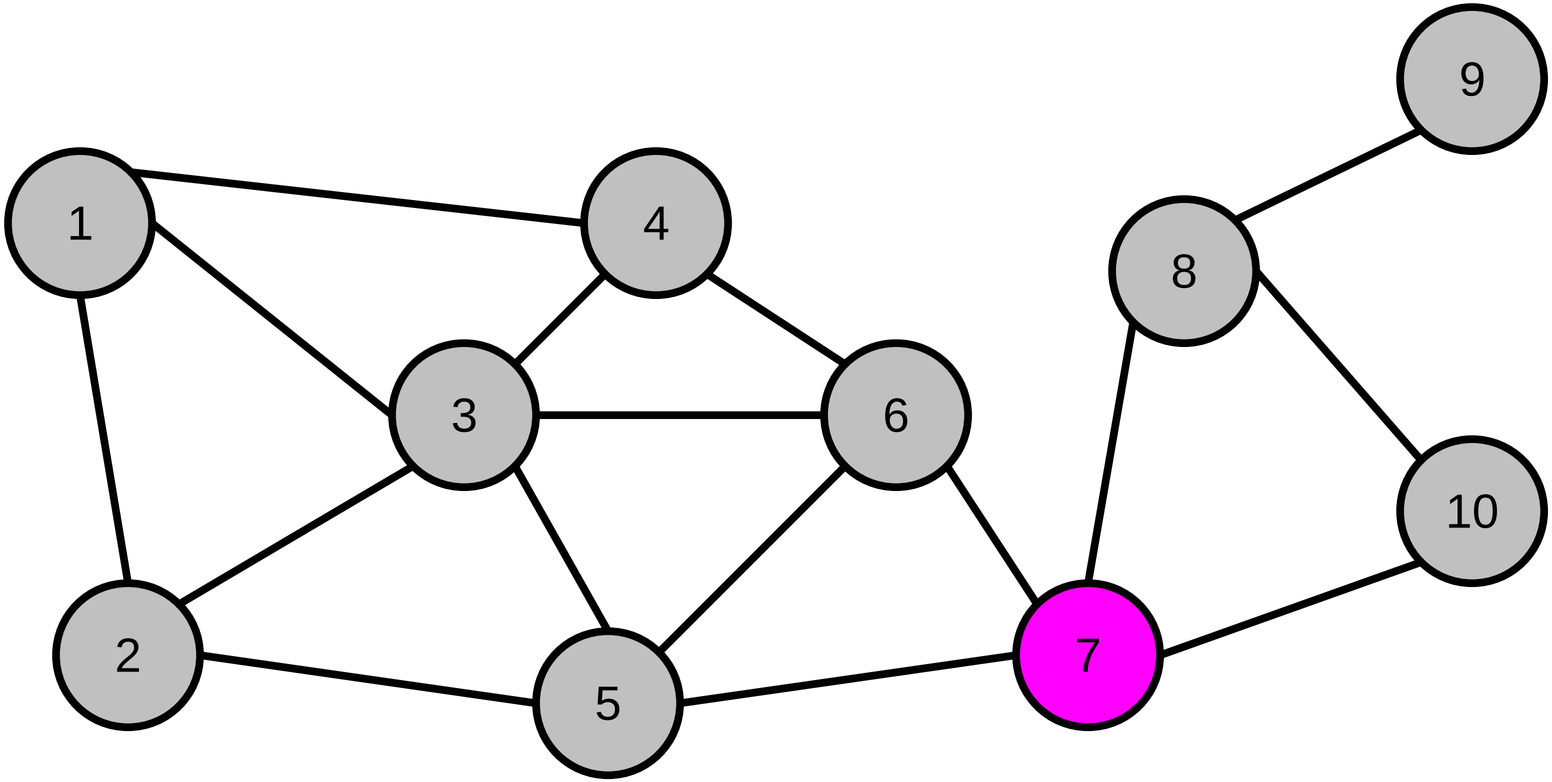
Centrality

- A measure of a node's importance
- *Betweenness centrality*:

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

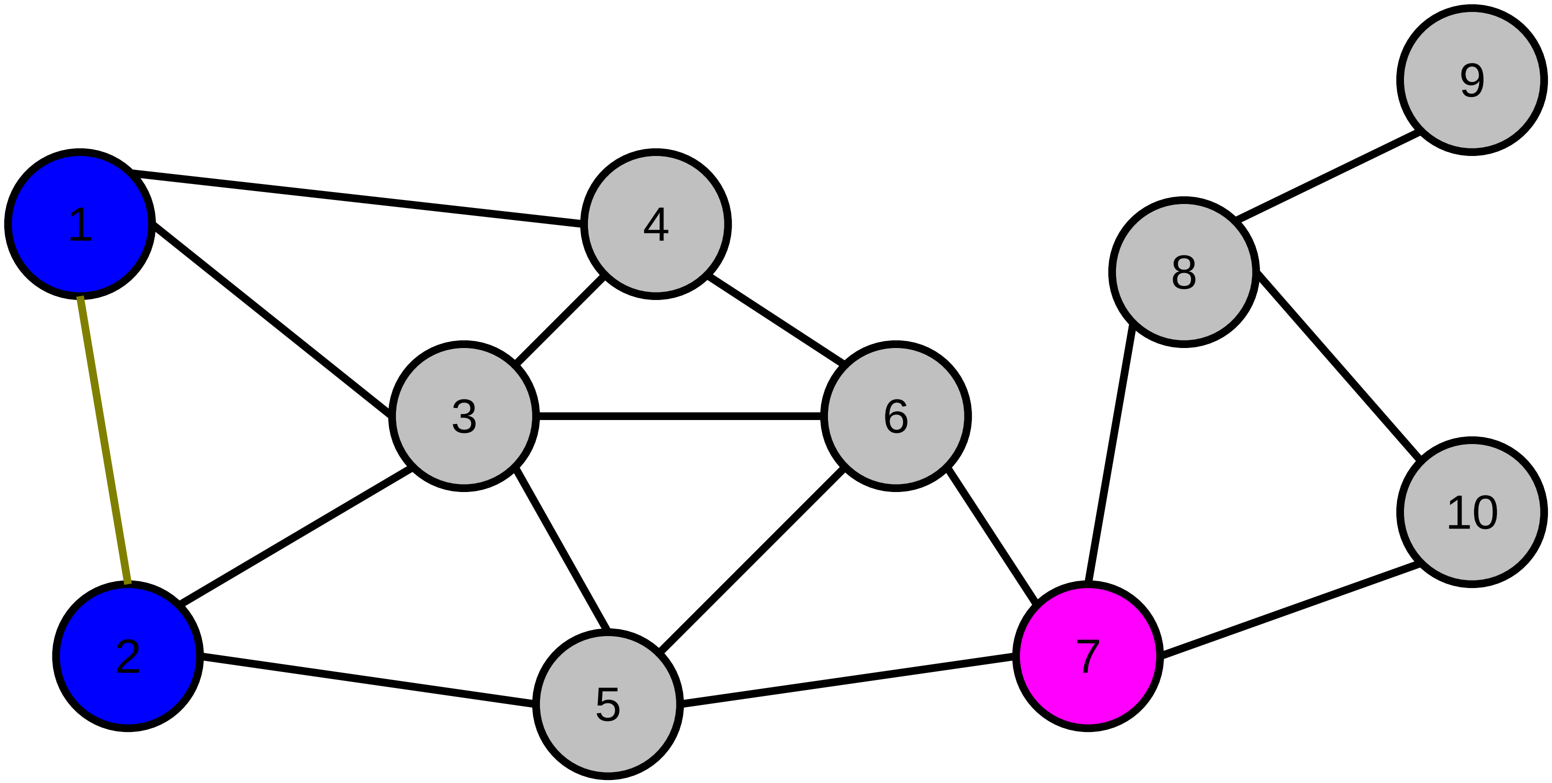
  - Measures the shortest paths in the graph that a particular vertex is part of

