

Graph Model Selection using Maximum Likelihood

Ivona Bezáková

Adam Tauman Kalai

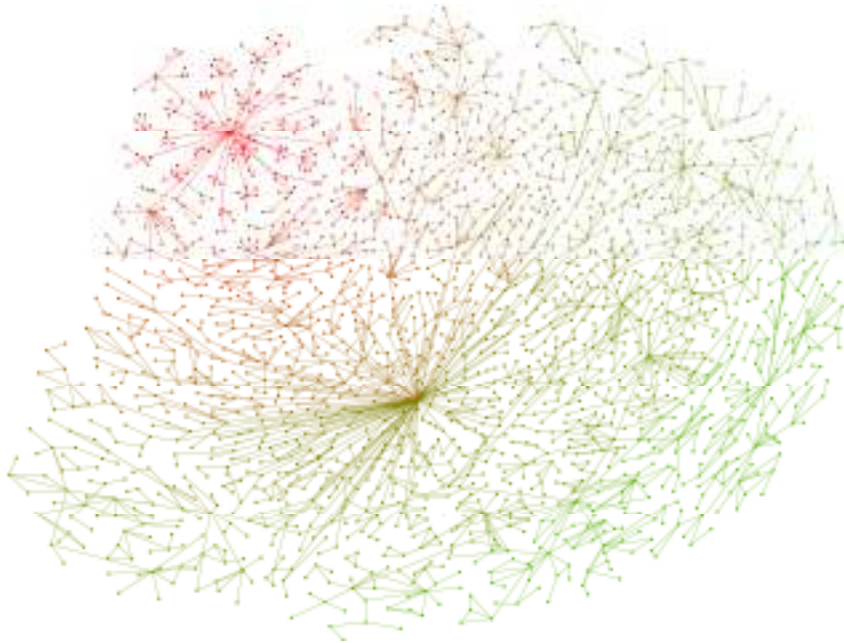
Rahul Santhanam

Theory Canal, Rochester, April 7th 2008

[ICML 2006 (International Conference on Machine Learning)]

Overview

Real-world network:
(Internet Graph)



Random graph
models:

Which model
to choose

?

Model A

Model B

Model C

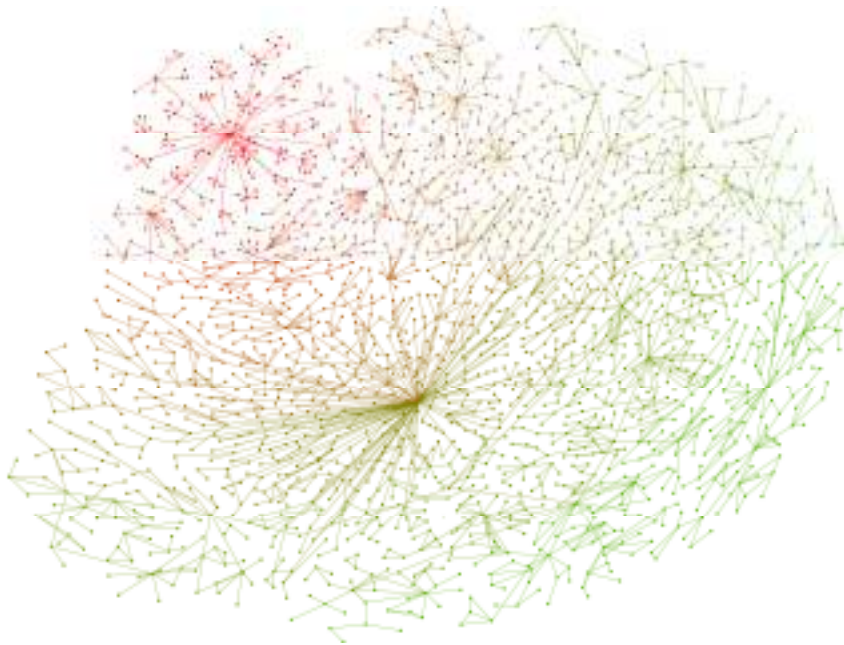
Model D

Model E

Picture downloaded from:
http://www.research.att.com/areas/visualization/projects_software/topfish.html

Overview

Real-world network:
(Internet Graph)



Random graph models:

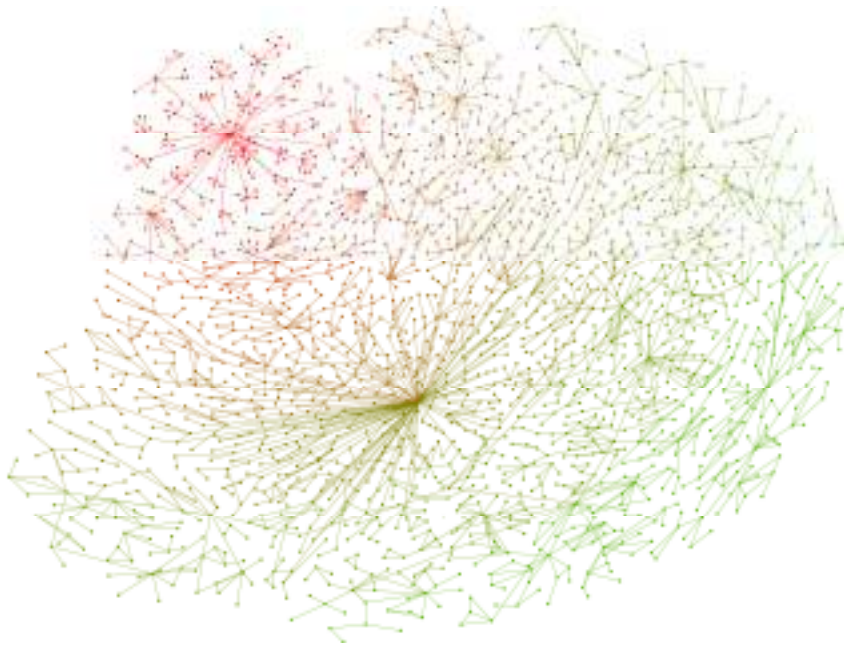
- ③ — Model A
- ⑦ — Model B
- ② — Model C
- ⑧ — Model D
- ⑤ — Model E

Picture downloaded from:
http://www.research.att.com/areas/visualization/projects_software/topfish.html



Overview

Real-world network:
(Internet Graph)



Random graph
models:

- ③ — Model A
- ⑦ — Model B
- ② — Model C
- ⑧ — Model D
- ⑤ — Model E

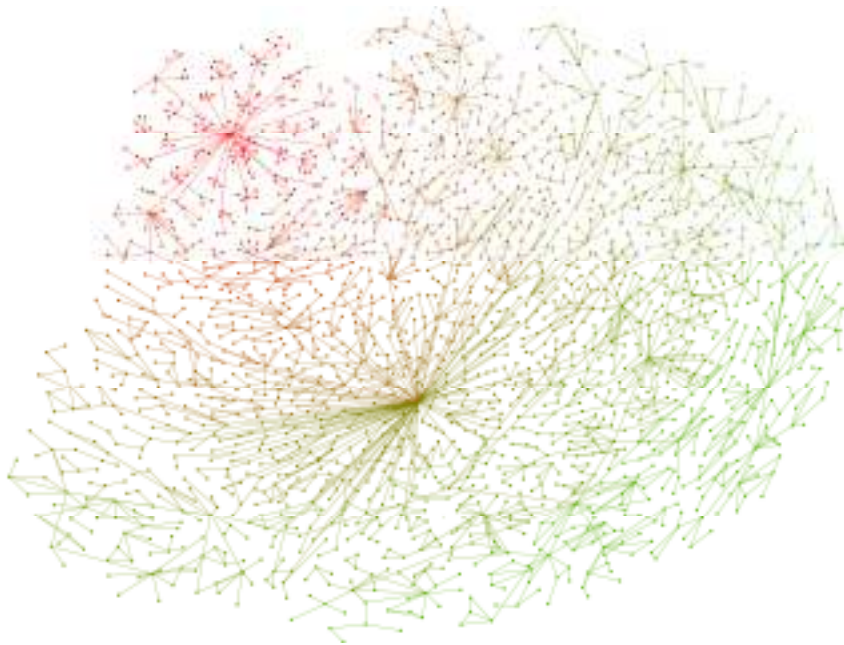
WINNER:



Picture downloaded from:
http://www.research.att.com/areas/visualization/projects_software/topfish.html

Overview

Real-world network:
(Internet Graph)



Random graph
models:

- ③ — Model A
- ⑦ — Model B
- ② — Model C
- ⑧ — Model D
- ⑤ — Model E

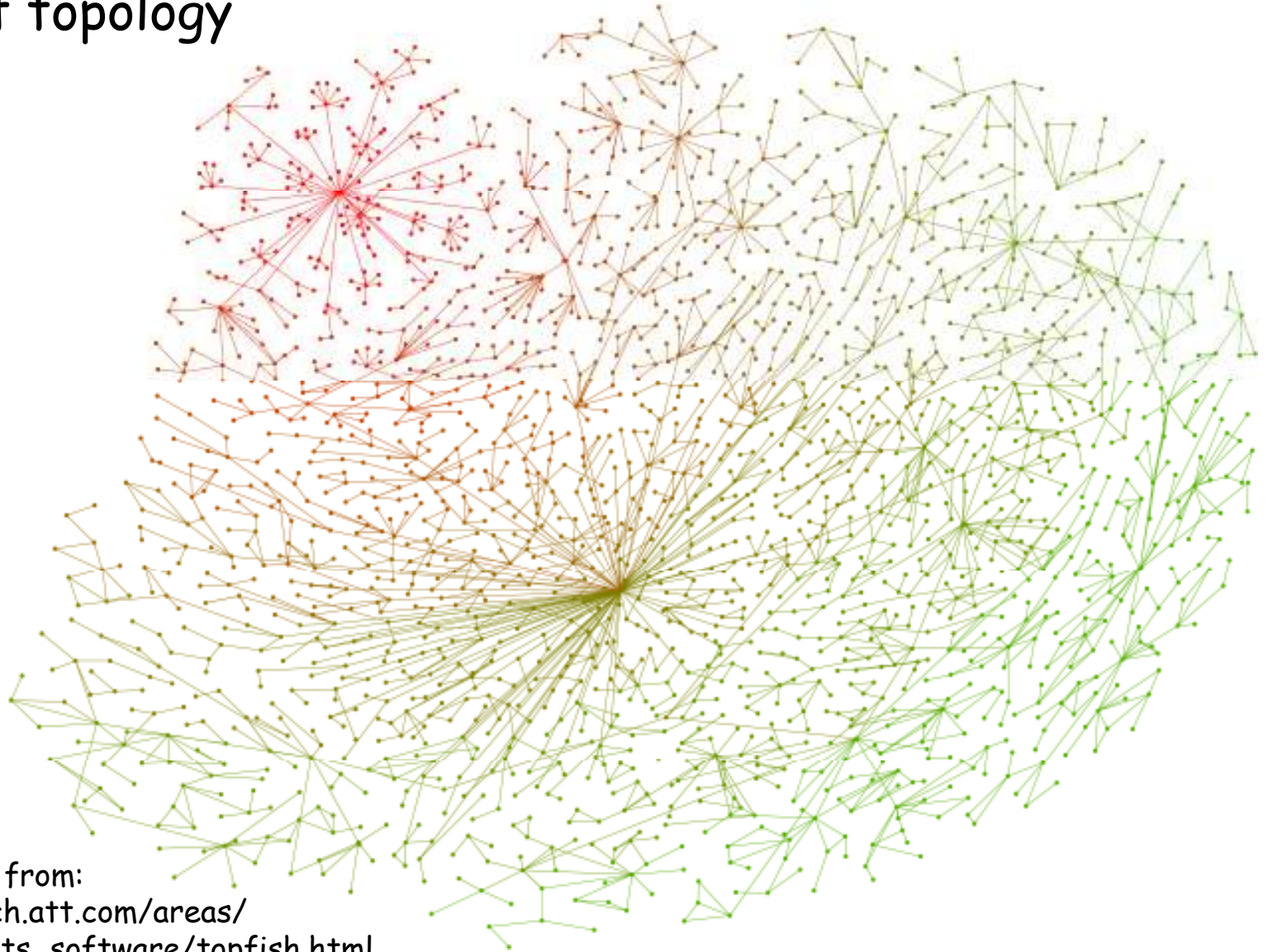
WINNER:

How to choose the score?



