The *h*-index of a graph and its application to dynamic subgraph statistics

David Eppstein
Univ. of California, Irvine
Computer Science Department

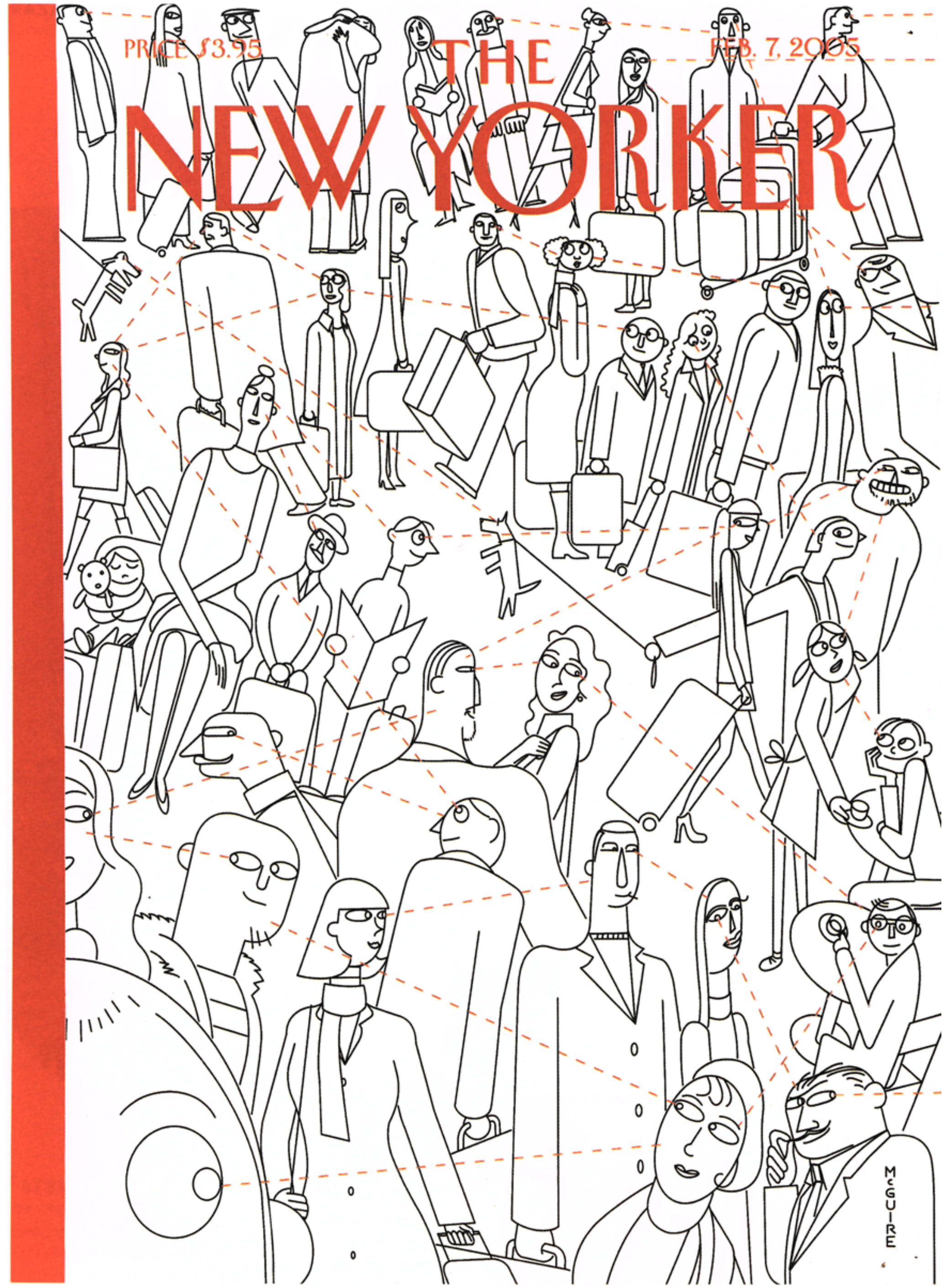
Emma S. Spiro
Univ. of California, Irvine
Department of Sociology

Context: Analysis of Social Networks

Represent interactions among people and their environments as graphs

There are many different kinds of social networks, with different data analysis challenges

Goal: develop mathematical models that are general enough to handle this heterogeneity and accurate enough to give us interesting predictions



Examples of social networks:

Real-life personal or sexual contacts

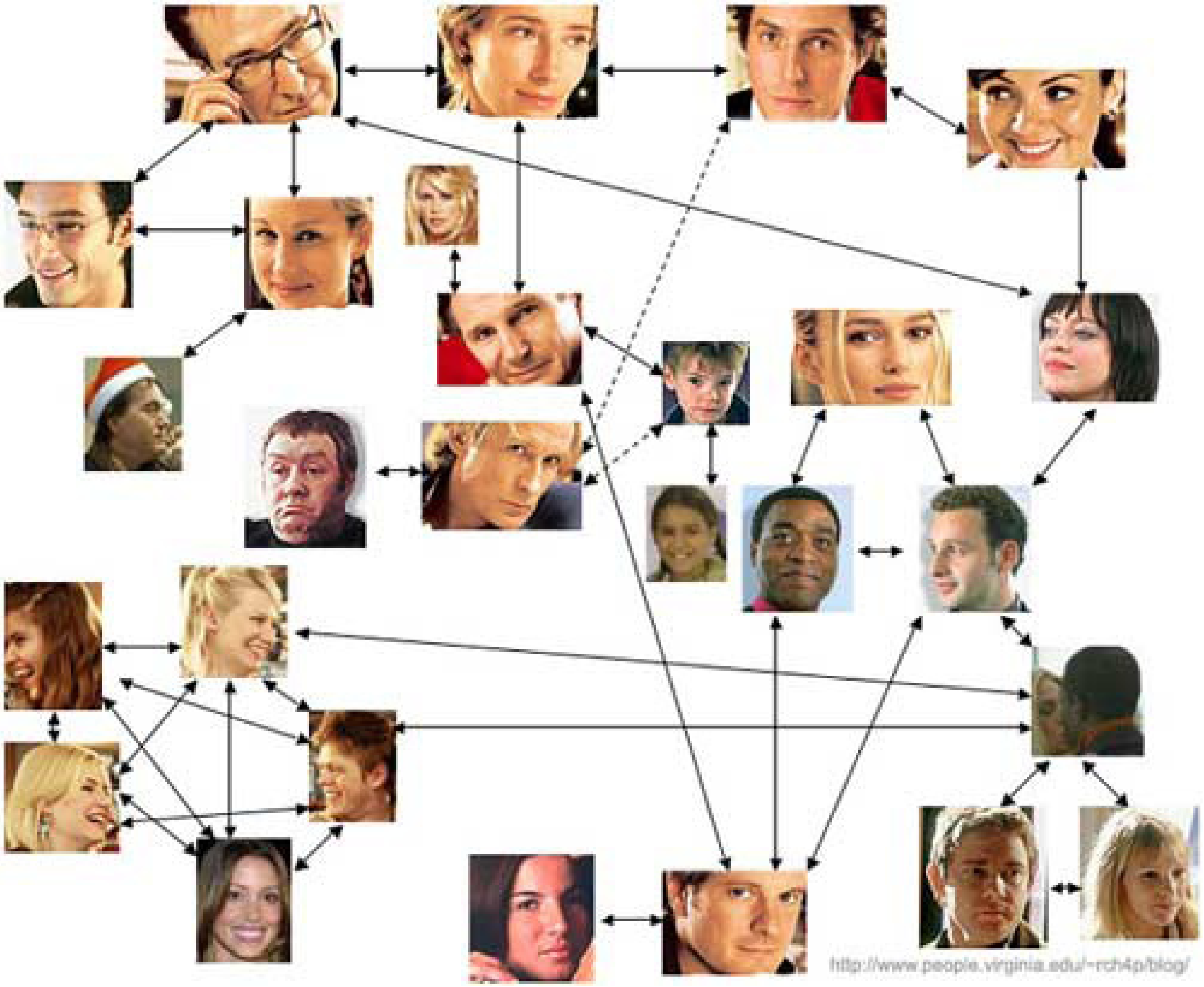


Illustration of contacts from the movie *Love, Actually*

Vertices = people

Edges = contacts

Graphs are small, difficult to obtain, and noisy

Structure depends on vertex/edge labels
(e.g. M-F sexual contact more frequent than M-M or F-F)

Examples of social networks:

On-line social networks such as LiveJournal

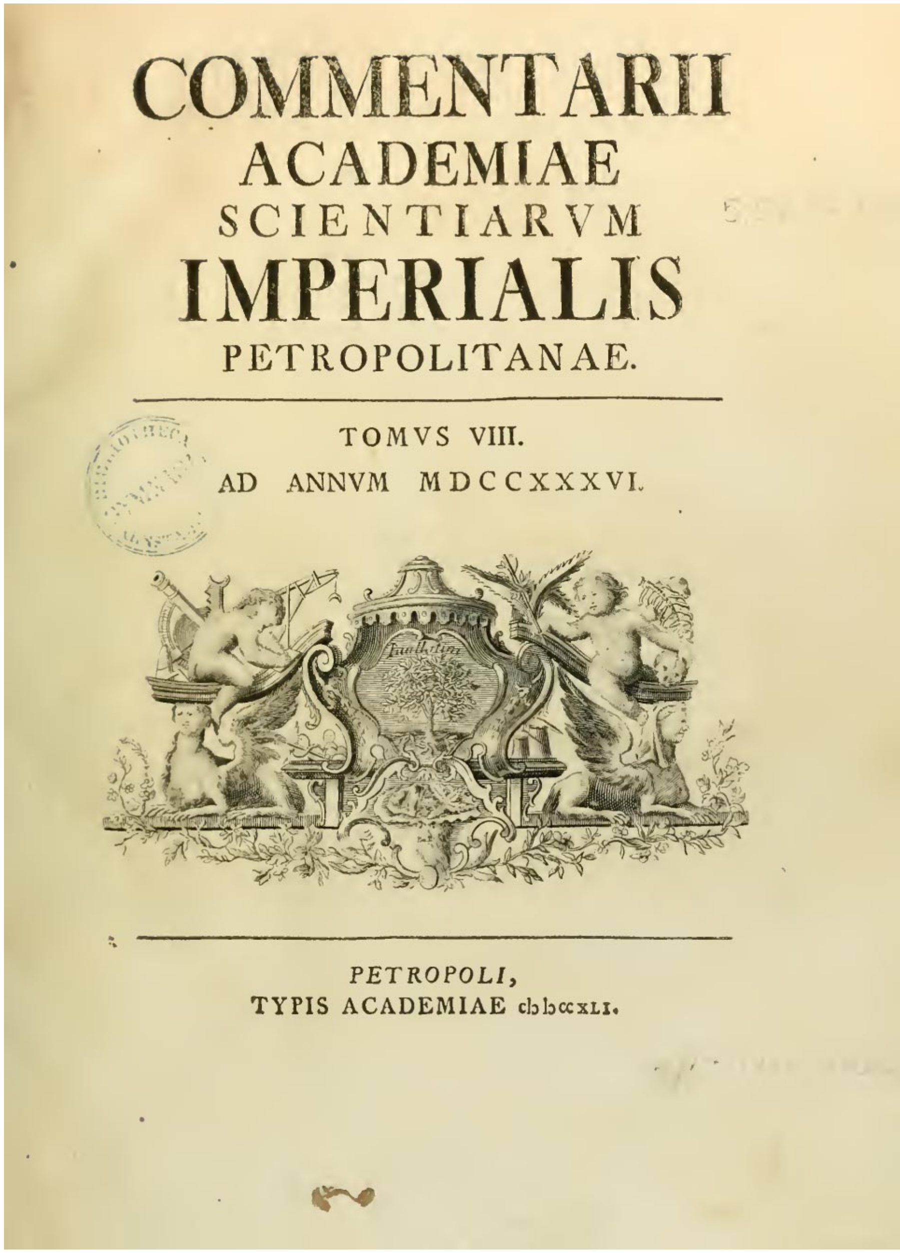
Vertices = online identities
(not 1-1 with people)

Edges = "friends"
(two meanings on LiveJournal: people whose entries one reads, and people with permission to read one's semi-private entries)

Graphs are large,
easy to obtain,
and heterogeneous
(many subcommunities with different connection patterns)

LiveJournal connections for mcfnord, from ljmindmap.com

Examples of social networks:
Scientific publication databases


COMMENTARII
ACADEMIAE
SCIENTIARVM
IMPERIALIS
PETROPOLITANAE.

TOMVS VIII.
AD ANNVM MDCCXXXVI.

PETROPOLI,
TYPIS ACADEMIAE cIↄIↄccxLI.