Centrality



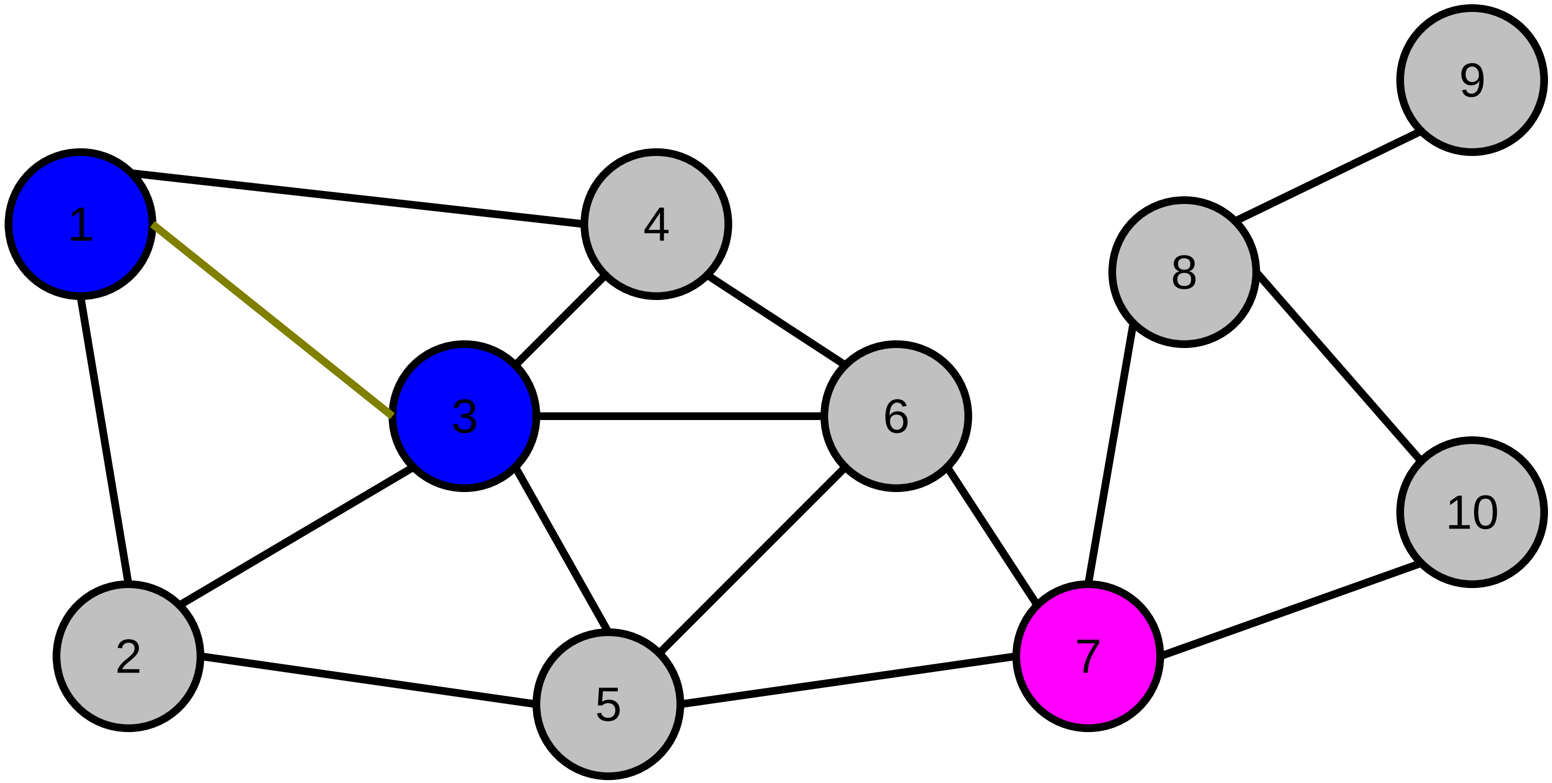
$C_B(v_7) =$

Centrality



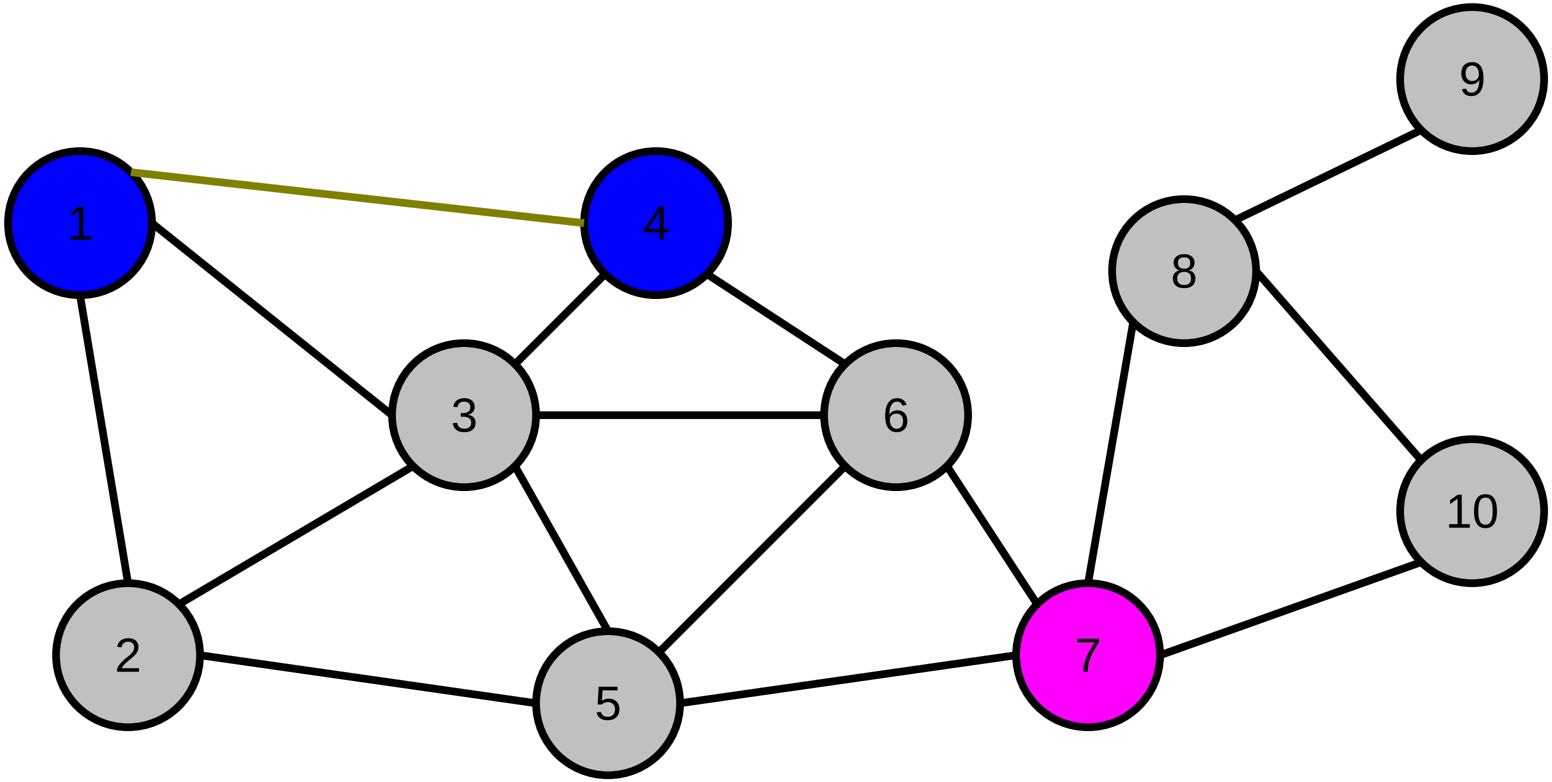
$$C_B(v_7) = \frac{0}{1} +$$

Centrality