Real-world networks

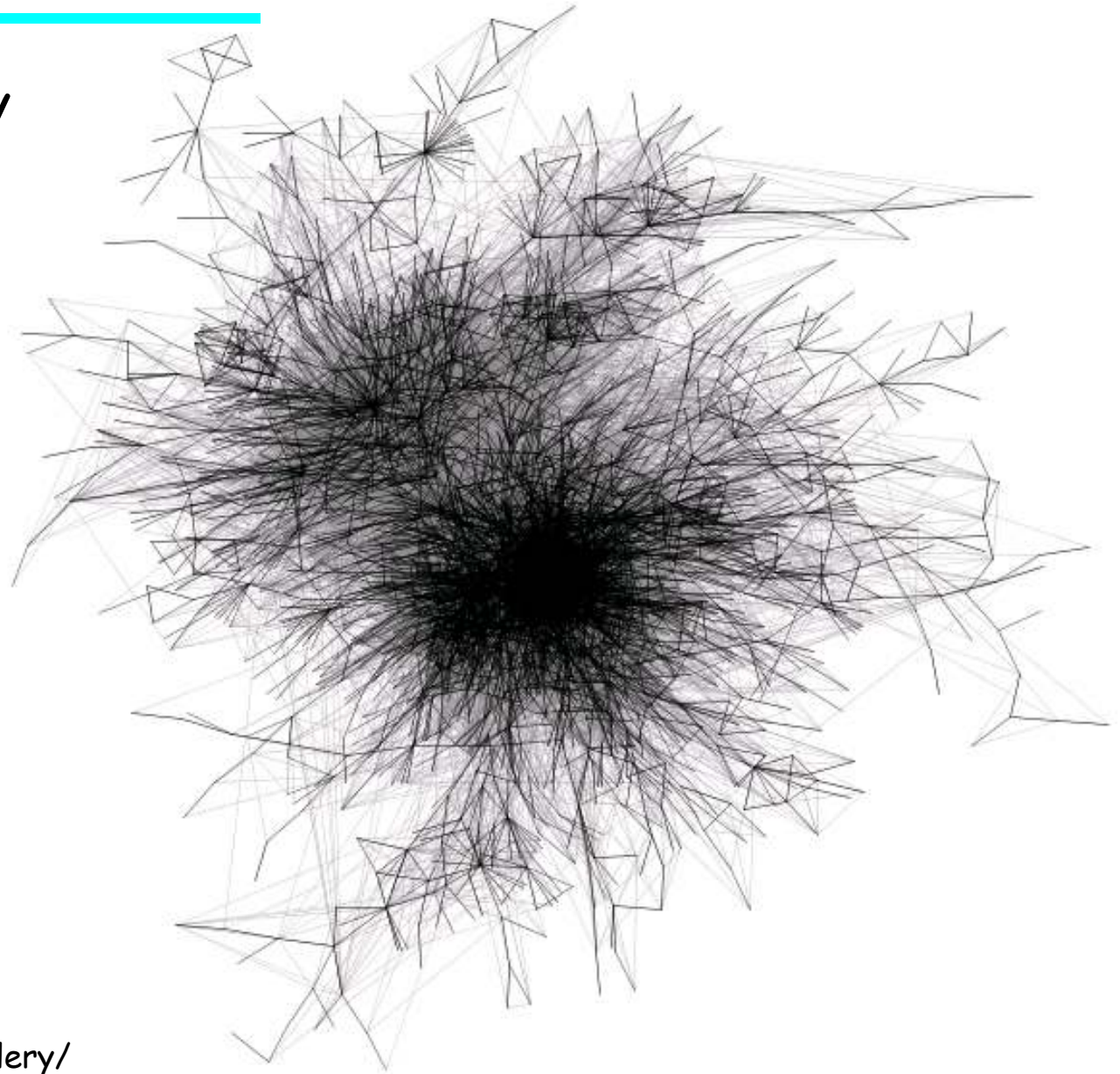
- Internet topology



Picture downloaded from:
http://www.research.att.com/areas/visualization/projects_software/topfish.html

Real-world networks

- Internet topology
- WWW



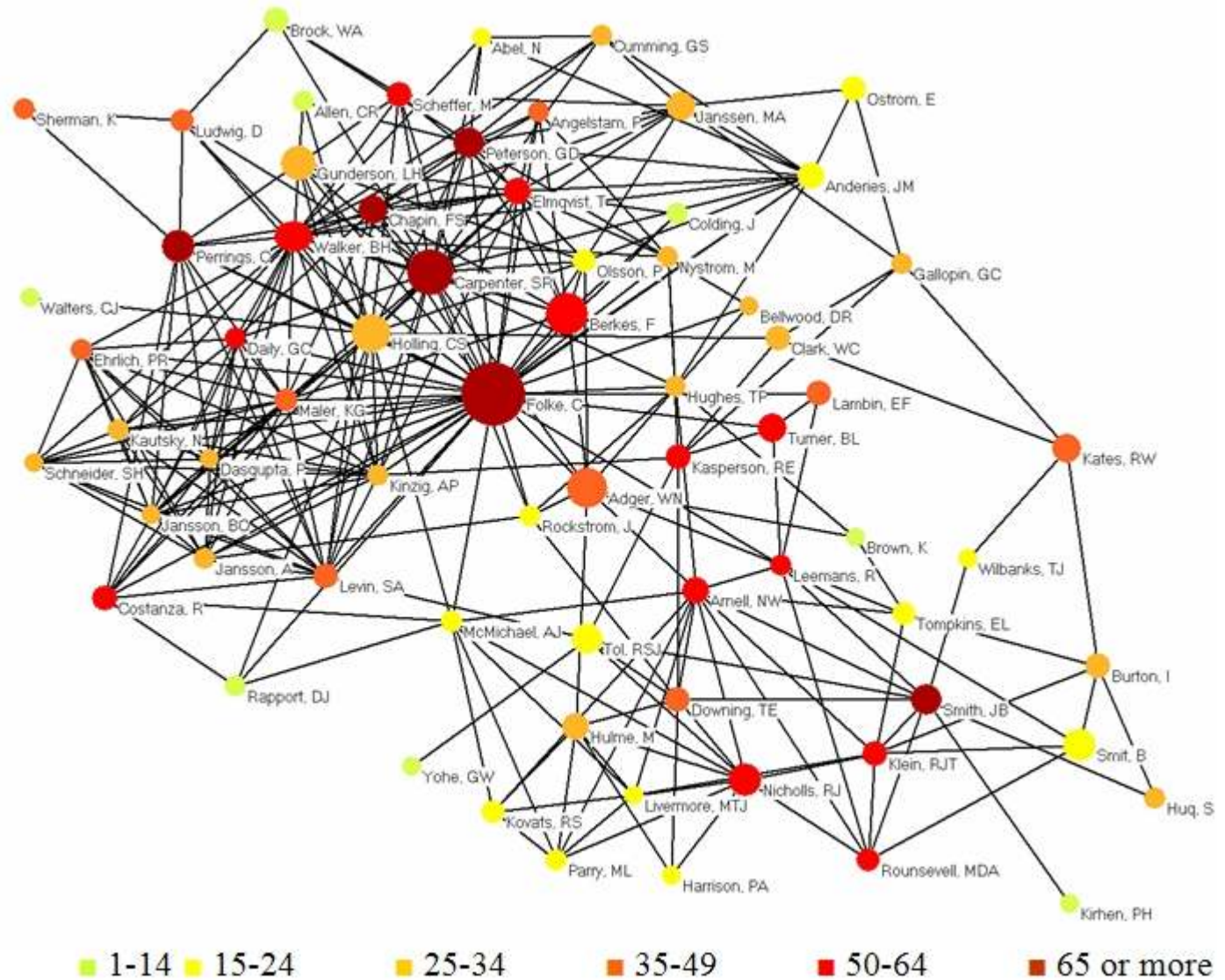
Picture downloaded from:
[http://datamining.typepad.com/gallery/
blog-map-gallery.html](http://datamining.typepad.com/gallery/blog-map-gallery.html)

Real-world networks

- Internet topo
- WWW
- Collaboration networks

Co-author network of the most productive and best connected authors with the strongest co-authorship relations. Circles denote author nodes, and are labeled by the author's last name and initials.

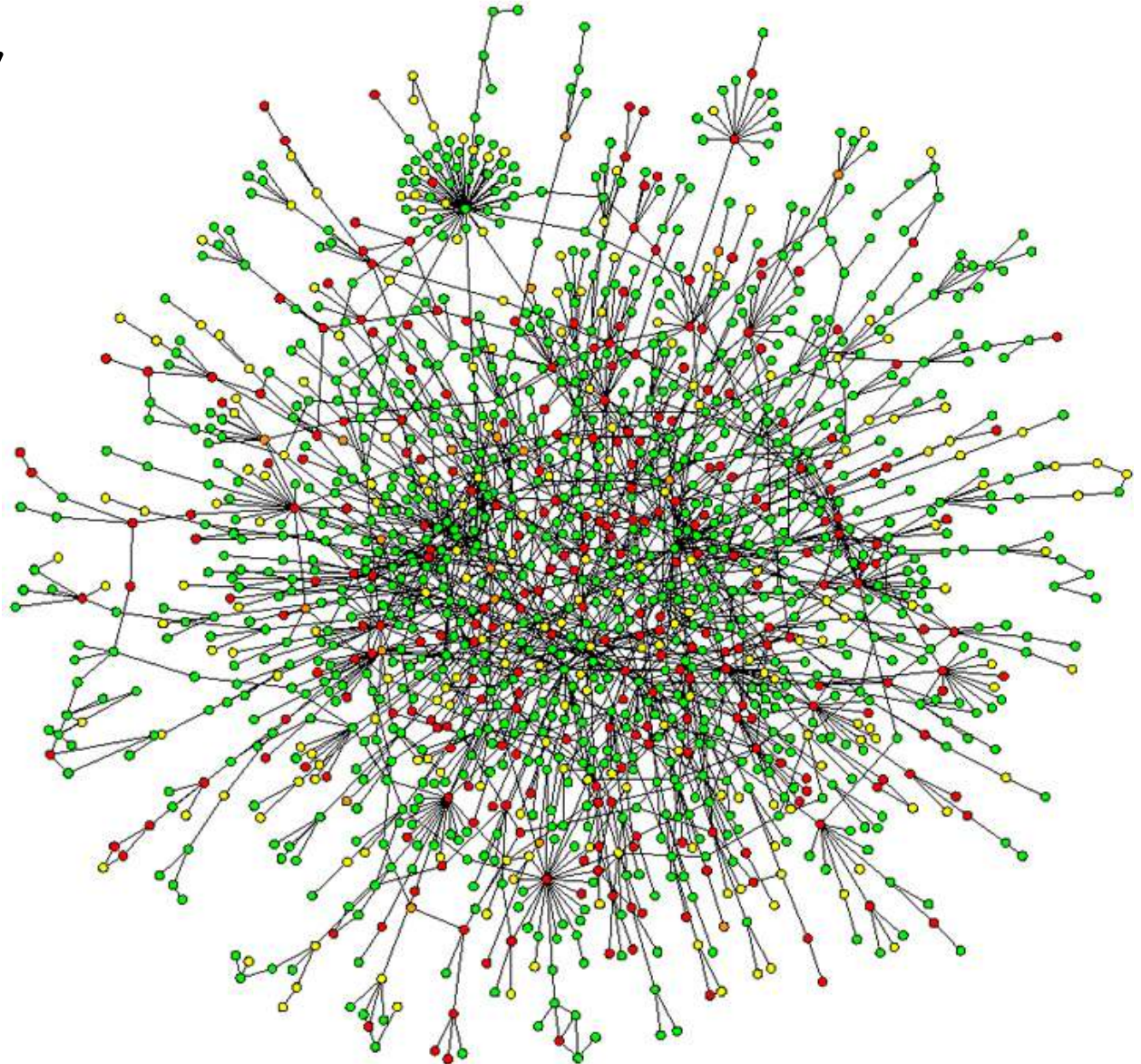
Legend: Node - author; Node area size—# of publications; Node area color—# of unique co-authors



Picture downloaded from:
<http://www.ecologyandsociety.org/vol12/iss2/art9/figure2.html>

