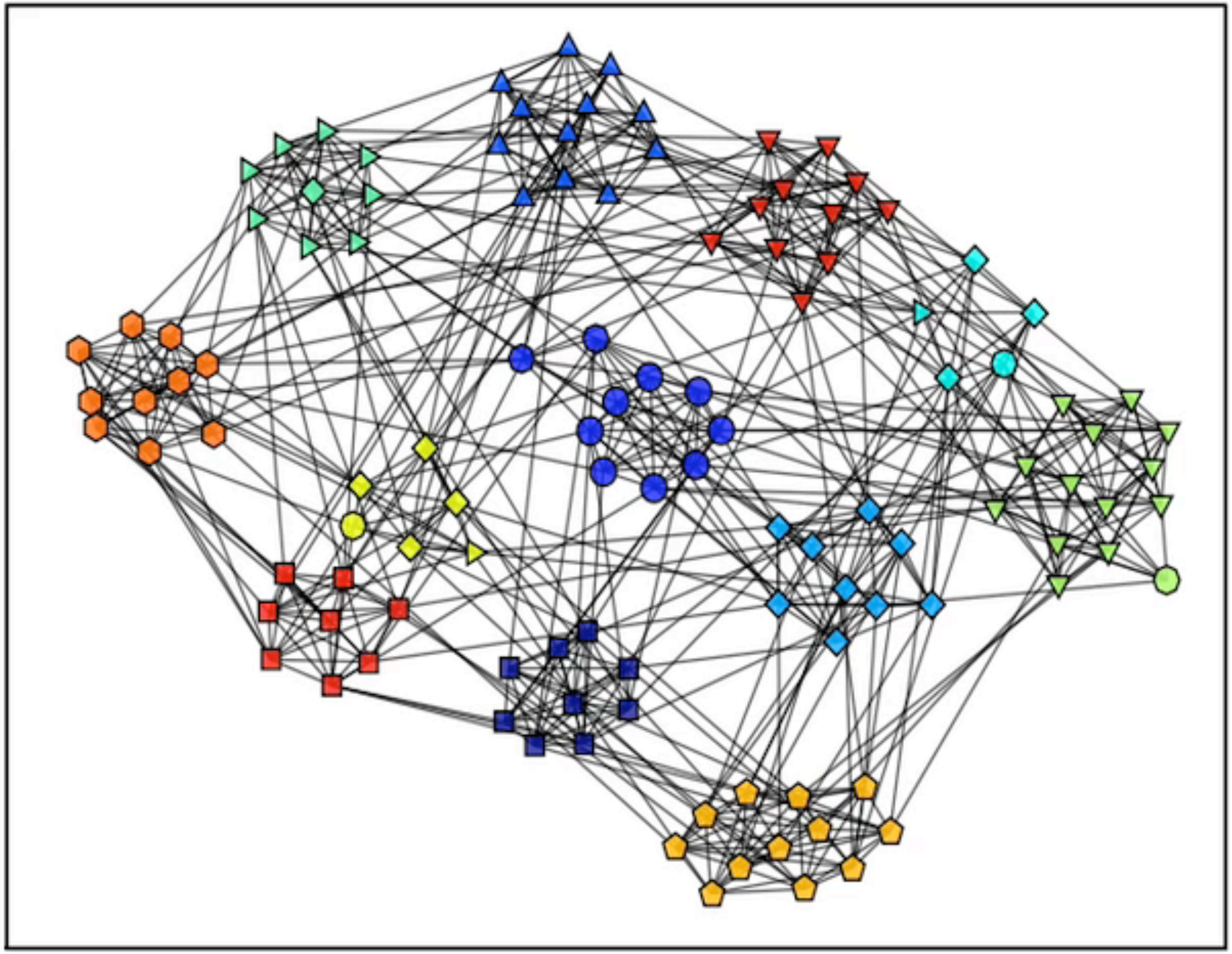


Community detection: model fitting, comparison, and utility

Jake Hofman
Yahoo! Research

2008.12.04

Community detection

- *Model* structure (e.g. summarize data)
- *Visualize* structure (e.g. graph layout)
- *Analyze* interactions (e.g. affinities within/between groups)
- *Predict* (e.g. function, attributes, links)