Two kinds of vertices,
authors and publications

Two kinds of edges,
authorship and citation

Graphs are large,
not hard to obtain,
but noisy

(difficulty: determining
when two similarly named
entities are the same)

The journal containing Euler's original graph theory paper

Exponential random graph model: graphs shaped by their local structures

Fix a set of vertices

Determine local features
- Presence of an edge
- Degree of a vertex
- Small subgraphs

Assign weights to features: positive = more likely, negative = less likely

Log-likelihood of G = sum of weights of features + normalizing constant

Different feature sets and weights give different models capable of fitting different types of social network

Public-domain image by Mohylek on Wikimedia commons, http://commons.wikimedia.org/wiki/File:Magnifying.jpg

Probabilistic reasoning in exponential random graphs

Most basic problem: pull the handle, generate a random graph from the model

Crop of CC-BY-SA licensed image "Slot Machine" by Jeff Kubina on Flickr, http://www.flickr.com/photos/95118988@N00/347687569

With a generation subroutine, we can also:

- Find normalizing constant
- Fit weights to data
- Understand typical behavior of graphs in this model (e.g. how many edges?)
- Detect unusual structures in real-world graphs

Standard method for random generation: Markov Chain Monte Carlo (random walk)

Idea: start with any graph

Repeatedly choose a random edge to add or remove

Choose whether to perform that update based on its effect on log-likelihood

After enough steps, graph is random with correct probability distribution

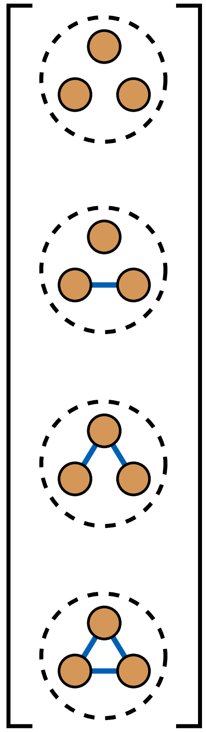
Key subproblem:
Maintain feature counts for a dynamically changing graph

“The Mambo”, public artwork by Jack Mackie and Chuck Greening, Seattle, 1979. Modified from GFDL-licensed photo by Joe Mabel on Wikimedia Commons, http://commons.wikimedia.org/wiki/File:Seattle_B%27way_Mambo_02.jpg

Assumption: feature = small induced subgraph

Feature counts can be related to other more easily-counted quantities:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$= \begin{bmatrix} n(n-1)(n-2)/6 \\ m(n-2) \\ \Sigma \deg(v)\,(\deg(v)-1)/2 \\ \text{number of triangles} \end{bmatrix}$$

So if we can count triangles in a dynamic graph
we can maintain all other possible 3-vertex feature counts

Main ideas of triangle-counting data structure (I)

Select a number D

Partition vertices into two subsets:

L: many vertices with degree less than D

H: few vertices with degree greater than D

Boys choosing sides for hockey on Sarnia Bay, Ontario, December 29, 1908. Public domain image from Library and Archives Canada / John Boyd Collection / PA-060732
http://www.collectionscanada.gc.ca/hockey/024002-2300-e.html

Main ideas of triangle-counting data structure (II)

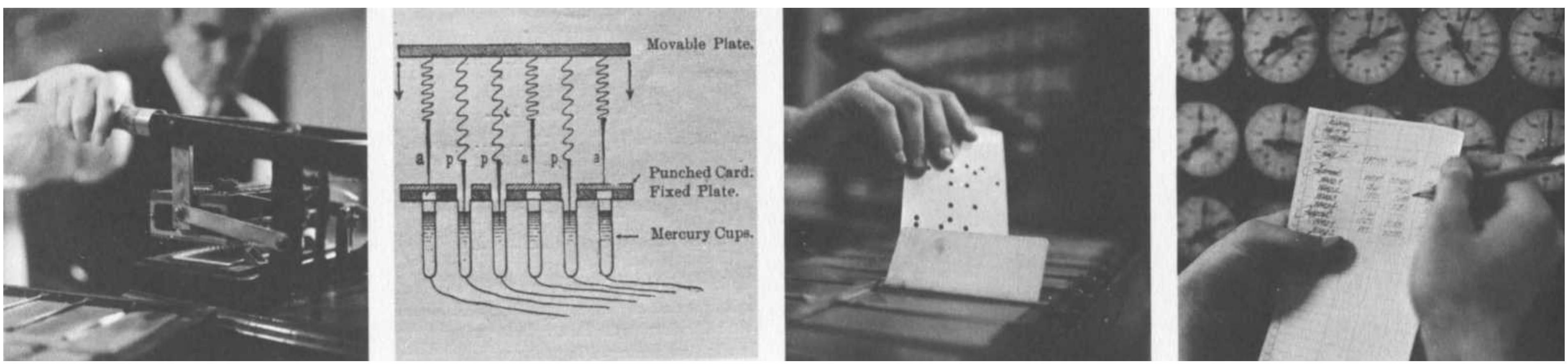
Maintain hash table C indexed by pairs (*u*,*v*) of vertices

C[*u*,*v*] = number of two-edge paths *u*—L—*v*

To count triangles involving an updated edge:

Look up its endpoints in C to find triangles with third point in L

Test each vertex in H to find triangles with third point in H



Hollerith 1890 census tabulator from http://www.columbia.edu/acis/history/census-tabulator.html

How much time does it take per change?