Real-world networks

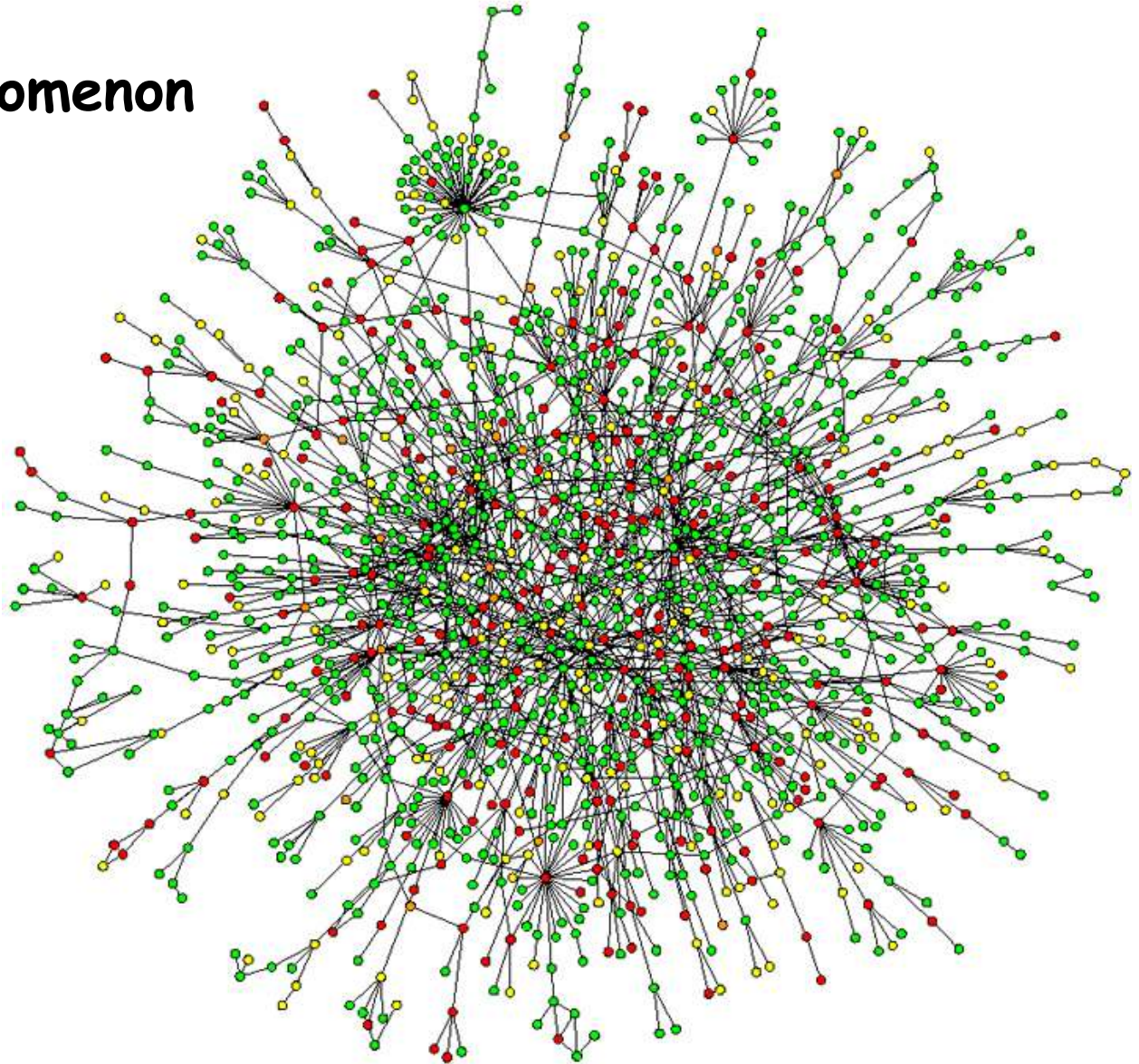
- Internet topology
- WWW
- Collaboration networks
- Protein interactions
- etc.



Picture downloaded from:
<http://www.math.cornell.edu/~durrett/RGD/RGD.html>

Properties of Complex Networks

- small-world phenomenon



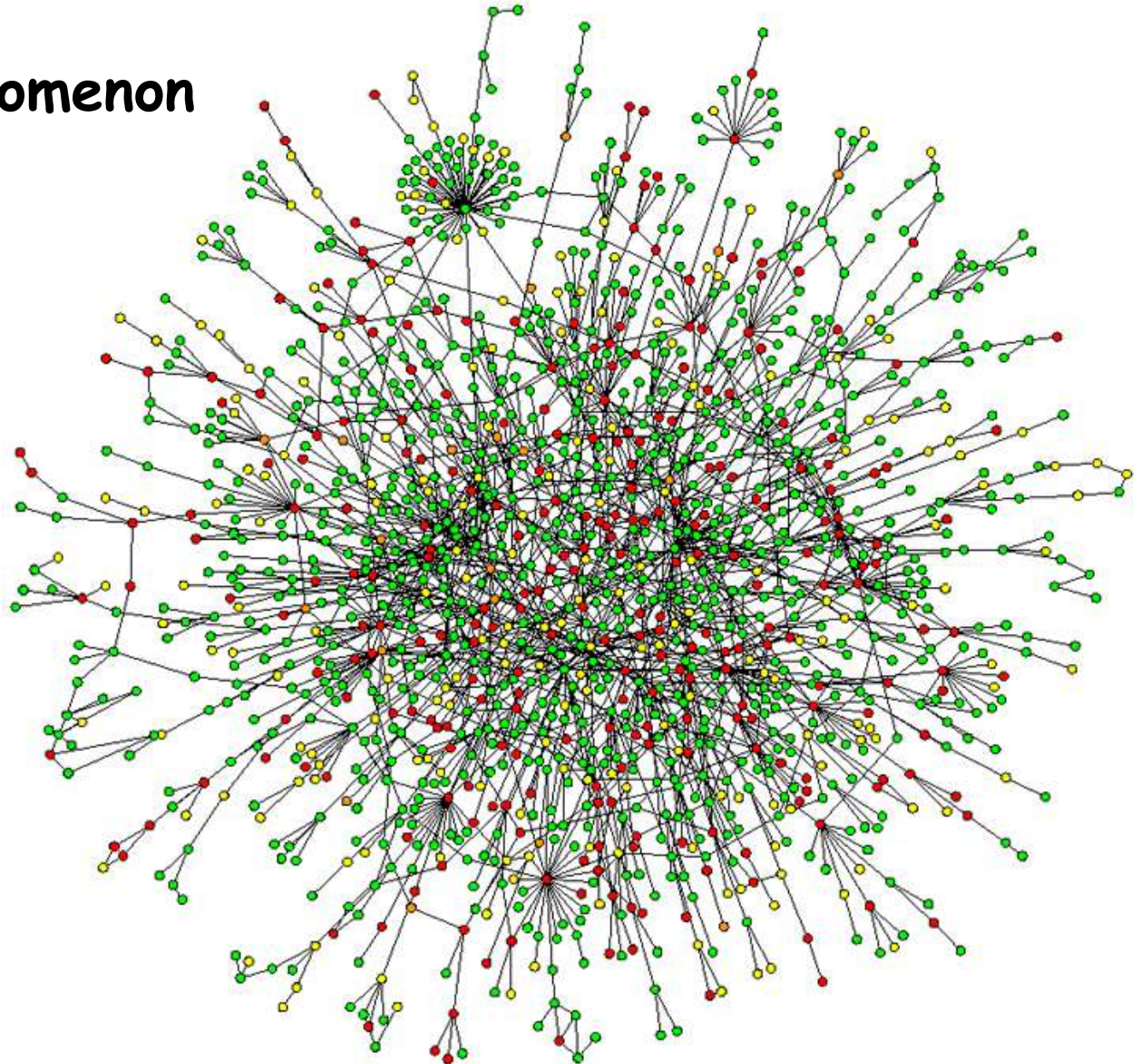
Picture downloaded from:
<http://www.math.cornell.edu/~durrett/RGD/RGD.html>

Properties of Complex Networks

- **small-world phenomenon**

- six degrees of separation

[Milgram '67]



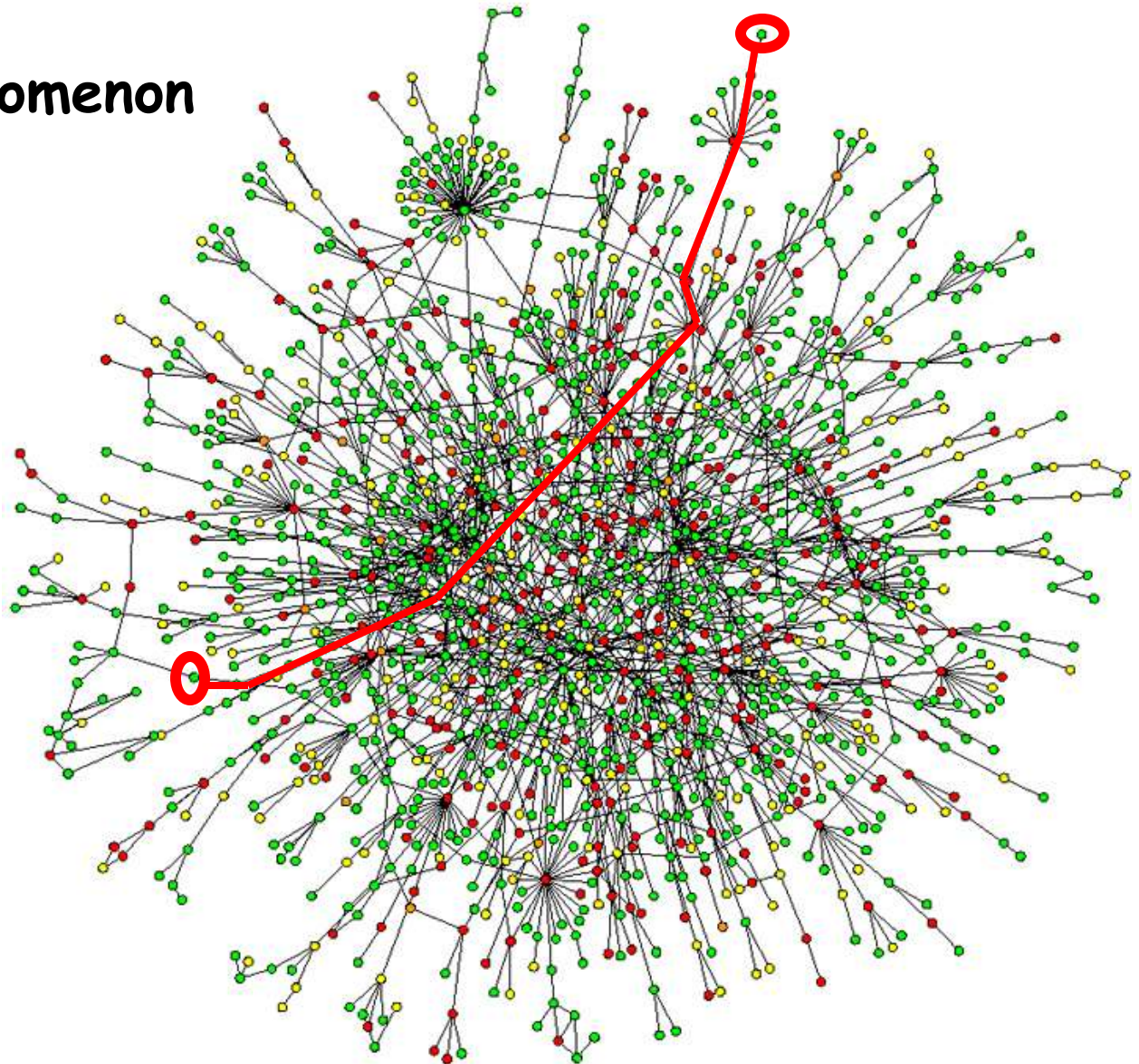
Picture downloaded from:
<http://www.math.cornell.edu/~durrett/RGD/RGD.html>

Properties of Complex Networks

- **small-world phenomenon**

- six degrees of separation

[Milgram '67]