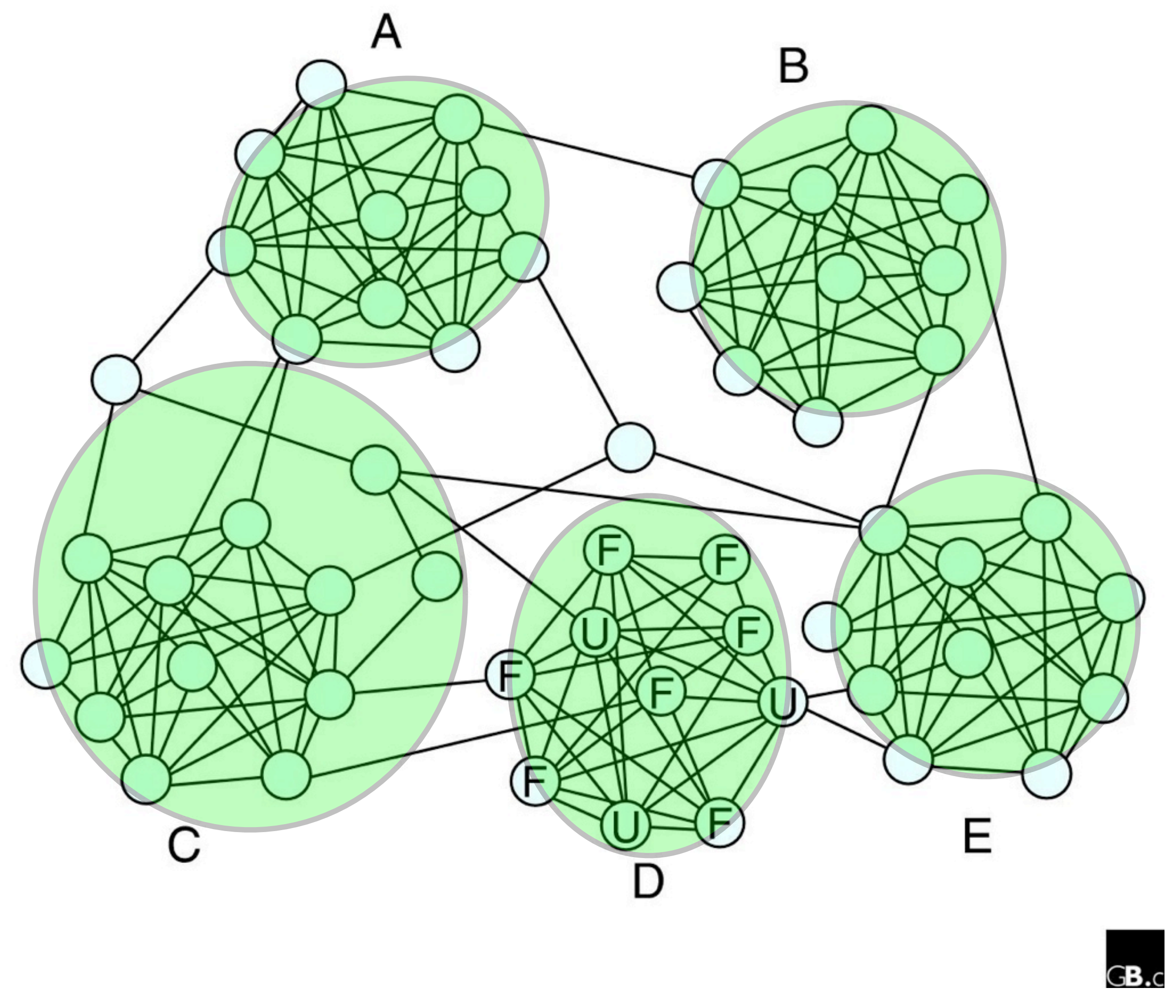


Image from Giorgini, et. al., Genome Biol, 2005

Community detection: Background

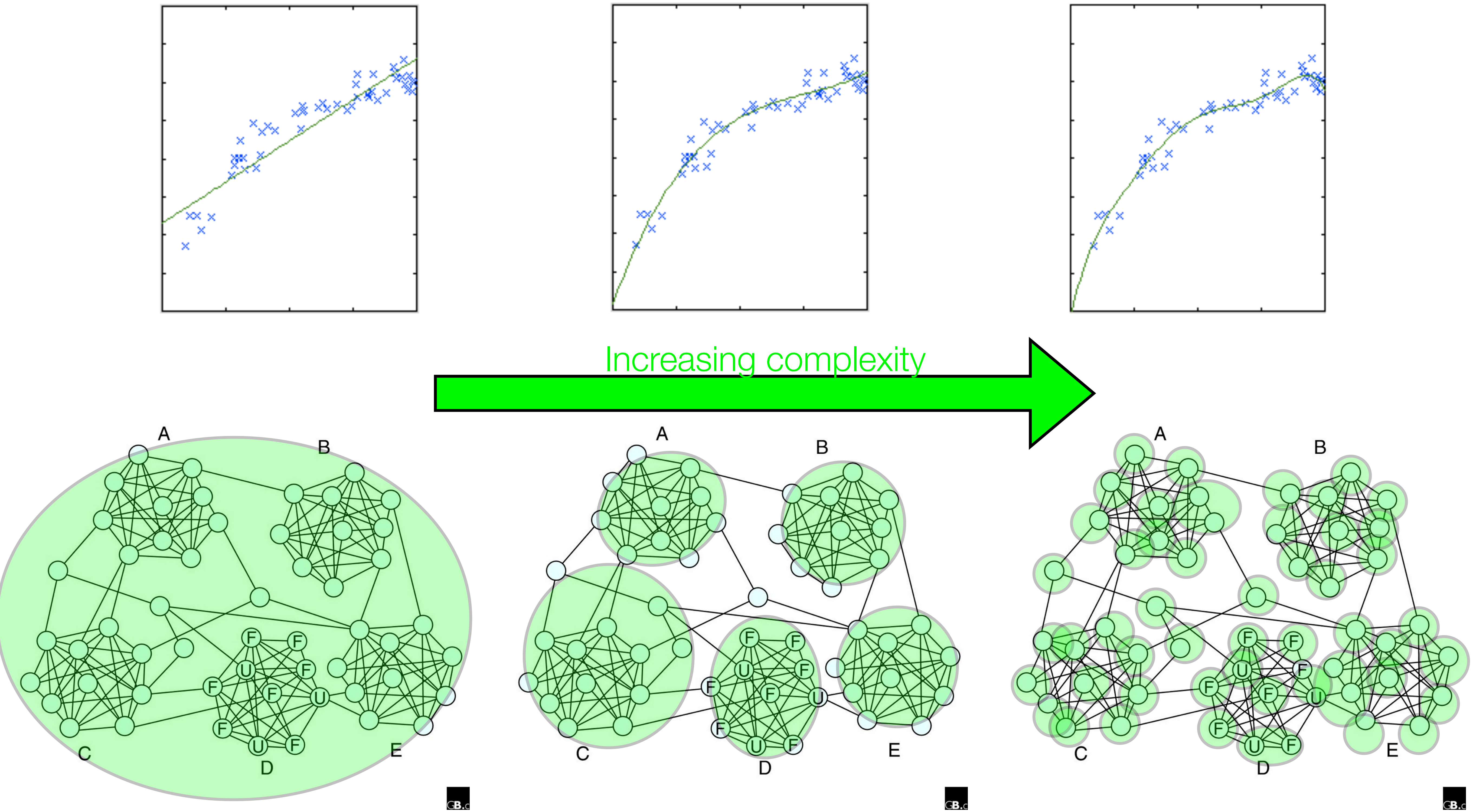
- Physics literature
  - Newman et. al. (2002,...)
  - Bornholdt & Reichardt (2006)
  - Hastings (2006)
  - ...
- Parametrized cost function (energy), mostly focus on *how* to optimize

- Machine learning literature
  - Nowicki & Snijders (2001)
  - Kemp et. al. (2004)
  - Leicht & Newman (2007)
  - Airoldi et. al. (2007)
  - Xu et. al. (2007)
  - Sinkkonen et. al. (2007)
- Complex models, approximate inference (often expensive)

Community detection: Questions

- Methodology:
  - Can we infer the complexity of a given network?
  - Can we compare competing network models?
- Applications:
  - How does topological community structure correlate with attributes/function?
  - How does community structure vary over time?