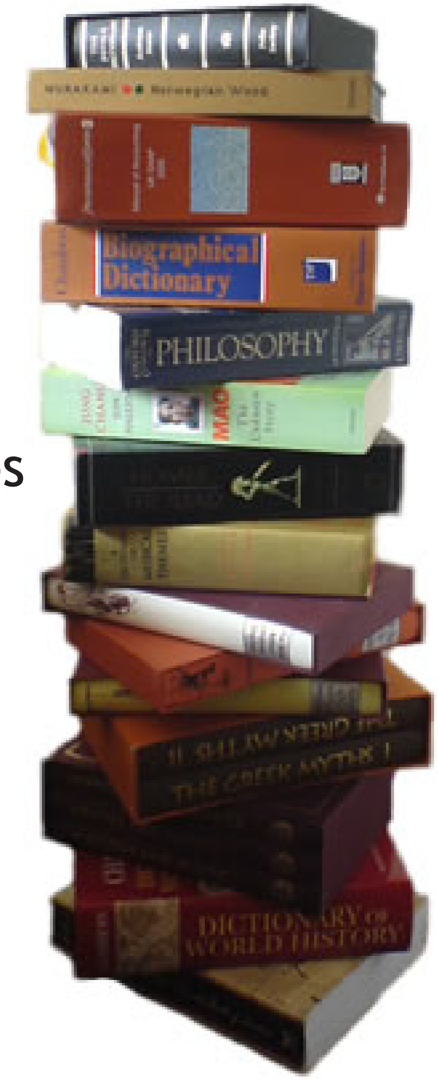
Finding triangles involving changed edge takes **O(|H|)**

Each edge is involved in O(D) x—L—x paths, so
updating hash table after a change takes **O(D)**

If L/H partition ever changes, update counts
for all x—L—x paths through moved vertex
taking time **$O(D^2)$**

**How to choose D so |H| + D is small
and partition changes infrequently?**

A detour into bibliometrics

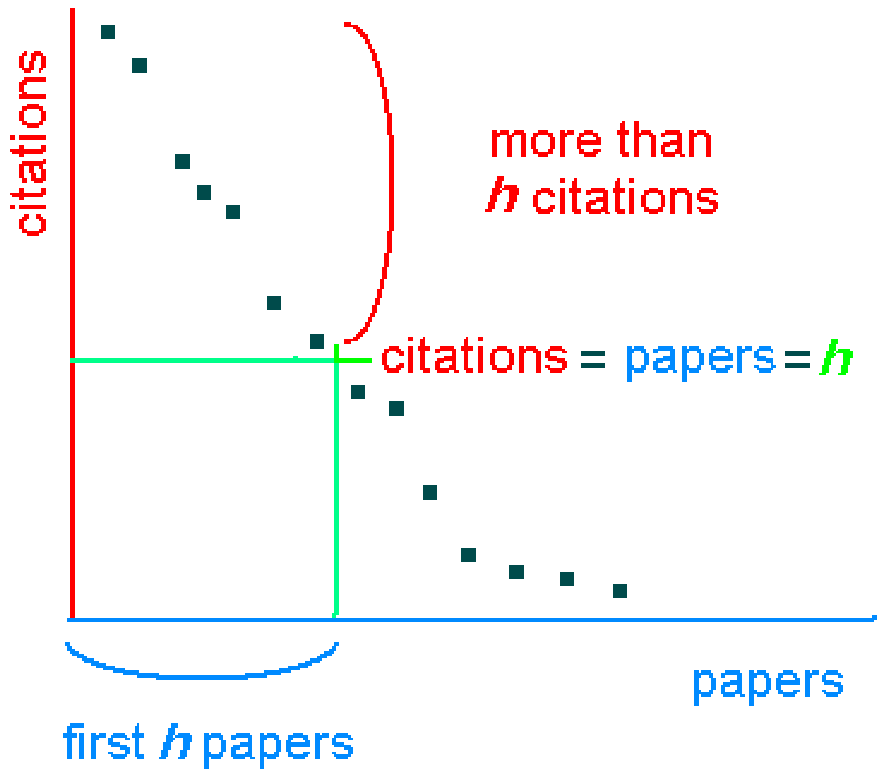


How to measure productivity of an academic researcher?