Picture downloaded from:
<http://www.math.cornell.edu/~durrett/RGD/RGD.html>

Properties of Complex Networks

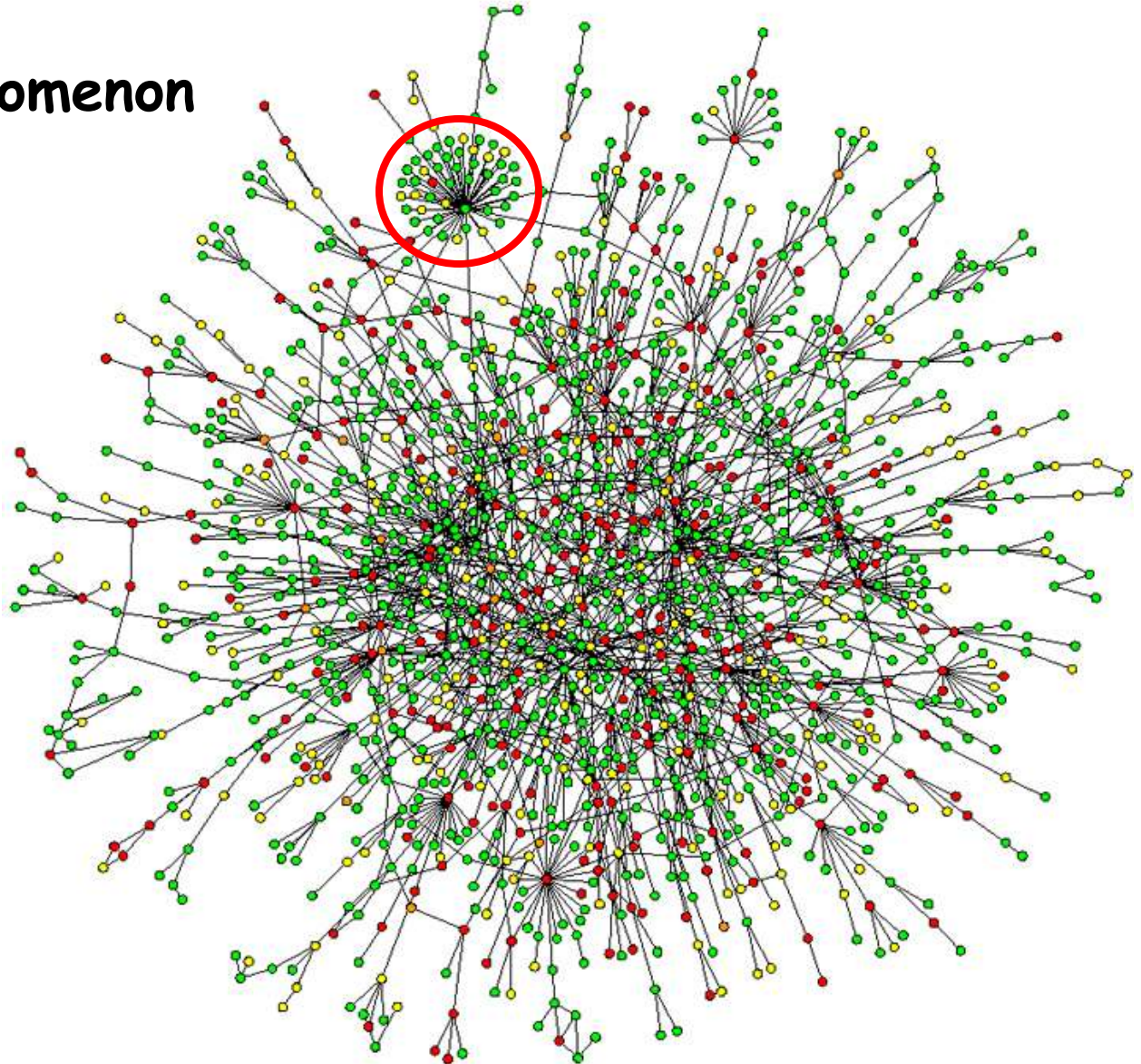
- **small-world phenomenon**

- six degrees of separation

[Milgram '67]

- high clustering

[Watts & Strogatz '98]



Picture downloaded from:
<http://www.math.cornell.edu/~durrett/RGD/RGD.html>

Properties of Complex Networks

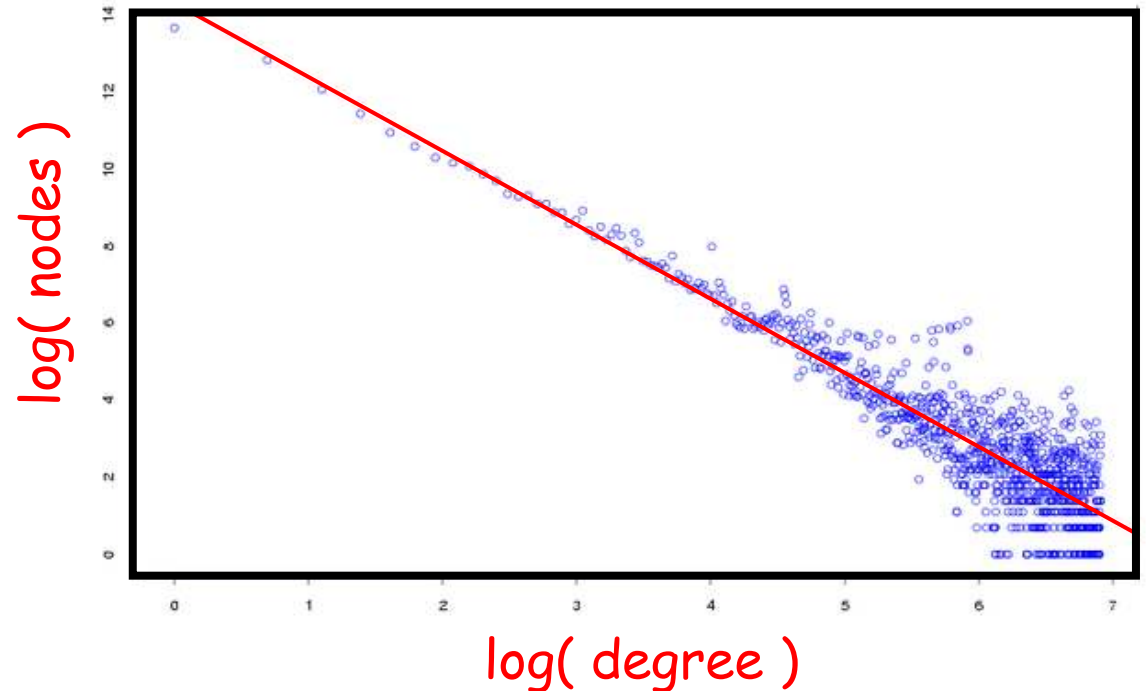
- **small-world phenomenon**

- six degrees of separation

- [Milgram '67]

- high clustering

- [Watts & Strogatz '98]



- **power-law degree distribution**

- [Siganos & Faloutsos '03]

Parameters: β (exponent), c (cutoff)

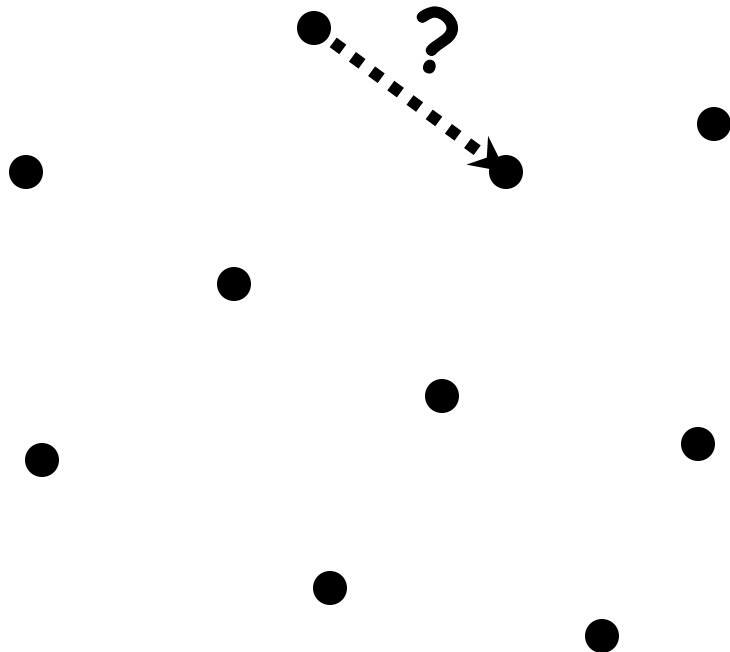
$i \in \{1, \dots, c\}$ proportional to $i^{-\beta}$

Modeling Complex Networks

Erdős-Rényi (the basic random graph model)

Parameters: n - # vertices

p - probability of an edge



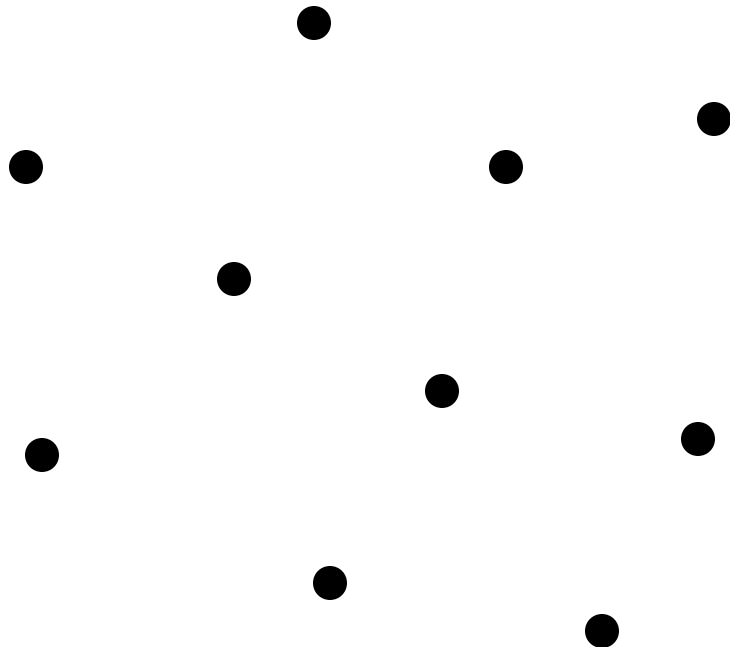
For every pair of vertices:
include edge with
probability p

Modeling Complex Networks

Powerlaw Random Graph [Bollobás '85; Aiello, Chung, Lu '00]

Parameters: n - # vertices

$\beta_{in}, c_{in}, \beta_{out}, c_{out}$ - powerlaw distribution



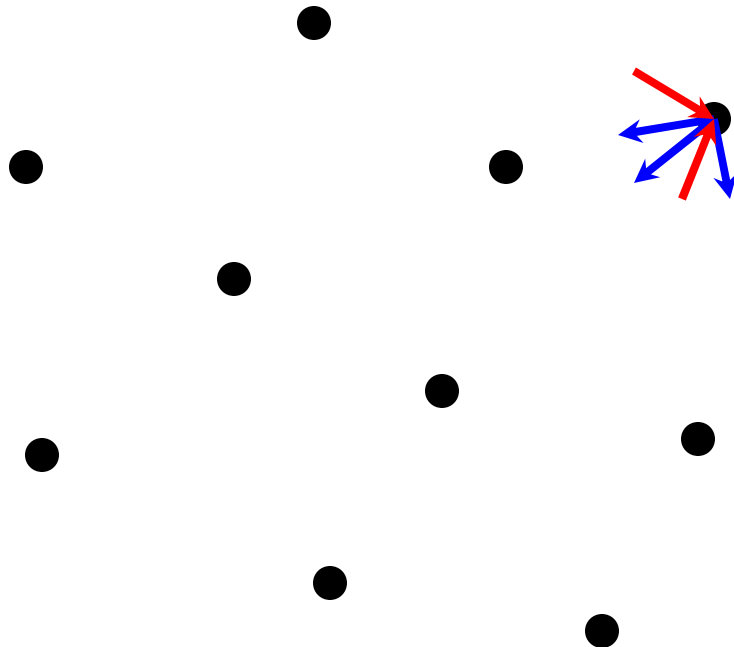
1. For every vertex:
generate **indeg**, **outdeg**
2. Randomly match
red/blue half-edges

Modeling Complex Networks

Powerlaw Random Graph [Bollobás '85; Aiello, Chung, Lu '00]