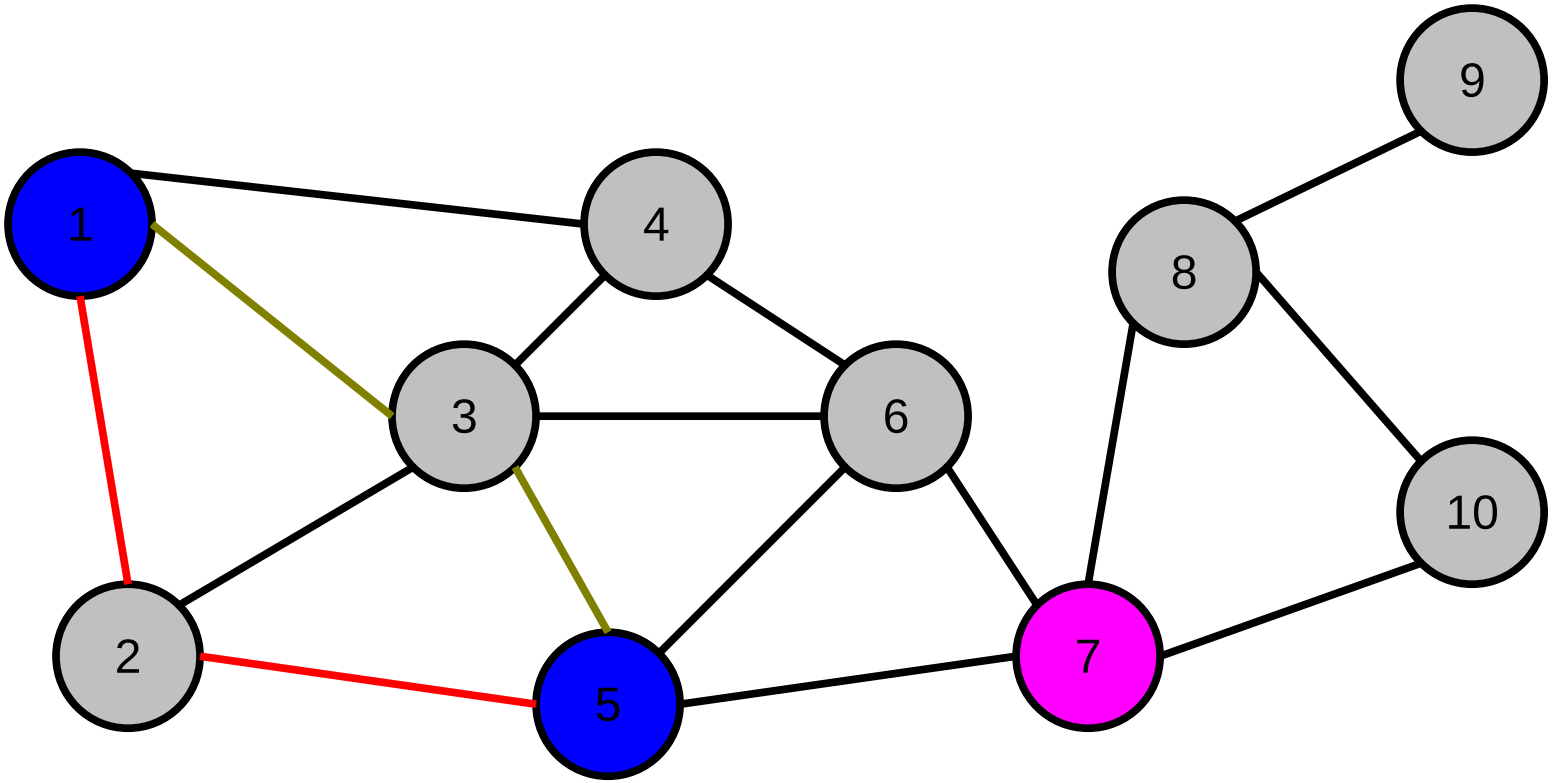
$$C_B(v_7) = \frac{0}{1} + \frac{0}{1} +$$

Centrality



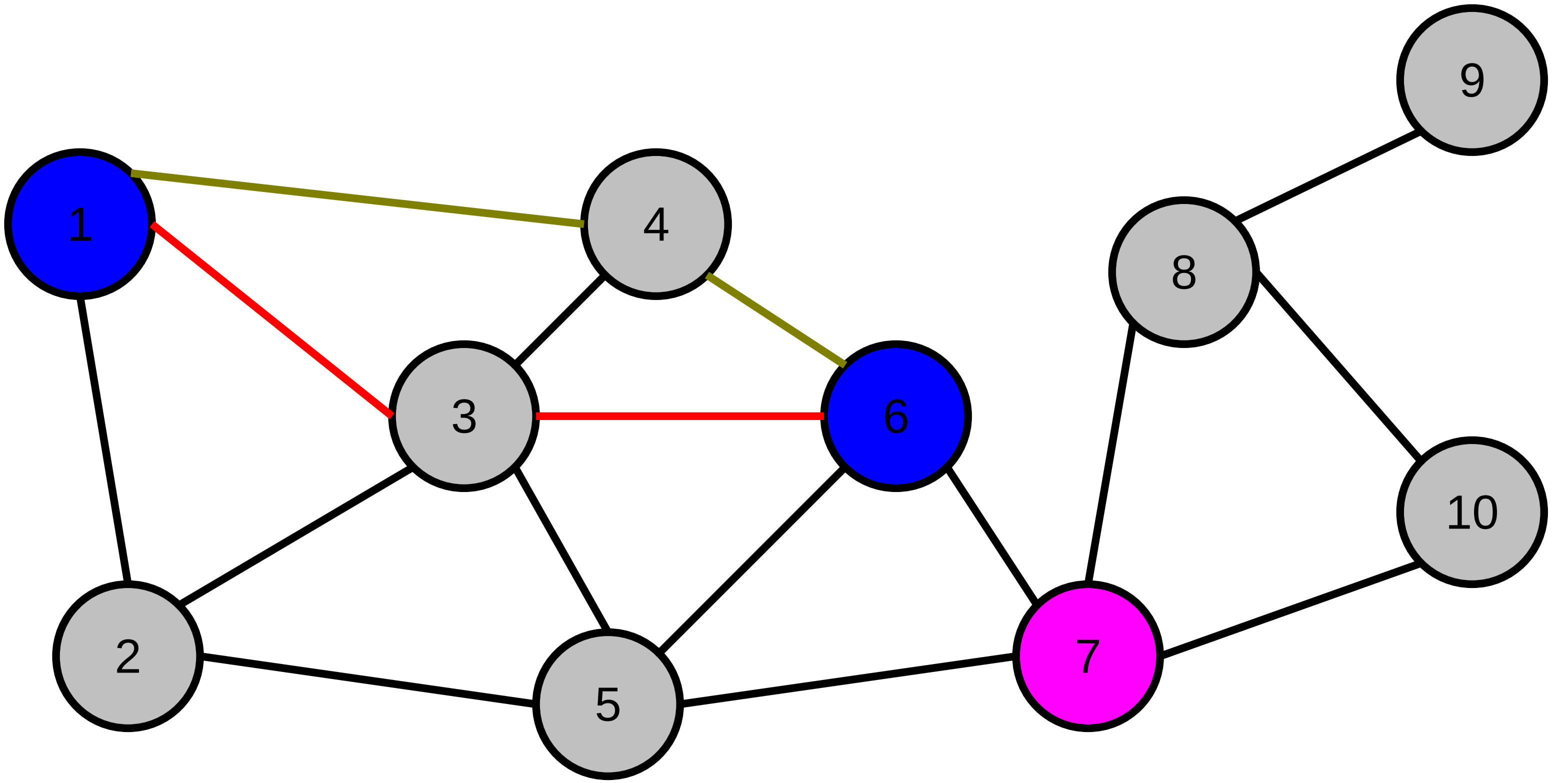
$$C_B(v_7) = \frac{0}{1} + \frac{0}{1} + \frac{0}{1} +$$