Total publication count: encourages many low-impact papers

Total citation count: unduly influenced by few high-impact pubs

h-index [J. E. Hirsch, PNAS 2005]:
maximum number such that *h* papers each have ≥ *h* citations



The *h*-index of a graph:

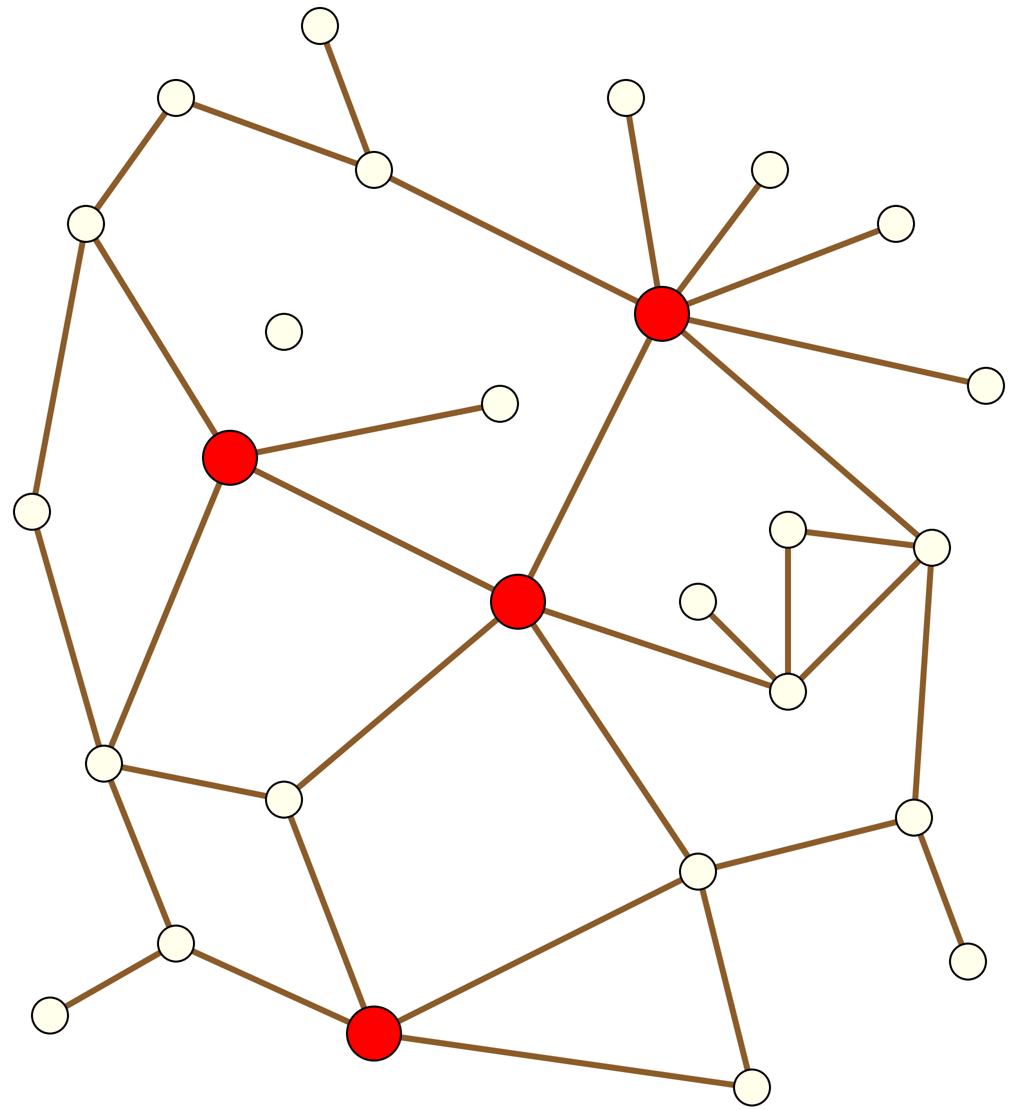
Maximum number such that
h vertices each have ≥ *h* neighbors