Parameters: n - # vertices

$\beta_{in}, c_{in}, \beta_{out}, c_{out}$ - powerlaw distribution



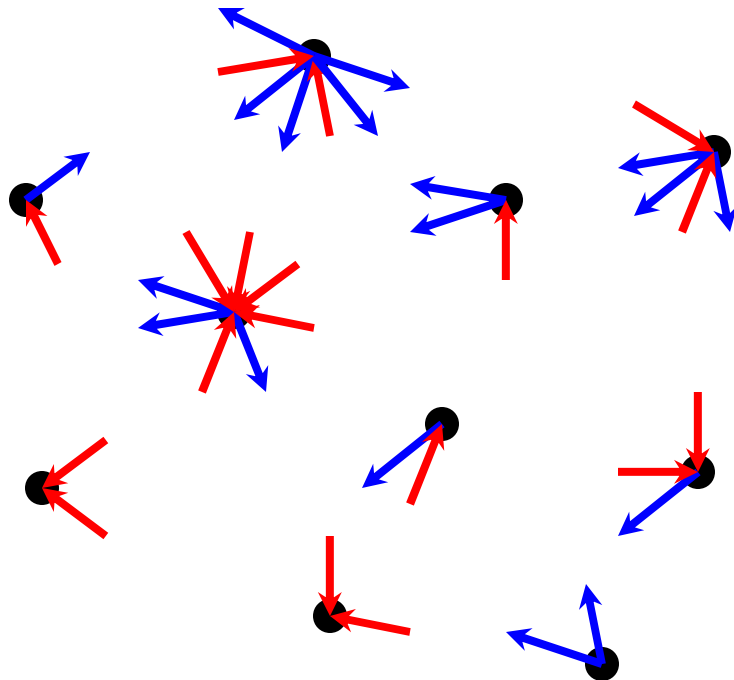
1. For every vertex:
generate **indeg**, **outdeg**
2. Randomly match
red/blue half-edges

Modeling Complex Networks

Powerlaw Random Graph [Bollobás '85; Aiello, Chung, Lu '00]

Parameters: n - # vertices

$\beta_{in}, c_{in}, \beta_{out}, c_{out}$ - powerlaw distribution



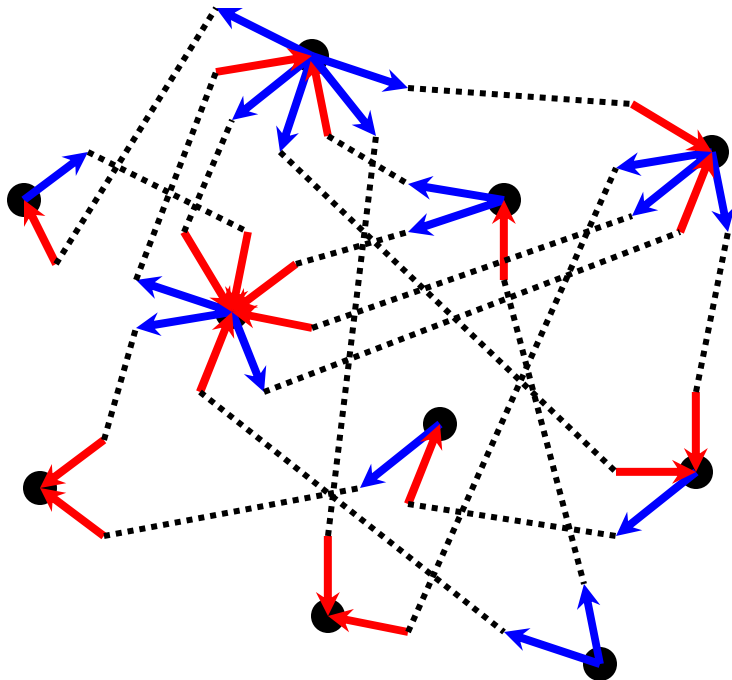
1. For every vertex:
generate **indeg**, **outdeg**
2. Randomly match
red/blue half-edges

Modeling Complex Networks

Powerlaw Random Graph [Bollobás '85; Aiello, Chung, Lu '00]

Parameters: n - # vertices

$\beta_{in}, c_{in}, \beta_{out}, c_{out}$ - powerlaw distribution



1. For every vertex:
generate **indeg**, **outdeg**
2. Randomly match
red/blue half-edges

Modeling Complex Networks

Preferential Attachment [Mitzenmacher '01]

Parameters: n - # vertices γ - "self-loop"

p, q ($p+q < 1$) - probability of **out**/**in** edge

1. Start with a single vertex
2. In iteration $i=2, \dots, n$ create edges between vtx i and $< i$. Repeat until c):
 - a) with prob. p **out**edge
 - b) with prob. q **in**edge
 - c) with prob. $1-p-q$ next iter.

In 2. other end-point chosen prop. to **in**/**out**degree $+\gamma$.

Modeling Complex Networks

Preferential Attachment [Mitzenmacher '01]

Parameters: n - # vertices γ - "self-loop"

p, q ($p+q < 1$) - probability of **out**/**in** edge

1 ●

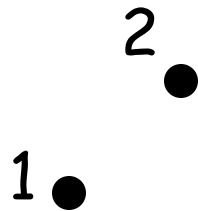
1. Start with a single vertex
2. In iteration $i=2, \dots, n$ create edges between vtx i and $<i$. Repeat until c):
 - a) with prob. p **out**edge
 - b) with prob. q **in**edge
 - c) with prob. $1-p-q$ next iter.In 2. other end-point chosen prop. to **in**/**out**degree $+\gamma$.

Modeling Complex Networks

Preferential Attachment [Mitzenmacher '01]

Parameters: n - # vertices γ - "self-loop"

p, q ($p+q < 1$) - probability of **out**/**in** edge



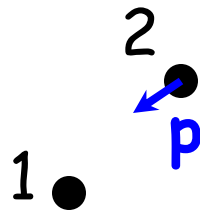
1. Start with a single vertex
2. In iteration $i=2, \dots, n$ create edges between vtx i and $<i$. Repeat until c):
 - a) with prob. p **out**edge
 - b) with prob. q **in**edge
 - c) with prob. $1-p-q$ next iter.In 2. other end-point chosen prop. to **in**/**out**degree $+\gamma$.

Modeling Complex Networks

Preferential Attachment [Mitzenmacher '01]

Parameters: n - # vertices γ - "self-loop"

p, q ($p+q < 1$) - probability of **out**/**in** edge



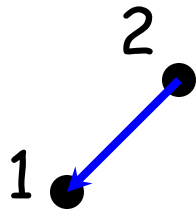
1. Start with a single vertex
2. In iteration $i=2, \dots, n$ create edges between vtx i and $<i$. Repeat until c):
 - a) with prob. **p** outedge
 - b) with prob. **q** inedge
 - c) with prob. $1-p-q$ next iter.In 2. other end-point chosen prop. to **in**/**out**degree $+\gamma$.

Modeling Complex Networks

Preferential Attachment [Mitzenmacher '01]

Parameters: n - # vertices γ - "self-loop"

p, q ($p+q < 1$) - probability of **out**/**in** edge



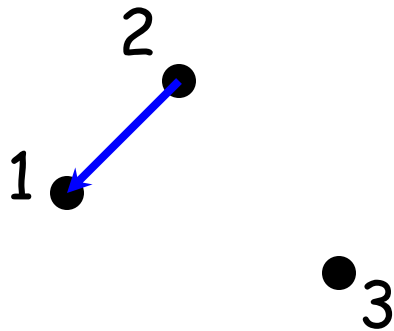
1. Start with a single vertex
2. In iteration $i=2, \dots, n$ create edges between vtx i and $<i$. Repeat until c):
 - a) with prob. p **out**edge
 - b) with prob. q **in**edge
 - c) with prob. $1-p-q$ next iter.In 2. other end-point chosen prop. to **in**/**out**degree $+\gamma$.

Modeling Complex Networks

Preferential Attachment [Mitzenmacher '01]

Parameters: n - # vertices γ - "self-loop"

p, q ($p+q < 1$) - probability of **out**/**in** edge



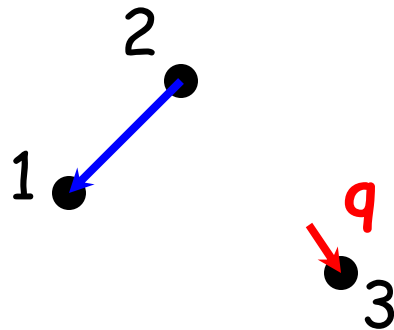
1. Start with a single vertex
2. In iteration $i=2, \dots, n$ create edges between vtx i and $<i$. Repeat until c):
 - a) with prob. p **out**edge
 - b) with prob. q **in**edge
 - c) with prob. $1-p-q$ next iter.In 2. other end-point chosen prop. to **in**/**out**degree $+\gamma$.

Modeling Complex Networks

Preferential Attachment [Mitzenmacher '01]

Parameters: n - # vertices γ - "self-loop"

p, q ($p+q < 1$) - probability of **out**/**in** edge



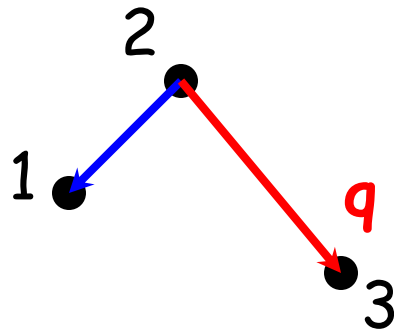
1. Start with a single vertex
2. In iteration $i=2, \dots, n$ create edges between vtx i and $<i$. Repeat until c):
 - a) with prob. p **out**edge
 - b) with prob. q **in**edge
 - c) with prob. $1-p-q$ next iter.In 2. other end-point chosen prop. to **in**/**out**degree $+\gamma$.

Modeling Complex Networks

Preferential Attachment [Mitzenmacher '01]

Parameters: n - # vertices γ - "self-loop"

p, q ($p+q < 1$) - probability of **out**/**in** edge



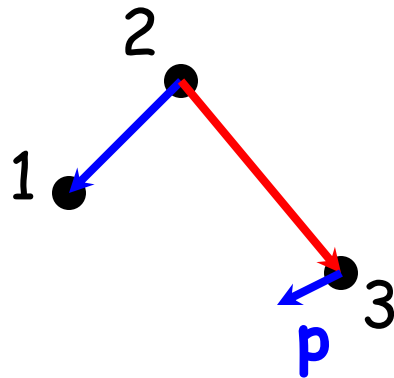
1. Start with a single vertex
2. In iteration $i=2, \dots, n$ create edges between vtx i and $<i$. Repeat until c):
 - a) with prob. p outedge
 - b) with prob. q inedge
 - c) with prob. $1-p-q$ next iter.In 2. other end-point chosen prop. to **in**/**out**degree $+\gamma$.

Modeling Complex Networks

Preferential Attachment [Mitzenmacher '01]

Parameters: n - # vertices γ - "self-loop"

p, q ($p+q < 1$) - probability of **out**/**in** edge



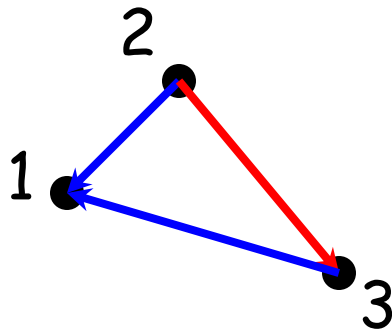
1. Start with a single vertex
2. In iteration $i=2, \dots, n$ create edges between vtx i and $<i$. Repeat until c):
 - a) with prob. p outedge
 - b) with prob. q inedge
 - c) with prob. $1-p-q$ next iter.In 2. other end-point chosen prop. to **in**/**out**degree $+\gamma$.

Modeling Complex Networks

Preferential Attachment [Mitzenmacher '01]

Parameters: n - # vertices γ - "self-loop"

p, q ($p+q < 1$) - probability of **out**/**in** edge



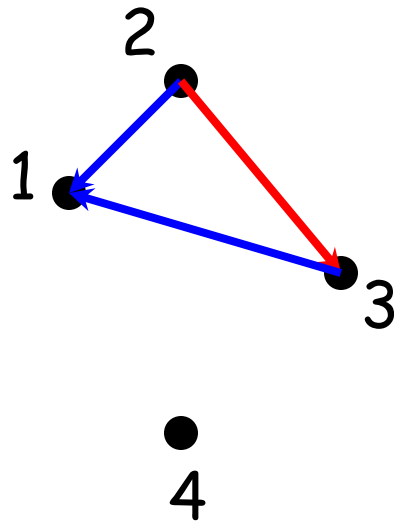
1. Start with a single vertex
2. In iteration $i=2, \dots, n$ create edges between vtx i and $<i$. Repeat until c):
 - a) with prob. p **out**edge
 - b) with prob. q **in**edge
 - c) with prob. $1-p-q$ next iter.In 2. other end-point chosen prop. to **in**/**out**degree $+\gamma$.