Centrality



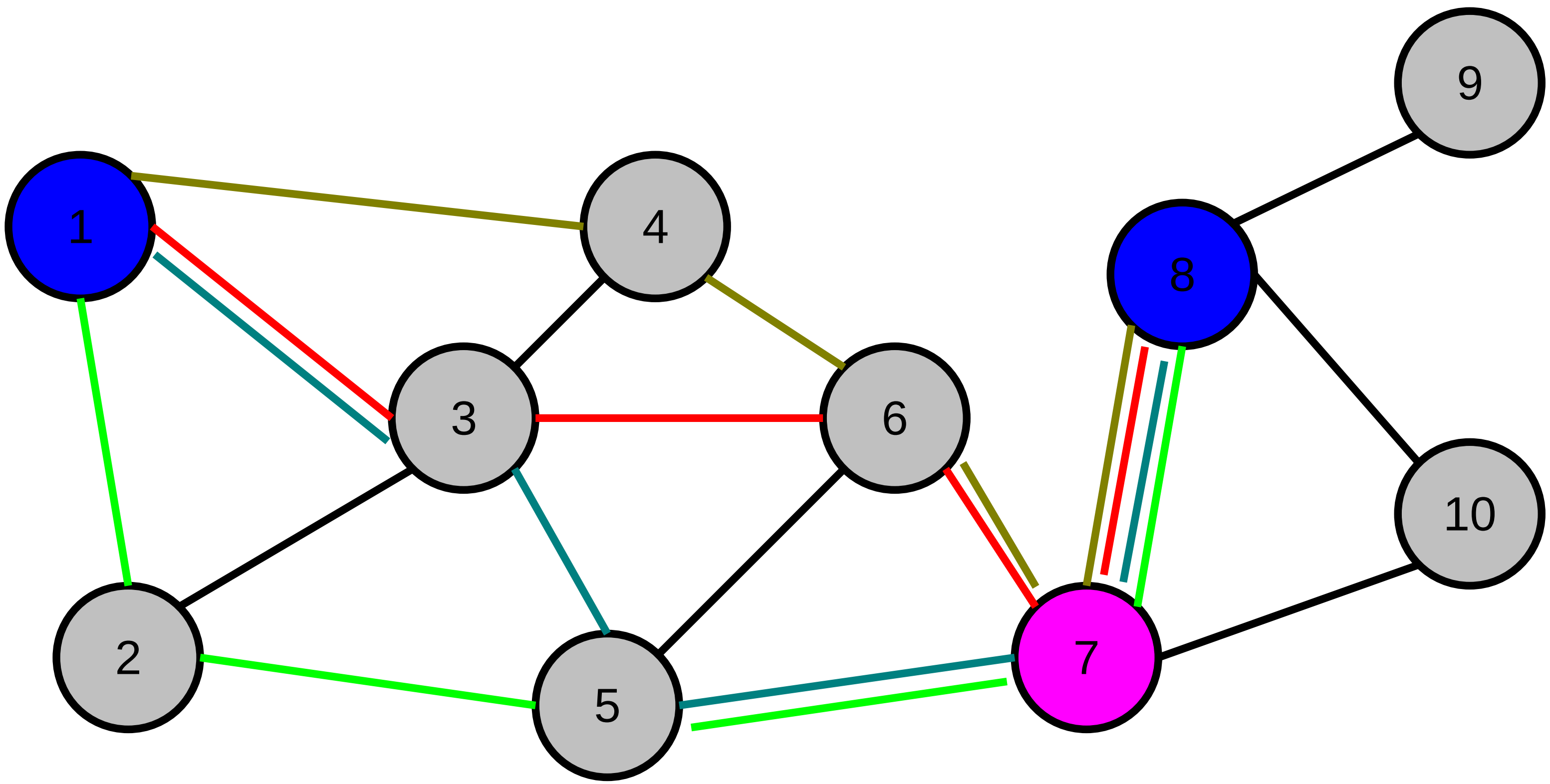
$$C_B(v_7) = \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{2} +$$

Centrality



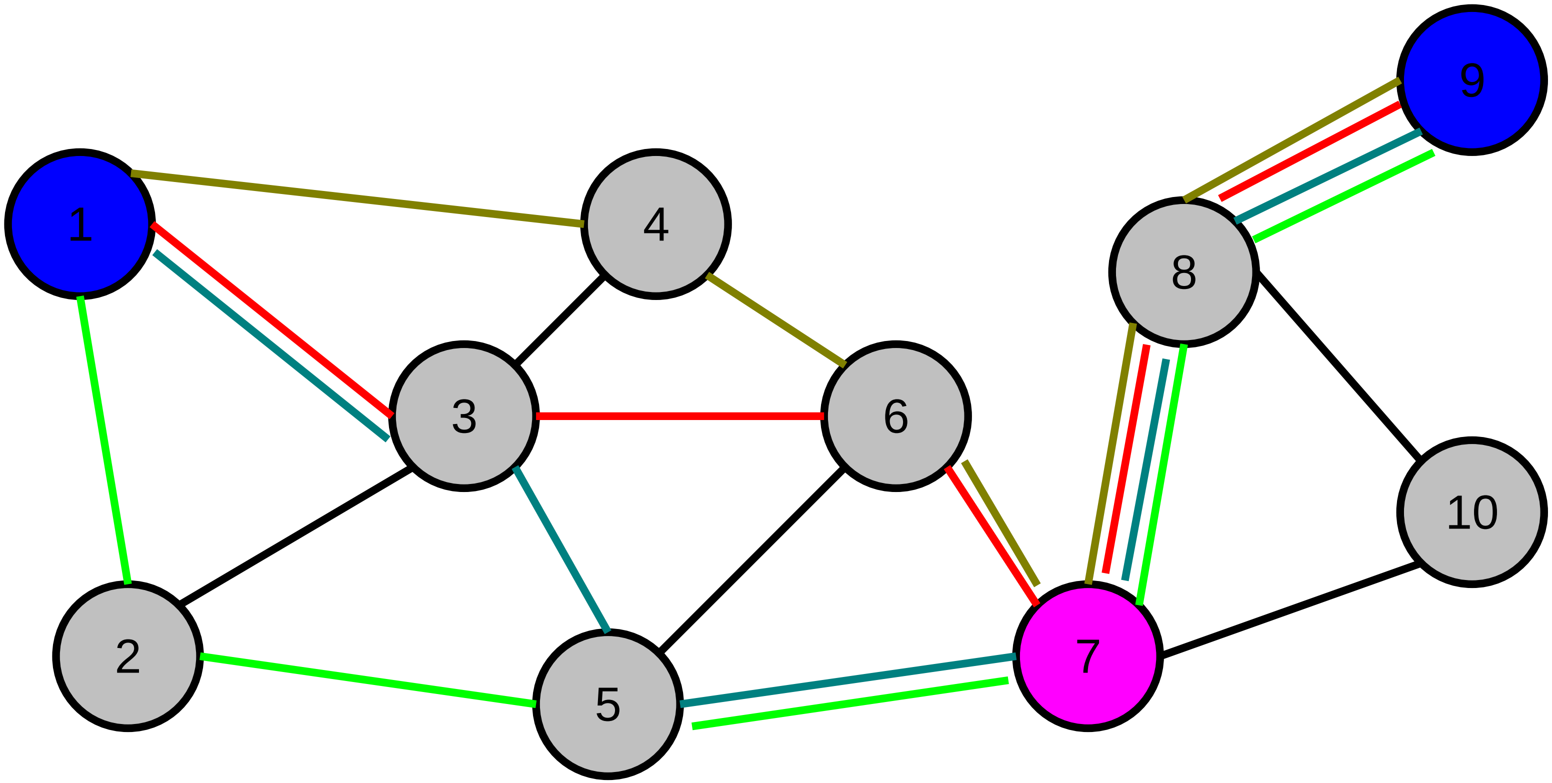
$$C_B(v_7) = \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{2} + \frac{0}{2} +$$