H = set of *h* high-degree vertices
L = remaining vertices

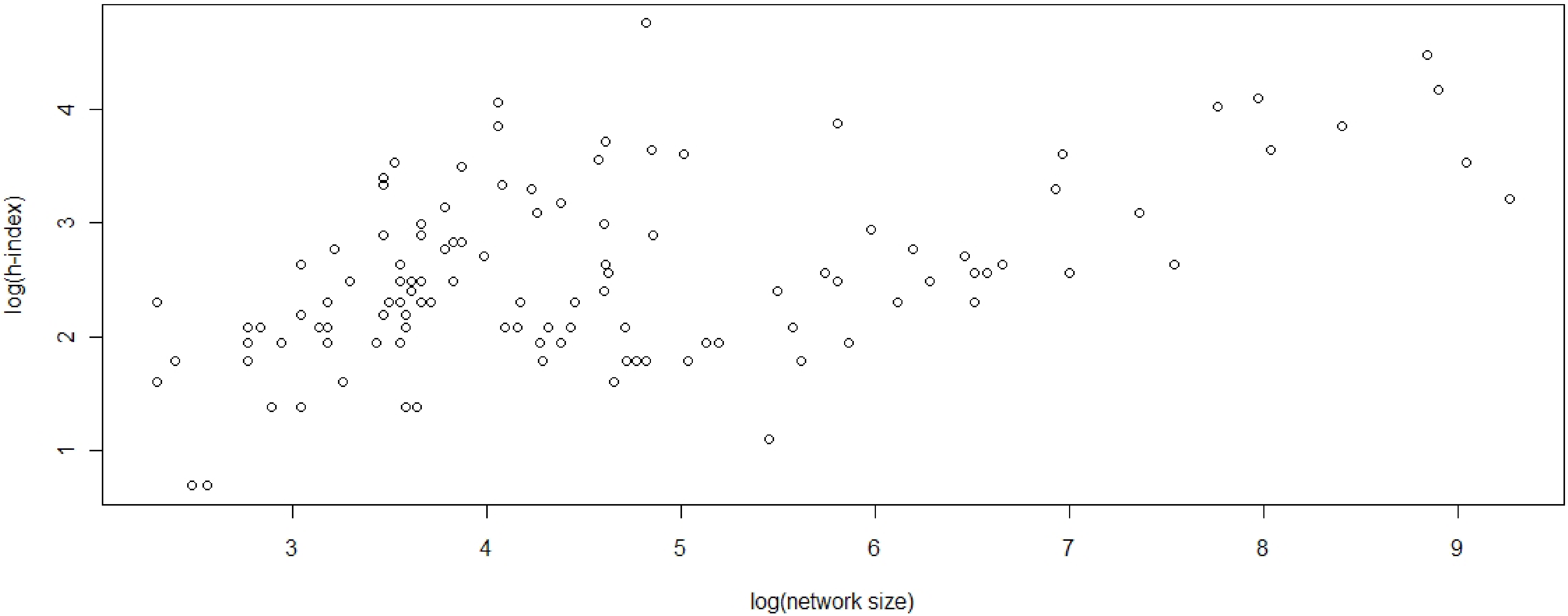
All vertices in L have degree ≤ *h*

Provides optimal tradeoff
between |H| and D

Never more than sqrt(*m*)
Else H would have too many edges



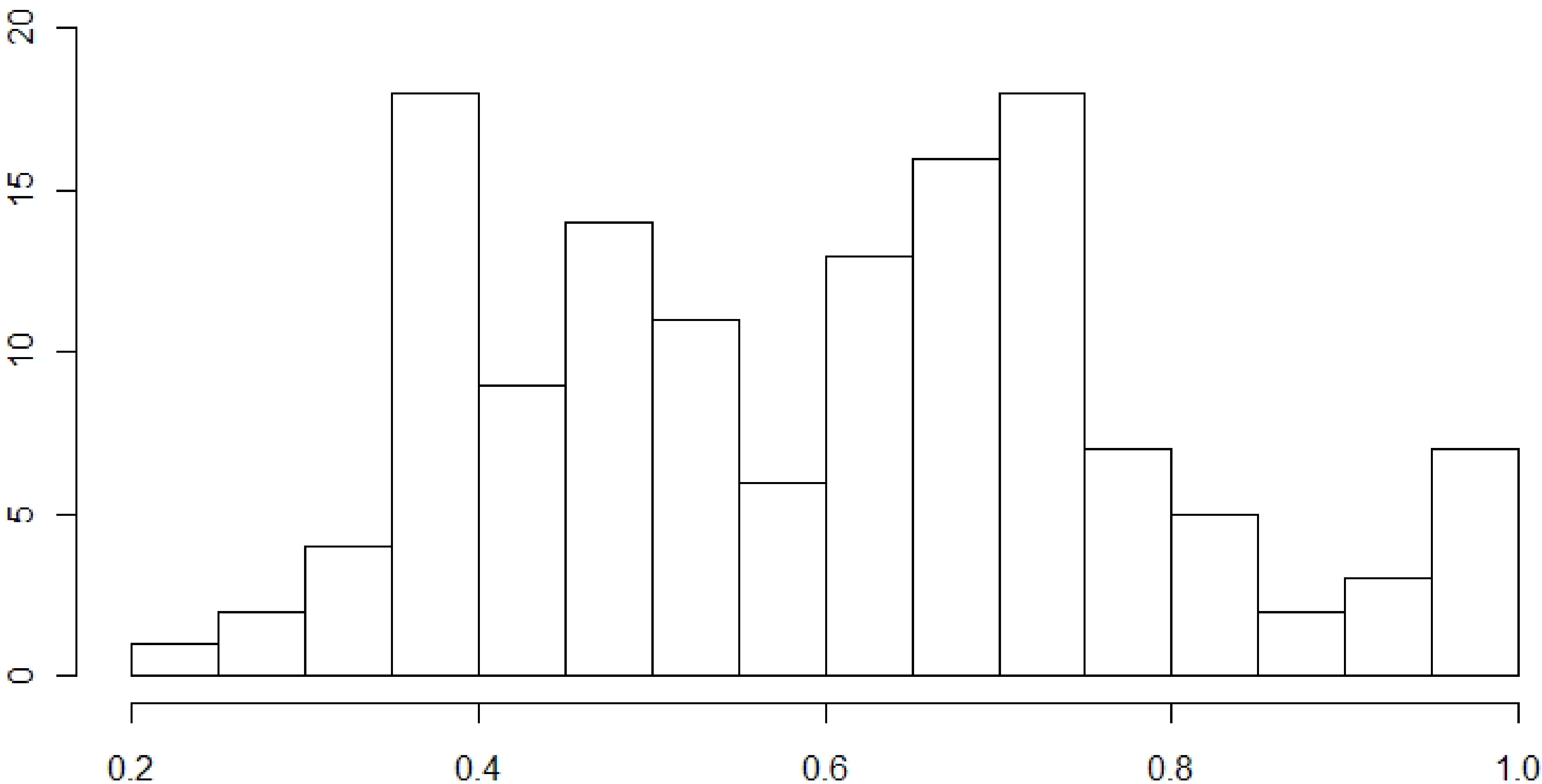
The *h*-index of some actual social networks



136 networks from Pajek, UCINET, statnet, UCI Network Data Repository

h-index scaling as a power of *n*

(frequency histogram of log *h* / log *n*)



Appears to be bimodal; we don't have an explanation

Algorithms based on h-index will be faster for networks in the first peak

Maintaining *h*-index and *h*-partition efficiently

Group vertices by degree

Degree > *h*: always in H
Degree < *h*: always in L
Degree = *h*: some in H and some not (store as two separate groups)

When adding an edge to vertex *v*:

Move *v* to new degree group

If *v* was in L but degree now > *h*:
Move it into H
Find *w* in H with degree *h*, move to L
If no *w* exists, increase *h*

When removing an edge from *v*:

Move *v* to new degree group

If *v* was in H but degree now < *h*:
Move it into L
Find *w* in L with degree *h*, move to H
If no *w* exists, decrease *h*

O(1) time per update

O(1) changes to the partition per update (too frequent!)

Even more efficient

Maintain h-index itself as before

Modify partition into H and L so that it changes less frequently

When degree exceeds $2h$, move vertex into H

When degree drops below h, move vertex into L

Average number of changes to partition per update: **$O(1/h)$**

Easy part of analysis: if h remains constant,
h updates needed to move a vertex through neutral zone

Less easy: what if h itself changes?

Conclusions

Data structure for speeding up MCMC steps in ERGM simulation

O(*h*) time per step to update all possible 3-vertex feature counts
New graph invariant *h* may be of independent interest

Can be generalized to labeled vertices
(e.g. male/female or researcher/publication)
and weighted edges

Future directions

So far, analysis is theoretical
Needs experimental validation

Faster for sparse graphs?

Additional ERGM features?