Modeling Complex Networks

Preferential Attachment [Mitzenmacher '01]

Parameters: n - # vertices γ - "self-loop"

p, q ($p+q < 1$) - probability of **out**/**in** edge



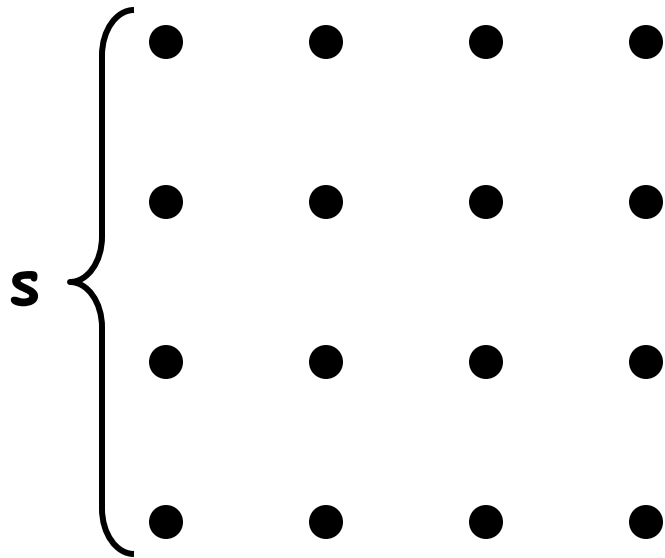
1. Start with a single vertex
2. In iteration $i=2, \dots, n$ create edges between vtx i and $<i$. Repeat until c):
 - a) with prob. p **out**edge
 - b) with prob. q **in**edge
 - c) with prob. $1-p-q$ next iter.In 2. other end-point chosen prop. to **in**/**out**degree $+\gamma$.

Modeling Complex Networks

Small World [Watts-Strogatz '98, Kleinberg '00]

Parameters: s - side of the grid

α, β - determine distribution on edges



1. Arrange vertices in $s \times s$ grid.
2. Add an edge from u to v with probability

$$\alpha \text{dist}(u, v)^{-\beta}$$

where $\text{dist}(u, v)$ is the Manhattan distance from u to v .

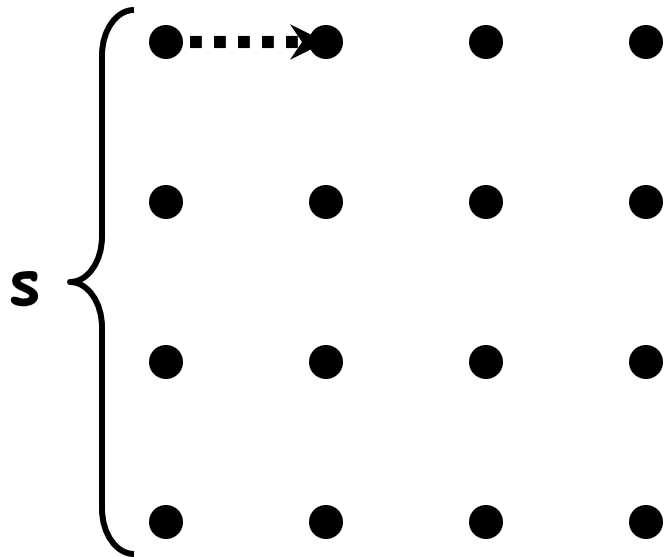
3. Omit isolated vertices.

Modeling Complex Networks

Small World [Watts-Strogatz '98, Kleinberg '00]

Parameters: s - side of the grid

α, β - determine distribution on edges



1. Arrange vertices in $s \times s$ grid.
2. Add an edge from u to v with probability

$$\alpha \text{dist}(u, v)^{-\beta}$$

where $\text{dist}(u, v)$ is the Manhattan distance from u to v .

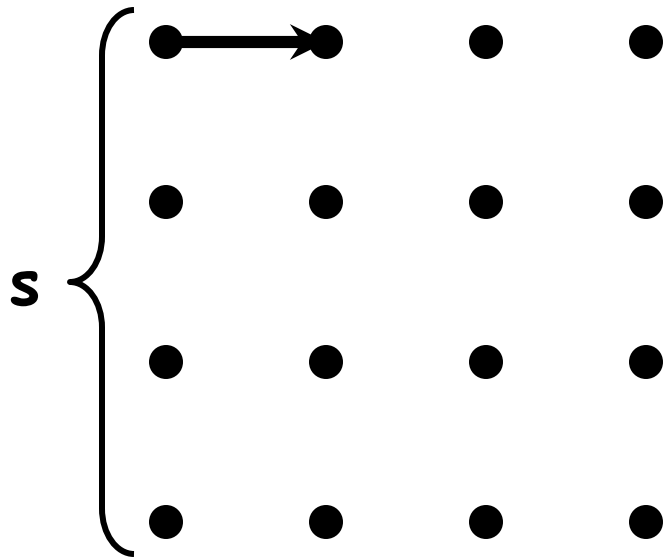
3. Omit isolated vertices.

Modeling Complex Networks

Small World [Watts-Strogatz '98, Kleinberg '00]

Parameters: s - side of the grid

α, β - determine distribution on edges



1. Arrange vertices in $s \times s$ grid.
2. Add an edge from u to v with probability

$$\alpha \text{dist}(u, v)^{-\beta}$$

where $\text{dist}(u, v)$ is the Manhattan distance from u to v .

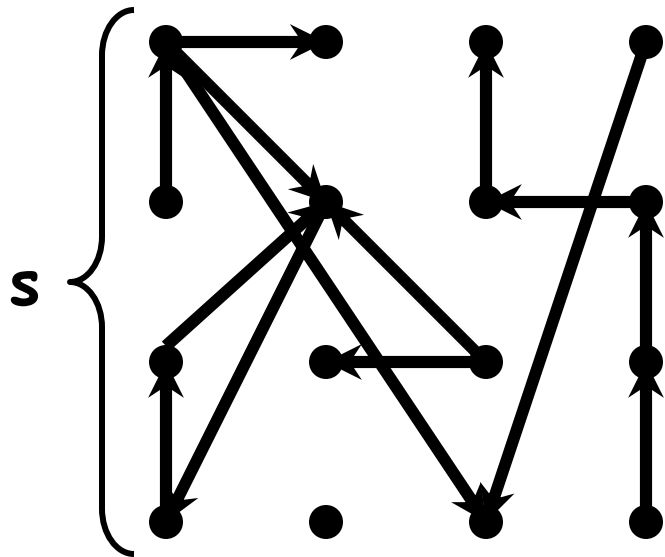
3. Omit isolated vertices.

Modeling Complex Networks

Small World [Watts-Strogatz '98, Kleinberg '00]

Parameters: s - side of the grid

α, β - determine distribution on edges



1. Arrange vertices in $s \times s$ grid.
2. Add an edge from u to v with probability

$$\alpha \text{dist}(u, v)^{-\beta}$$

where $\text{dist}(u, v)$ is the Manhattan distance from u to v .

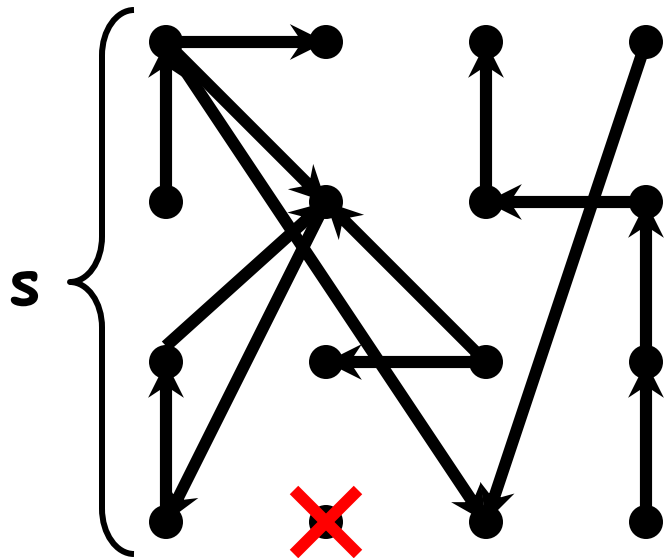
3. Omit isolated vertices.

Modeling Complex Networks

Small World [Watts-Strogatz '98, Kleinberg '00]

Parameters: s - side of the grid

α, β - determine distribution on edges



1. Arrange vertices in $s \times s$ grid.
2. Add an edge from u to v with probability

$$\alpha \text{dist}(u, v)^{-\beta}$$

where $\text{dist}(u, v)$ is the Manhattan distance from u to v .

3. Omit isolated vertices.

Previous Work: Scoring Graph Models

Score by model's capability to reproduce certain properties.

[Medina - Matta - Byers '00]

[Bu - Towsley '02]

[Barabási - Albert - Jeong '00]

"Closing-the-loop" approach

[Willinger - Govindan - Jamin - Paxson - Shenker '02]

[Chen - Chang - Govindan - Jamin - Shenker - Willinger '02]

"Performance" and "Likelihood" metrics

[Li - Alderson - Willinger - Doyle '04]

Previous Work: Scoring Graph Models

Score by model's capability to reproduce certain properties.

[Medina - Matta - Byers '00]

[Bu - Towsley '02]

[Barabási - Albert - Jeong '00]

"Closing-the-loop" approach

[Willinger - Govindan - Jamin - Paxson - Shenker '02]

[Chen - Chang - Govindan - Jamin - Shenker - Willinger '02]

"Performance" and "Likelihood" metrics

[Li - Alderson - Willinger - Doyle '04]

probability of a graph being generated by a specific powerlaw random graph model.

Our Approach: Maximum Likelihood (ML)

$$\text{Score (model)} = -\log (\text{prob. model generates } G)$$

What: scoring random graph models wrt a given graph G

Why ML?

- Kolmogorov complexity inspired (Kolmogorov, Solomonoff, Chaitin, Levin 1960s)
- Minimum Description Length Principle (MDL)

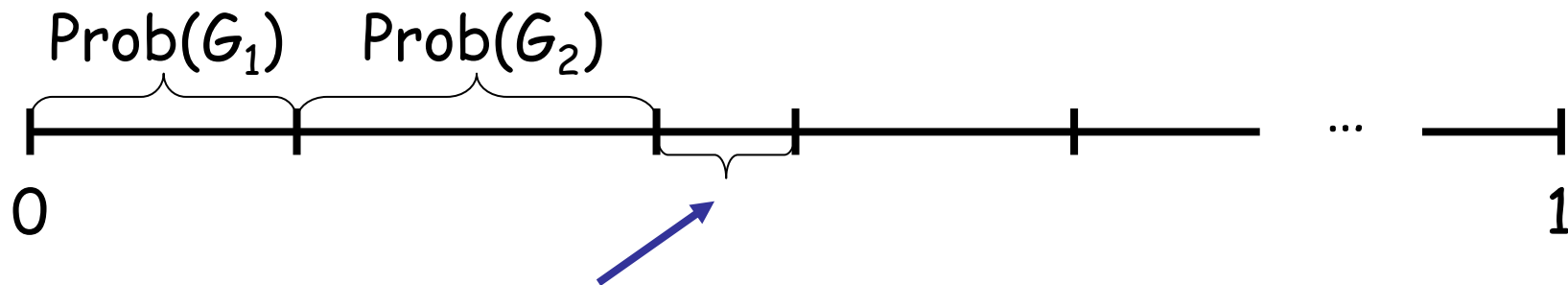
Our Approach: Maximum Likelihood (ML)

$$\text{Score (model)} = -\log (\text{prob. model generates } G)$$

+ model description length and parameter setting

Related to **Minimum Description Length Principle (MDL)**

- consider a fixed order of all possible graphs $G: G_1, G_2, \dots$



Suppose this interval is of length $1/16$. Then it contains one of the numbers $0/16, 1/16, 2/16, \dots, 16/16$. Thus, we can encode the interval by that number, using ~ 4 bits = $-\log(1/16)$.