Centrality



$$C_B(v_7) = \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{2} + \frac{0}{2} + \frac{4}{4} +$$

Centrality



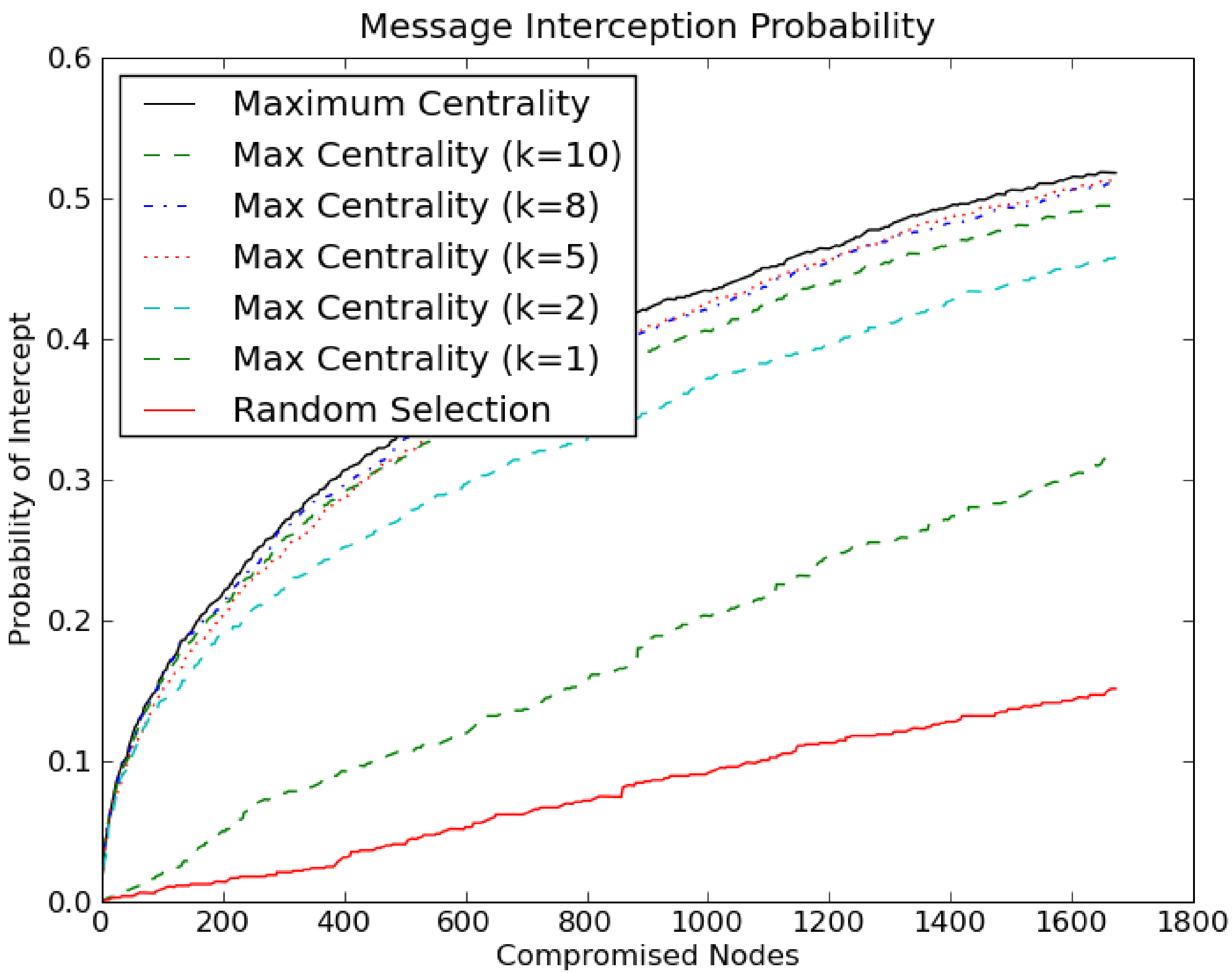
$$C_B(v_7) = \frac{0}{1} + \frac{0}{1} + \frac{0}{1} + \frac{0}{2} + \frac{0}{2} + \frac{4}{4} + \frac{4}{4} + \ldots$$

Message Interception Scenario

- Messages sent via shortest (least-cost) paths
- Adversary can compromise *N* nodes
- How much traffic can s/he intercept?

$$p_{intercept}(v_s, v_d) = \frac{C_B(v)}{|V|^2}$$

Message Interception



K = 1 is still twice as good as random selection

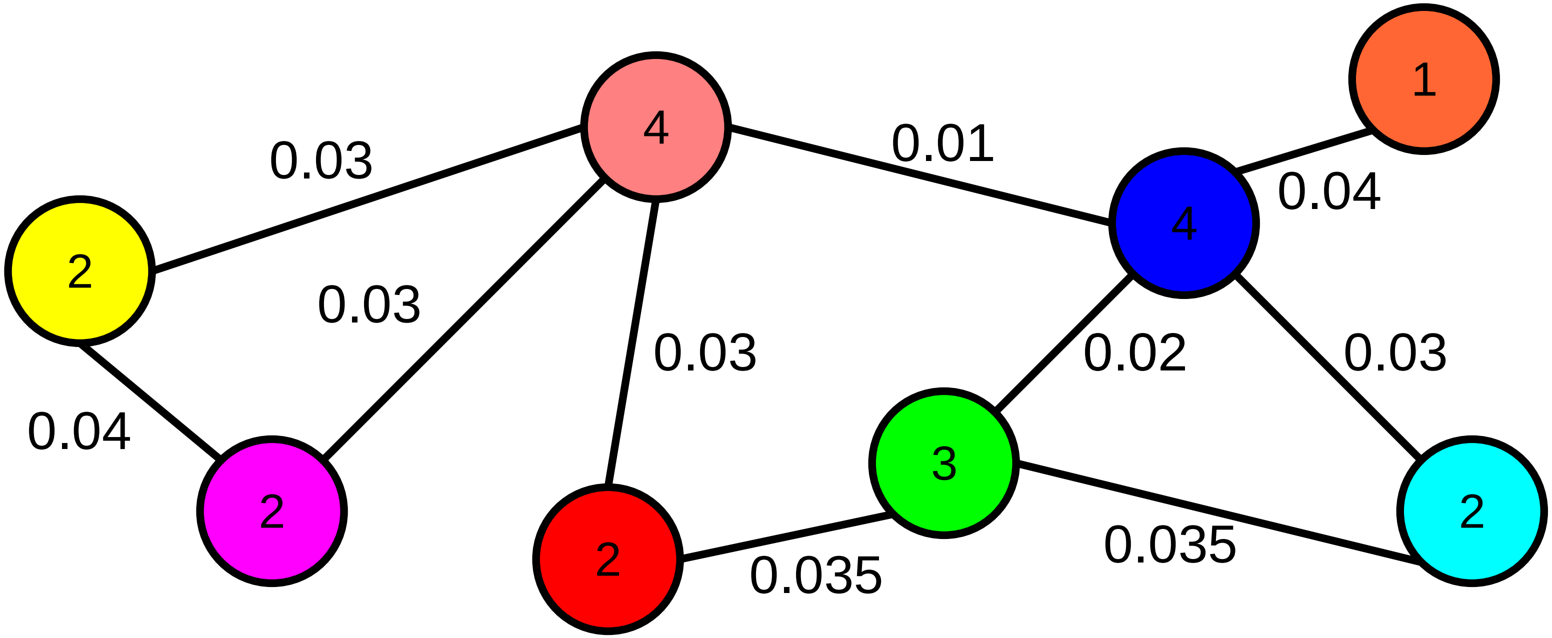
Community Detection

- Goal: Find highly-connected sub-groups
- Measure success by high *modularity*:

$$Q = \frac{1}{2m} \sum_{v,w} \left[A_{vw} - \frac{d(v)d(w)}{2m} \right]$$

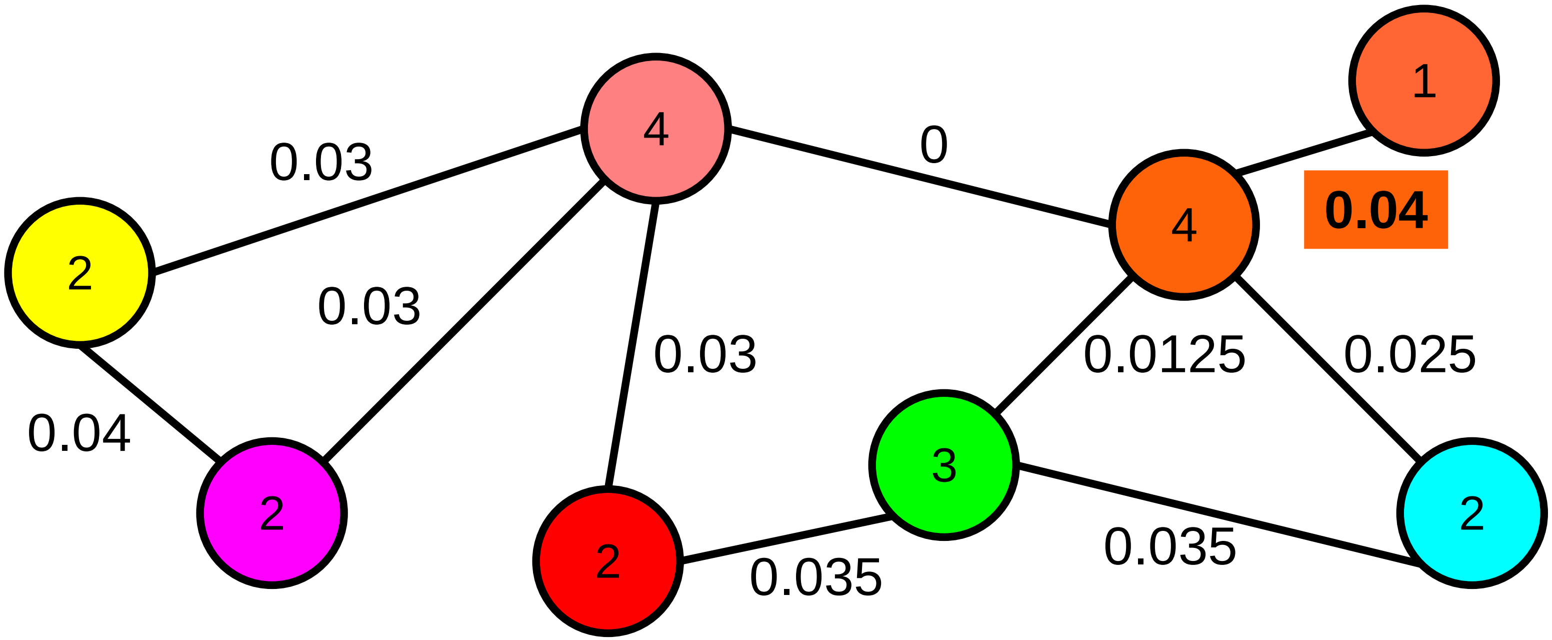
- Ratio of intra-community edges to random
- Normalised to be between -1 and 1

Community Detection



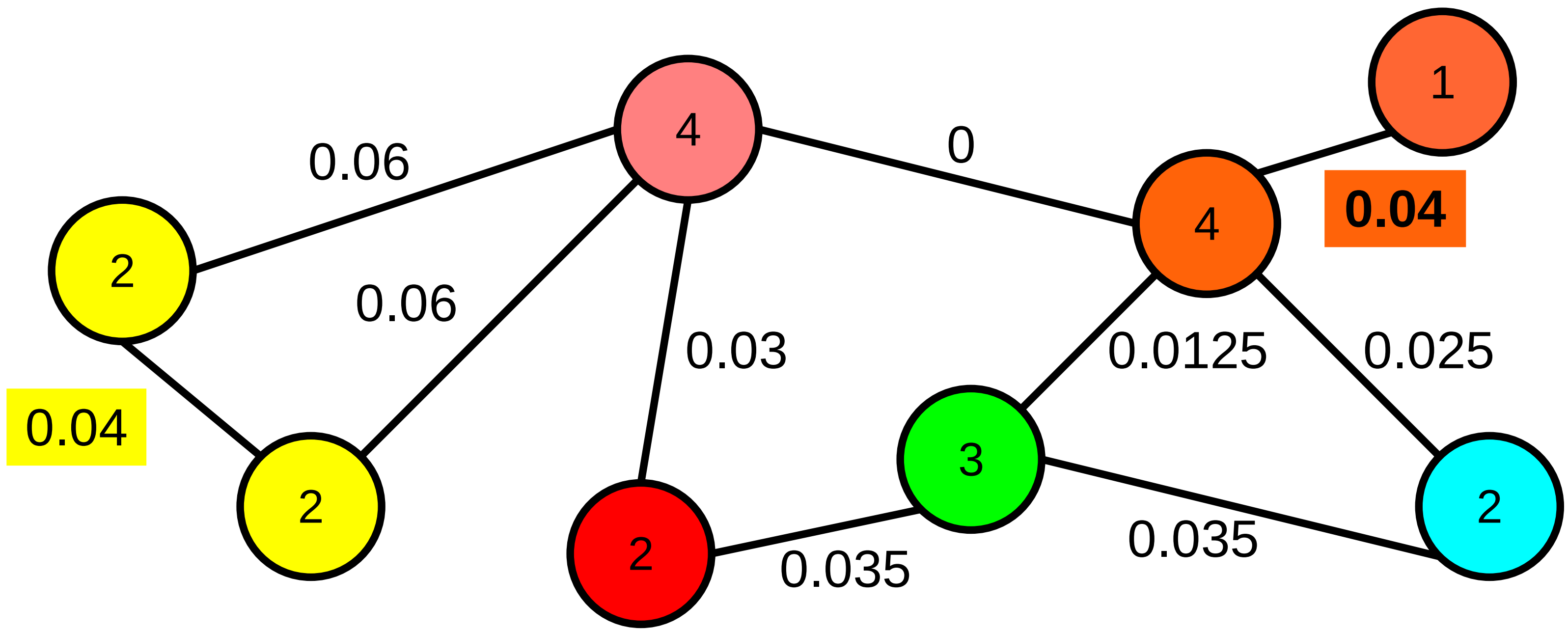
- Clausen et. al 2004 – find maximal modularity in O($n\lg^2 n$)
- Only track marginal modularity for edges
- Merging communities only affects adjacent edges