Our Approach: Maximum Likelihood (ML)

$$\text{Score (model)} = -\log (\text{prob. model generates } \mathcal{G})$$

Technical issues:

- a model must be able to generate every graph.
- node ordering
- how to compute ?

Our Approach: Maximum Likelihood (ML)

$$\text{Score (model)} = -\log (\text{prob. model generates } \mathcal{G})$$

Technical issues:

- a model must be able to generate every graph.
 - node ordering
 - how to compute ?
- Node labels: a random permutation of $\{1, \dots, n\}$.

$$\text{Score (model)} =$$

$$-\log \frac{1}{n!} \sum_{\pi} (\text{prob. model generates } \mathcal{G} \text{ w. labels } \pi)$$

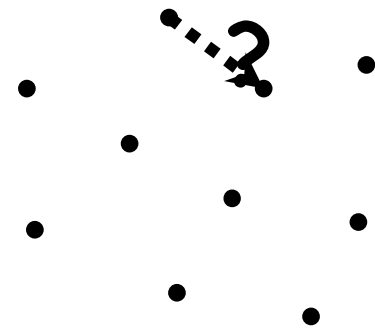
Algorithms: Warm-up

Erdős-Rényi

$$\text{Prob}_{\text{ER}} (G) = p^m(1-p)^{n(n-1)-m}$$

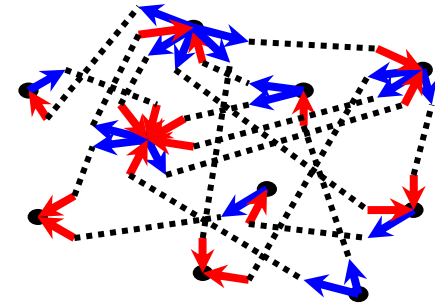
where m is the number of edges and n is the number of vertices of G .

$\text{Prob}_{\text{ER}} (G)$ is maximized for $p = m/(n(n-1))$.



Algorithms: Warm-up cont'

Powerlaw Random Graph (PRG)



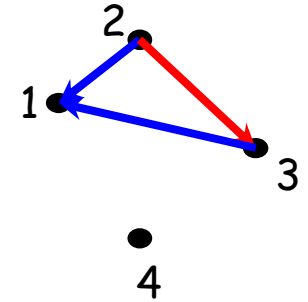
$$\text{Prob}_{\text{PRG}} (G) = \frac{\prod_v \text{Prob}(\text{in}(v)) \cdot \text{Prob}(\text{out}(v)) \cdot \text{in}(v)! \cdot \text{out}(v)!}{m!}$$

where $\text{in}(v)$ and $\text{out}(v)$ are the indegree and outdegree of a vertex v and

$$\text{Prob}(\text{in}(v)) = \text{in}(v)^{-\beta_{\text{in}}} / Z_{\text{in}} \quad (\text{similarly } \text{Prob}(\text{out}(v)))$$
$$Z_{\text{in}} = \sum_{k=1..c_{\text{in}}} k^{-\beta_{\text{in}}}$$

Note: the formula assumes a simple graph G (can be easily modified for non-simple graphs).

Algorithms: MCMC approach



Preferential Attachment (PA)

Given a vertex labeling π (vertices labeled 1..n) :

$$\text{Prob}_{\text{PA}} (G \mid \pi) = \prod_{i=2 \dots n} (\text{in}(i) + \text{out}(i))! (1 - p - q) \cdot \prod_{j < i, (j, i) \in E} p (\text{in}_i(j) + \gamma) / (m_i) \cdot \prod_{j < i, (i, j) \in E} q (\text{out}_i(j) + \gamma) / (m_i)$$

where $\text{in}_i(j)$ is the number of in-neighbors of vertex j labeled $< i$ (similarly $\text{out}_i(j)$), and m_i is the normalizing factor

$$m_i = \sum_{k=1 \dots i-1} (\text{in}_i(k) + \gamma) = \sum_{k=1 \dots i-1} (\text{out}_i(k) + \gamma)$$

Algorithms: MCMC approach

Preferential Attachment (PA)

Bottom line: $\text{Prob}_{PA}(G|\pi)$ (relatively) easy to compute

The problem: average over all π

Score (model) =

$$-\log \frac{1}{n!} \sum_{\pi} (\text{prob. model generates } G \text{ w. labels } \pi)$$

Algorithms: MCMC approach

Preferential Attachment (PA)

Bottom line: $\text{Prob}_{PA}(G|\pi)$ (relatively) easy to compute

Goal: compute $-\log(\sum_{\pi} \text{Prob}_{PA}(G|\pi))$

Idea: design a Markov chain on all labelings where a labeling ω is sampled with probability proportional to $\text{Prob}_{PA}(G|\omega)$

$$\sigma(\omega) = \text{Prob}_{PA}(G|\omega) / \sum_{\pi} \text{Prob}_{PA}(G|\pi)$$

How does the MC help?

Algorithms: MCMC approach

Preferential Attachment (PA)

Assume that we can sample labelings with probability:

$$\sigma(\omega) = \text{Prob}_{PA}(G|\omega) / \sum_{\pi} \text{Prob}_{PA}(G|\pi)$$

A standard trick to compute the "partition function":

$$\text{Prob}_{PA}(G|\{1,2,3,\dots,n\}) / \sum_{\pi} \text{Prob}_{PA}(G|\pi) =$$

$$\frac{\sum_{\pi \text{ starting with } 1} \text{Prob}_{PA}(G|\pi)}{\sum_{\pi} \text{Prob}_{PA}(G|\pi)} \cdot \frac{\sum_{\pi \text{ starting with } 1,2} \text{Prob}_{PA}(G|\pi)}{\sum_{\pi \text{ starting with } 1} \text{Prob}_{PA}(G|\pi)} \dots$$

Algorithms: MCMC approach

Preferential Attachment (PA)

The main point: design an efficient MC with stationary distribution proportional to $\text{Prob}_{PA}(G|\omega)$

MC on permutations:

$$\omega = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 5 & 2 & 7 & 3 & 1 & 8 & 6 & 4 \\ \hline \end{array}$$

A swapping chain ?

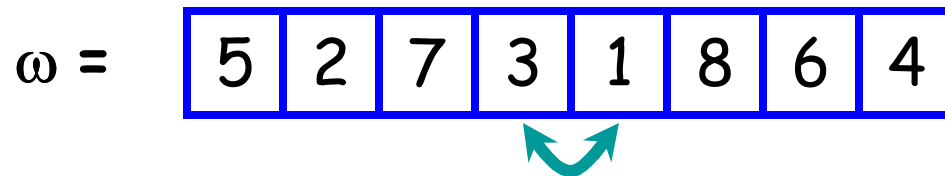
(+ Metropolis filter to get the stationary distribution)

Algorithms: MCMC approach

Preferential Attachment (PA)

The main point: design an efficient MC with stationary distribution proportional to $\text{Prob}_{PA}(G|\omega)$

MC on permutations:



A swapping chain ?

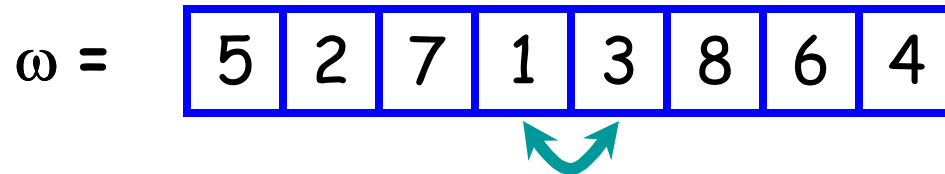
(+ Metropolis filter to get the stationary distribution)

Algorithms: MCMC approach

Preferential Attachment (PA)

The main point: design an efficient MC with stationary distribution proportional to $\text{Prob}_{PA}(G|\omega)$

MC on permutations:



A swapping chain ?

(+ Metropolis filter to get the stationary distribution)

Algorithms: MCMC approach

Preferential Attachment (PA)

The main point: design an efficient MC with stationary distribution proportional to $\text{Prob}_{PA}(G|\omega)$

MC on permutations:

$$\omega = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 5 & 2 & 7 & 1 & 3 & 8 & 6 & 4 \\ \hline \end{array}$$

~~A swapping chain?~~ Hit-and-run.

(+ Metropolis filter to get the stationary distribution)

Algorithms: MCMC approach

Preferential Attachment (PA)

The main point: design an efficient MC with stationary distribution proportional to $\text{Prob}_{PA}(G|\omega)$

MC on permutations:

$\omega =$

5	2	7	1	3	8	6	4
---	---	---	---	---	---	---	---

~~A swapping chain?~~ Hit-and-run.

(+ Metropolis filter to get the stationary distribution)

Algorithms: MCMC approach

Preferential Attachment (PA)

The main point: design an efficient MC with stationary distribution proportional to $\text{Prob}_{PA}(G|\omega)$

MC on permutations:

$$\omega = \begin{array}{|c|c|c|c|c|c|c|} \hline 5 & 2 & 7 & 3 & 8 & 6 & 4 \\ \hline \end{array}$$

$\begin{array}{|c|} \hline 1 \\ \hline \end{array}$

~~A swapping chain?~~ Hit-and-run.

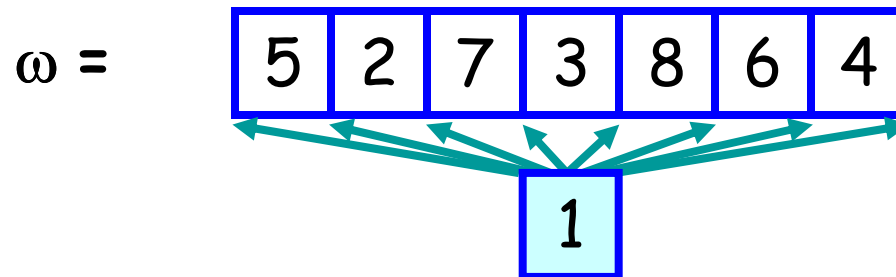
(+ Metropolis filter to get the stationary distribution)

Algorithms: MCMC approach

Preferential Attachment (PA)

The main point: design an efficient MC with stationary distribution proportional to $\text{Prob}_{PA}(G|\omega)$

MC on permutations:



~~A swapping chain?~~ Hit-and-run.

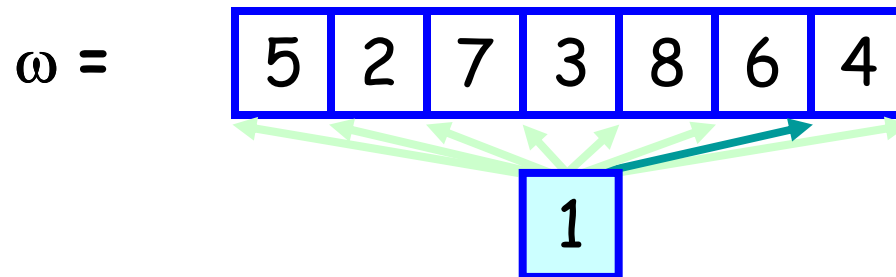
(+ Metropolis filter to get the stationary distribution)

Algorithms: MCMC approach

Preferential Attachment (PA)

The main point: design an efficient MC with stationary distribution proportional to $\text{Prob}_{PA}(G|\omega)$

MC on permutations:



~~A swapping chain?~~ Hit-and-run.

(+ Metropolis filter to get the stationary distribution)

Algorithms: MCMC approach

Preferential Attachment (PA)

The main point: design an efficient MC with stationary distribution proportional to $\text{Prob}_{PA}(G|\omega)$

MC on permutations:

$\omega =$

5	2	7	3	8	6	1	4
---	---	---	---	---	---	---	---

~~A swapping chain?~~ Hit-and-run.

(+ Metropolis filter to get the stationary distribution)

Algorithms: MCMC approach

Preferential Attachment (PA)

The main point: design an efficient MC with stationary distribution proportional to $\text{Prob}_{PA}(G|\omega)$

MC on permutations:

$$\omega = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 5 & 2 & 7 & 3 & 8 & 6 & 1 & 4 \\ \hline \end{array}$$

~~A swapping chain?~~ Hit-and-run.

(+ Metropolis filter to get the stationary distribution)

Algorithms: MCMC approach

Preferential Attachment (PA)

The main point: design an efficient MC with stationary distribution proportional to $\text{Prob}_{PA}(G|\omega)$

MC on permutations:

$$\omega = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 5 & 2 & 7 & 3 & 8 & 6 & 1 & 4 \\ \hline \end{array}$$

Metropolis filter to get the stationary distribution σ :

the "uniform chain" goes from ω_1 to ω_2 , we make the move with probability $\min\{ 1, \sigma(\omega_1)/\sigma(\omega_2) \}$

(in our case = $\min\{ 1, \text{Prob}_{PA}(G|\omega_1) / \text{Prob}_{PA}(G|\omega_2) \}$)

Algorithms: MCMC approach

Preferential Attachment (PA)

The main point: design an efficient MC with stationary distribution proportional to $\text{Prob}_{PA}(G|\omega)$

MC on permutations:

$$\omega = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 5 & 2 & 7 & 3 & 8 & 6 & 1 & 4 \\ \hline \end{array}$$

Metropolis filter & hit-and-run:

in ω_1 , choose a position (n choices), then move with probability $\min\{1, \sigma(\omega_1)/\sigma(\omega_2)\}$

(overall probability of move is $\min\{1, \sigma(\omega_1)/\sigma(\omega_2)\} / n$)

Algorithms: MCMC approach

Preferential Attachment (PA)

The main point: design an efficient MC with stationary distribution proportional to $\text{Prob}_{PA}(G|\omega)$

MC on permutations:

$$\omega = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 5 & 2 & 7 & 3 & 8 & 6 & 1 & 4 \\ \hline \end{array}$$

Metropolis filter & hit-and-run (improved/faster):

in ω_1 , choose a position (n choices) proportionally to $\sigma(\omega_2)$

(does not "waste" any steps of the MC)

Algorithms: MCMC approach

Preferential Attachment (PA)

The main point: design an efficient MC with stationary distribution proportional to $\text{Prob}_{PA}(G|\omega)$

MC on permutations:

$$\omega = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 5 & 2 & 7 & 3 & 8 & 6 & 1 & 4 \\ \hline \end{array}$$

Hit-and-run summary:

- have to be careful, depends on the distribution
- still quite slow: massive parallelism

Algorithms: MCMC approach

Preferential Attachment (PA)

The main point: design an efficient MC with stationary distribution proportional to $\text{Prob}_{PA}(G|\omega)$

MC on permutations:

$\omega =$

5	2	7	3	8	6	1	4
---	---	---	---	---	---	---	---

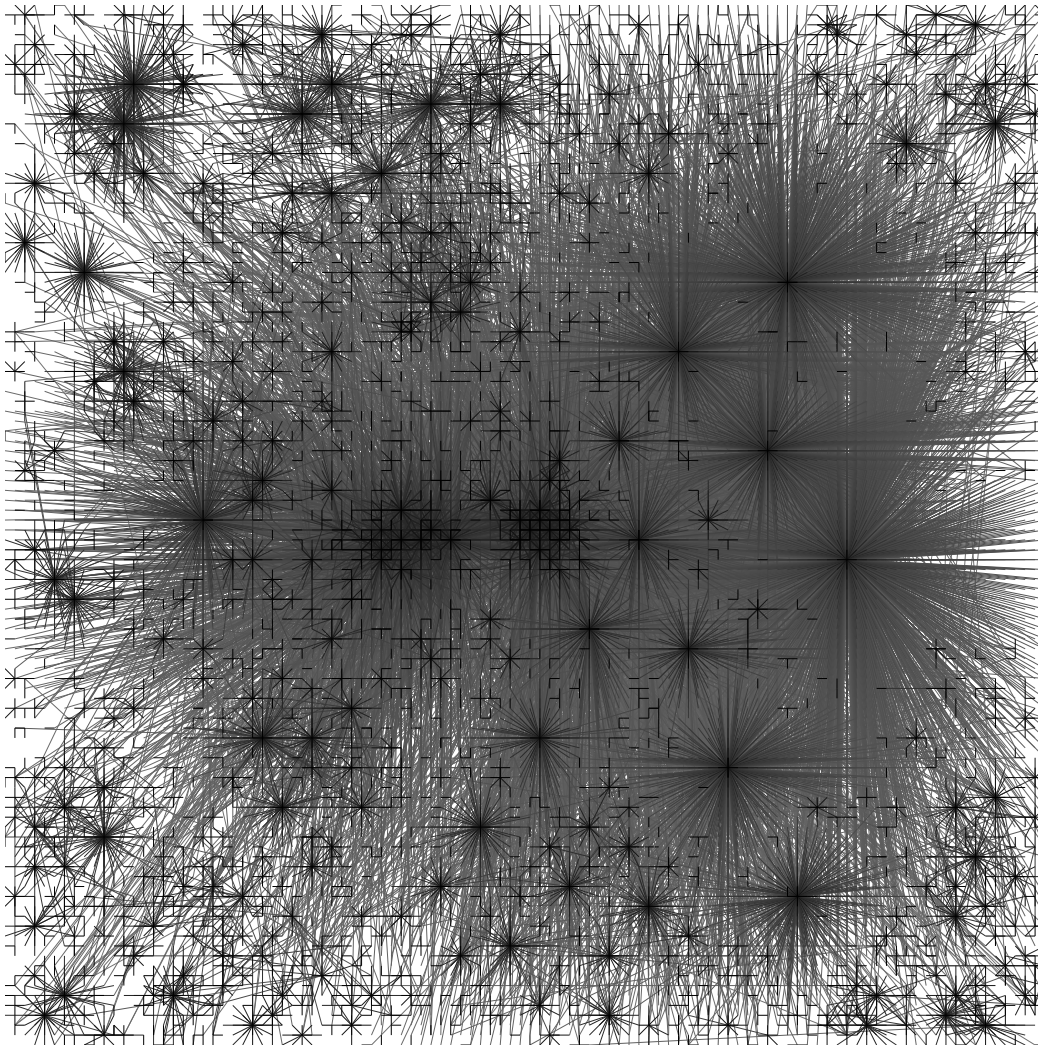
Hit-and-run summary:

- have to be careful, depends on the distribution
- still quite slow: massive parallelism

e.g. does not work for the small world model - we use simulated annealing (heuristically) to determine bounds on $\text{Prob}_{SW}(G)$

Experiments

AS-level Internet topology: three snapshots '97, '99, '01

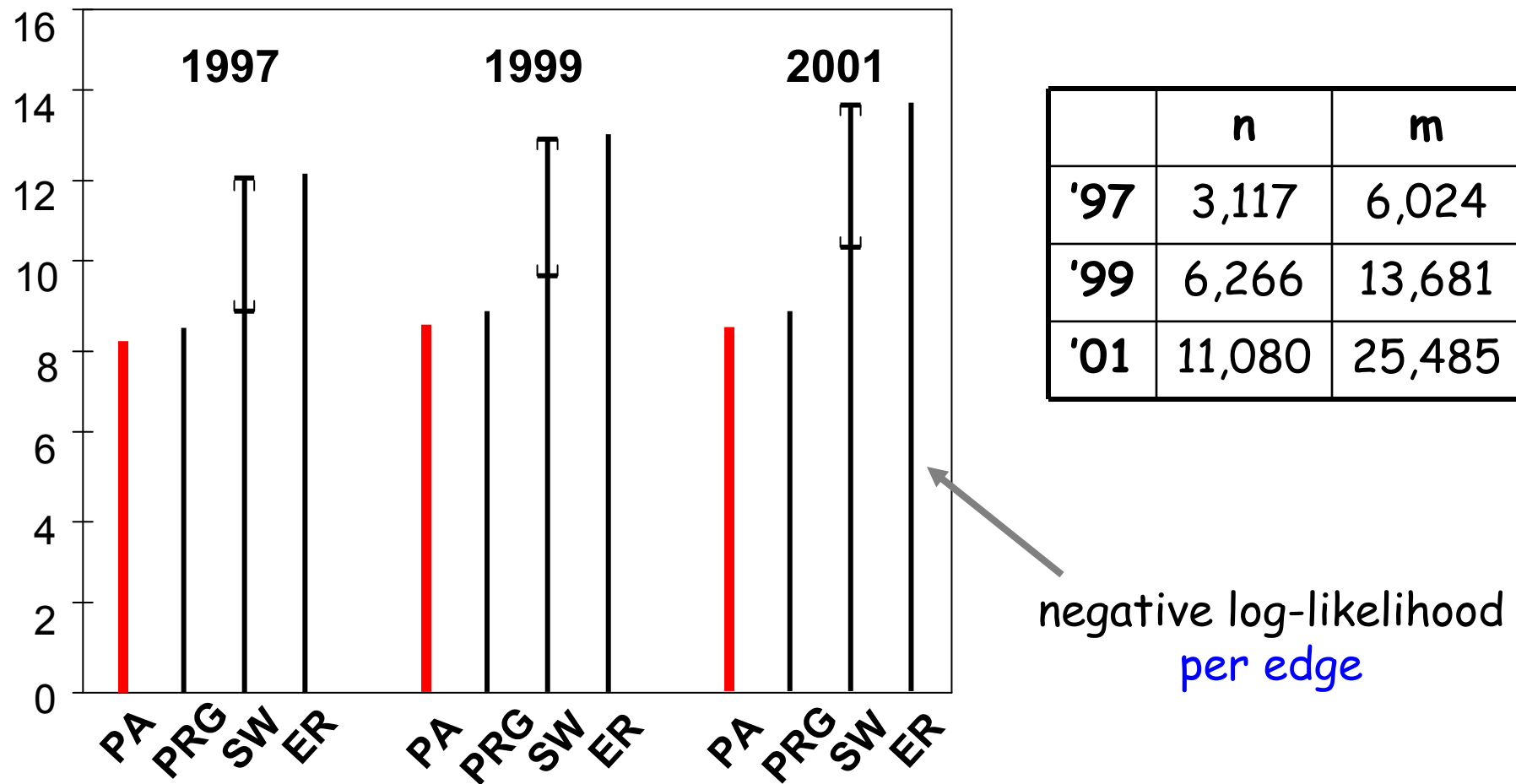


	n	m
'97	3,117	6,024
'99	6,266	13,681
'01	11,080	25,485



Experiments

AS-level Internet topology: three snapshots '97, '99, '01

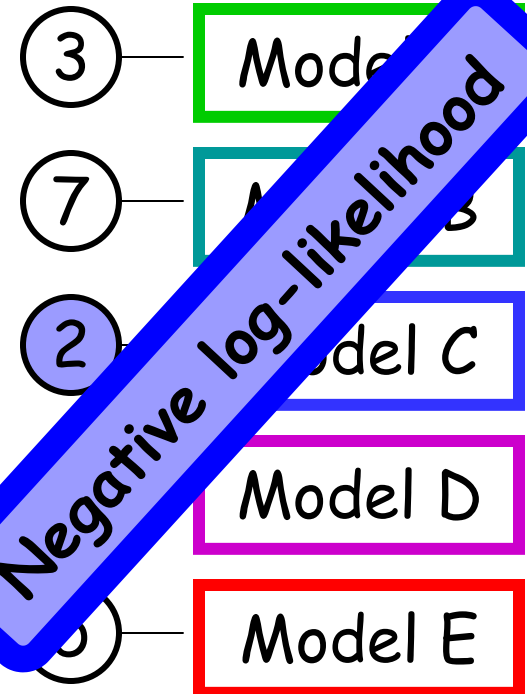


Experiments

	PA	PRG	SW	ER
'97	8.30	8.60	8.96	12.10
	$p = 0.58$ $q = 0.08$ $\gamma = 0.5$	$\beta_{in} = 1.55, c_{in} = 610$ $\beta_{out} = 2.39, c_{out} = 69$	$\alpha = 0.111$ $\beta = 1.9$	$p = 6.2e-4$
'99	8.55	8.83	9.76	12.93
	$p = 0.61$ $q = 0.08$ $\gamma = 0.4$	$\beta_{in} = 1.57, c_{in} = 1410$ $\beta_{out} = 2.44, c_{out} = 172$	$\alpha = 0.092$ $\beta = 1.8$	$p = 3.5e-4$
'01	8.58	8.85	10.42	13.68
	$p = 0.63$ $q = 0.07$ $\gamma = 0.3$	$\beta_{in} = 1.57, c_{in} = 2421$ $\beta_{out} = 2.50, c_{out} = 214$	$\alpha = 0.088$ $\beta = 1.8$	$p = 2.1e-4$

Summary

- objective ranking mechanism
- algorithms
- experiments



Open Directions

- design new models
- more general / faster algorithms
- experiments and implications in other contexts