Community Detection



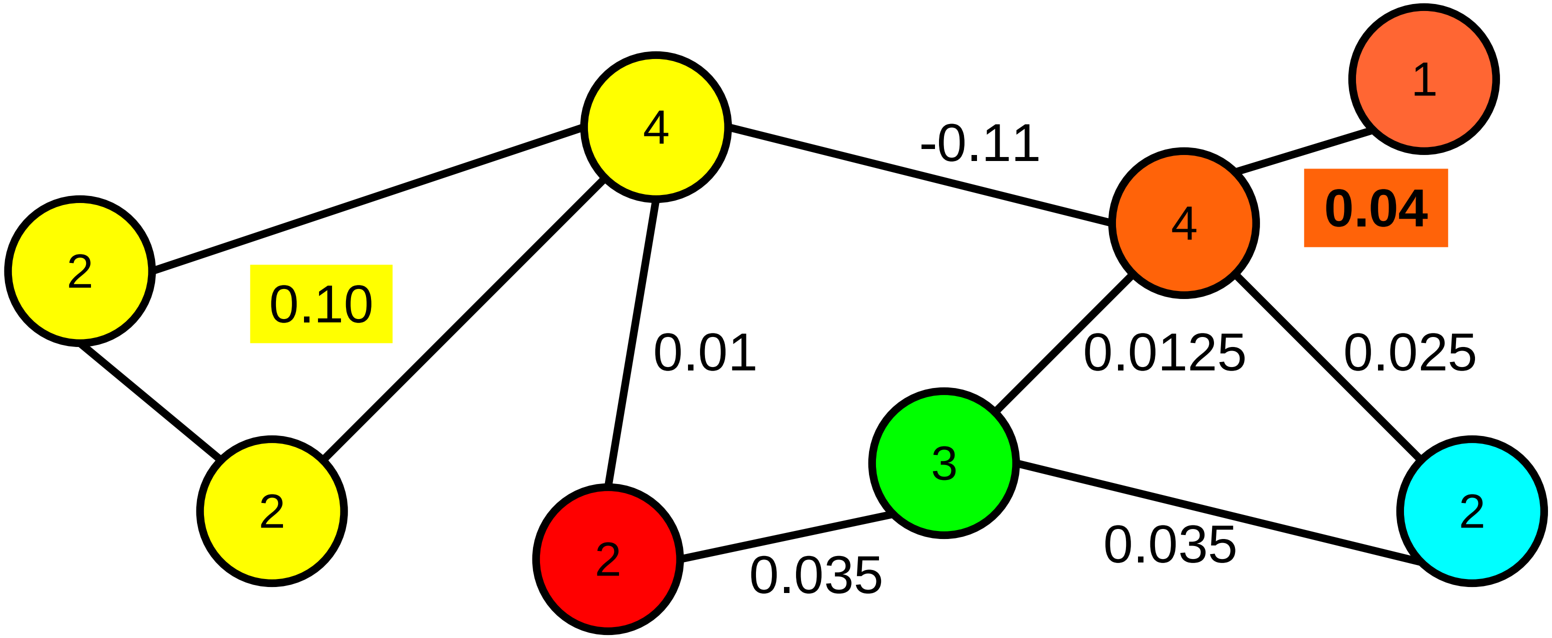
Q=0.04

Community Detection



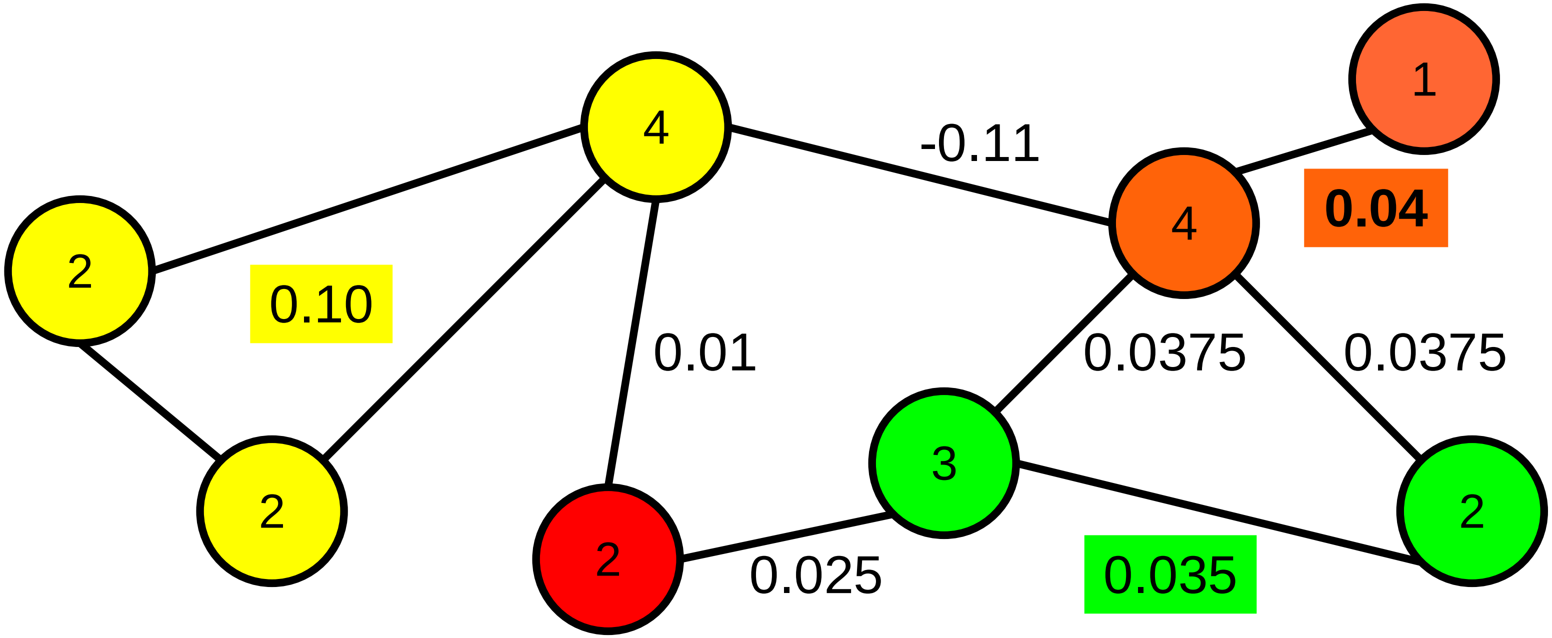
Q=0.08

Community Detection



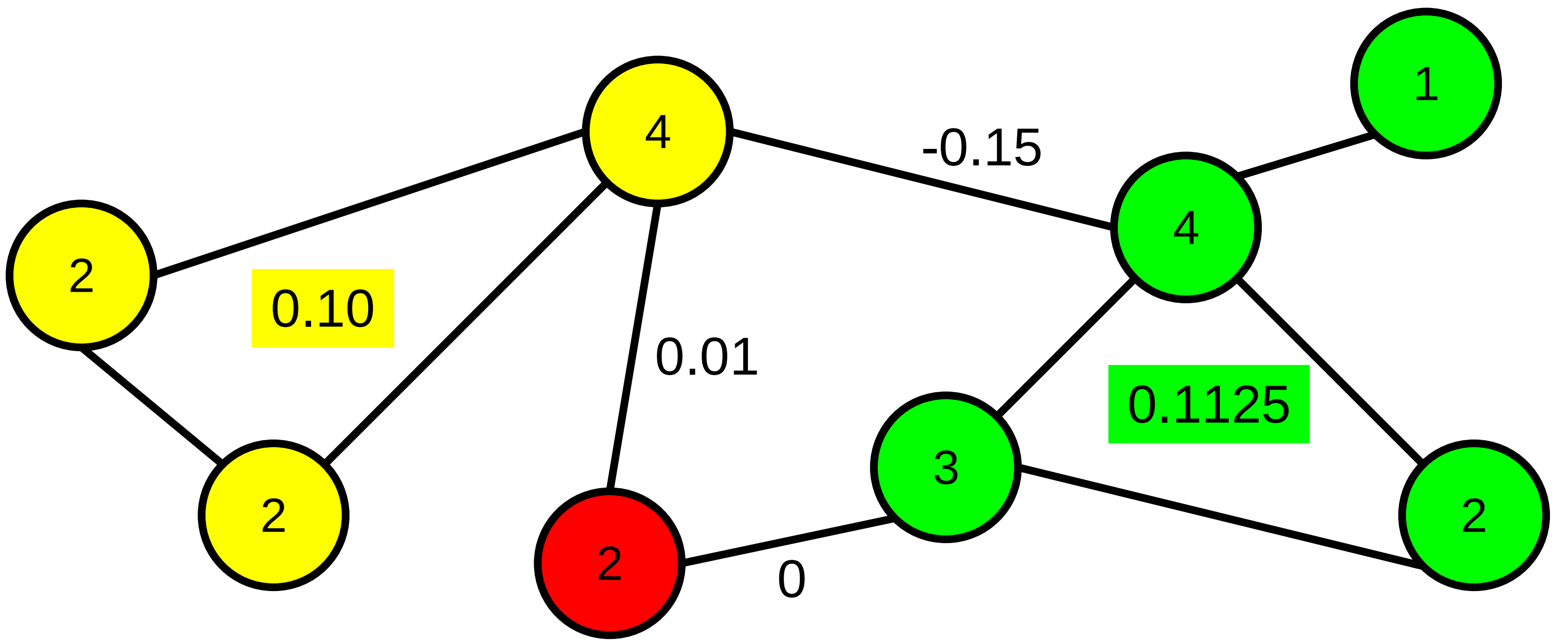
Q=0.14

Community Detection



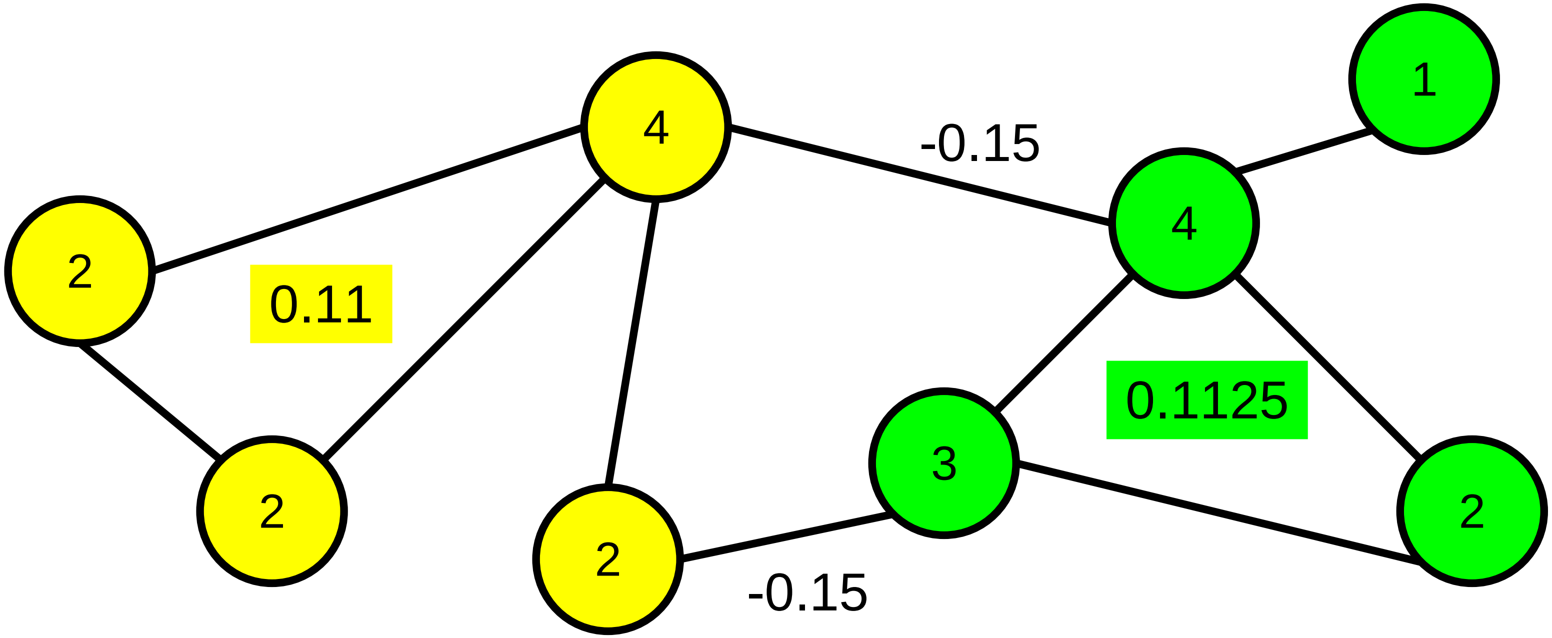
Q=0.175

Community Detection



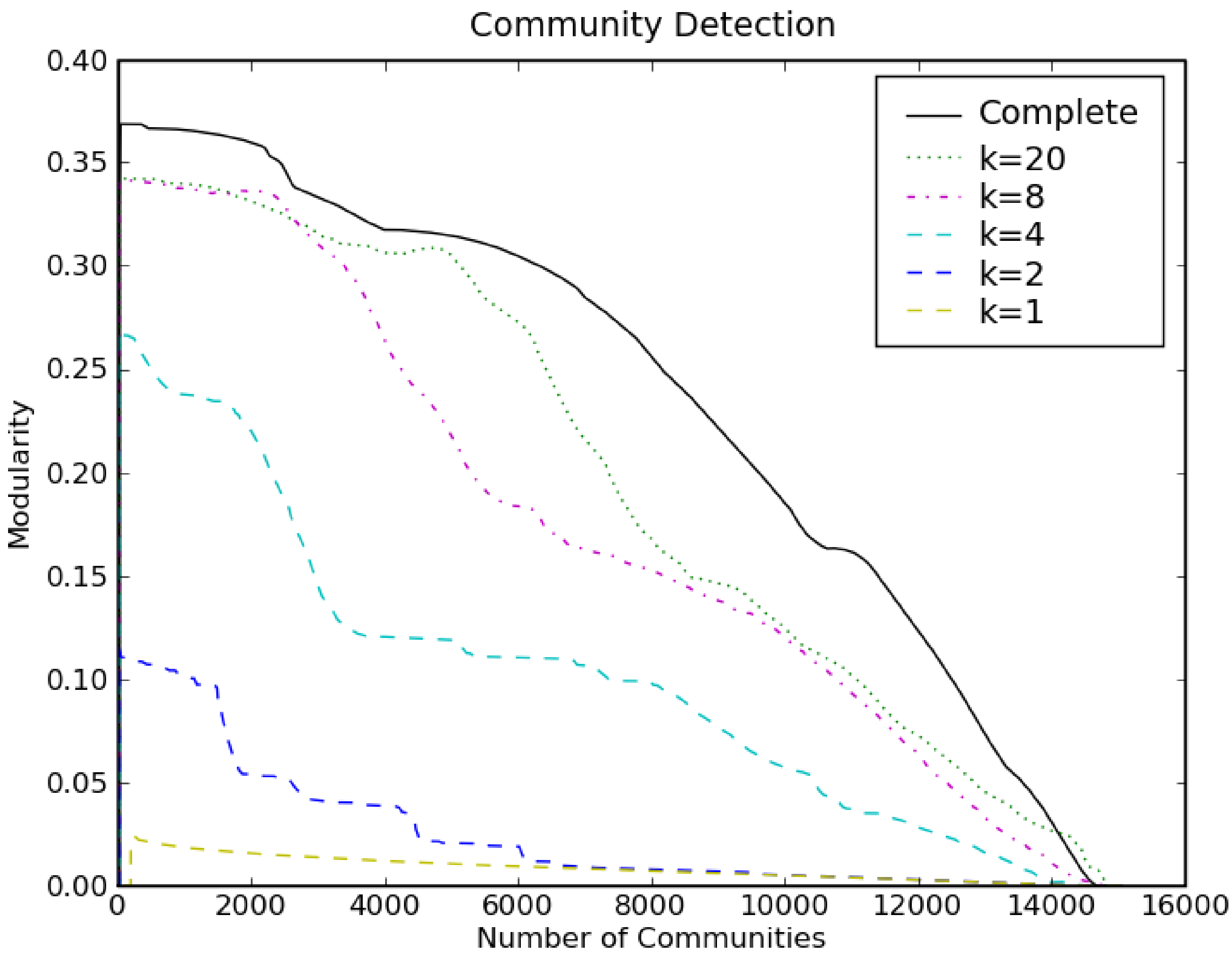
Q=0.2125

Community Detection



Q=0.2225

Community Detection



Works fairly well, much better for larger communities

Conclusions

- Social graph is fragile to partial disclosure
    - Consistent with Danezis/Wittneben, Nagaraja results
- Public Listings Leak Too Much
    - Dominating sets, centrality, communities in particular
- SNS's need a dedicated privacy review team
    - Comparable to security audit & penetration testing

Questions?

jcb82@cl.cam.ac.uk

jra40@cl.cam.ac.uk