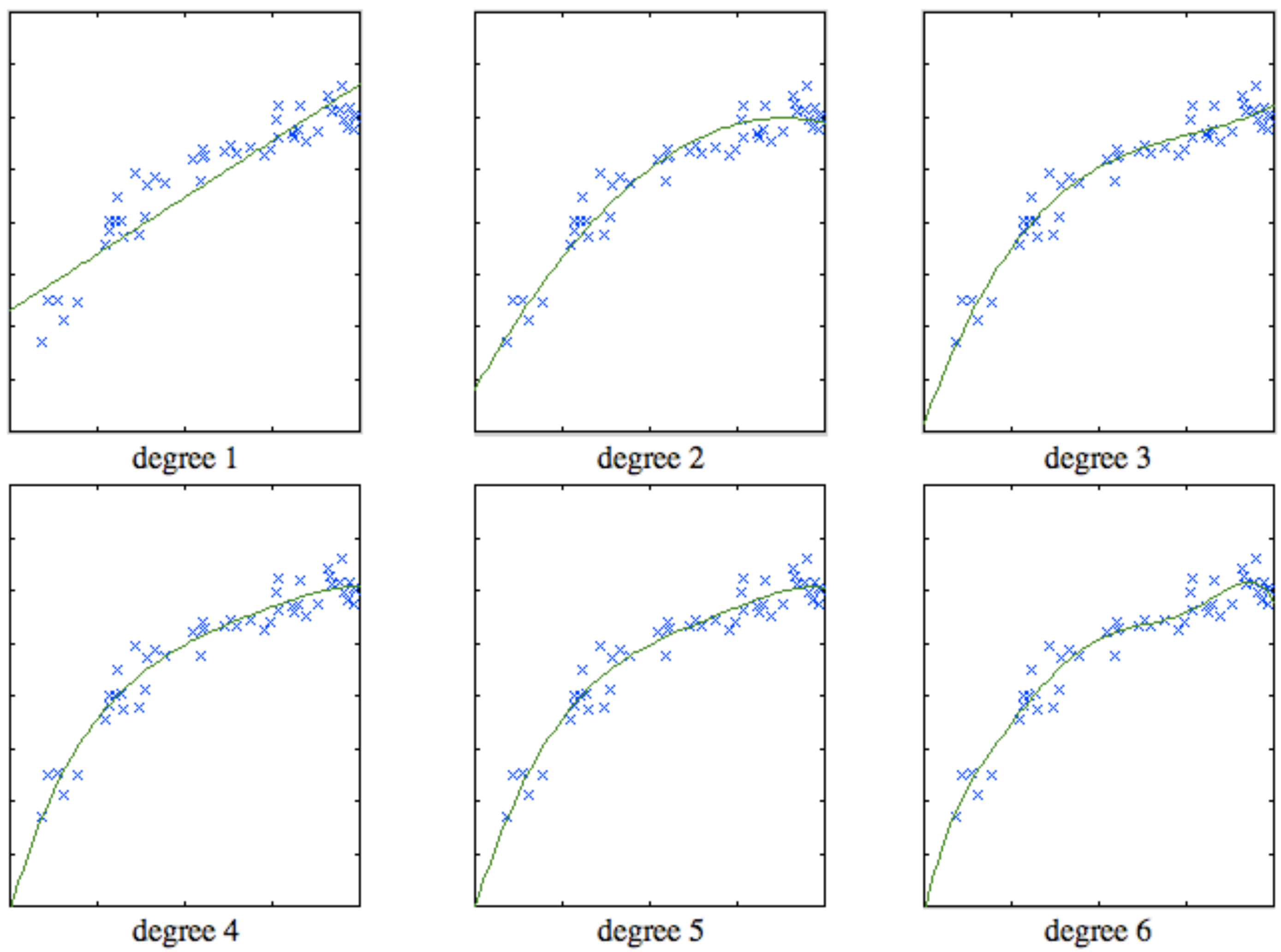
Complexity control



Images from http://research.microsoft.com/~minka/statlearn/demo/ and Giorgini, et. al., Genome Biol, 2005

Regression

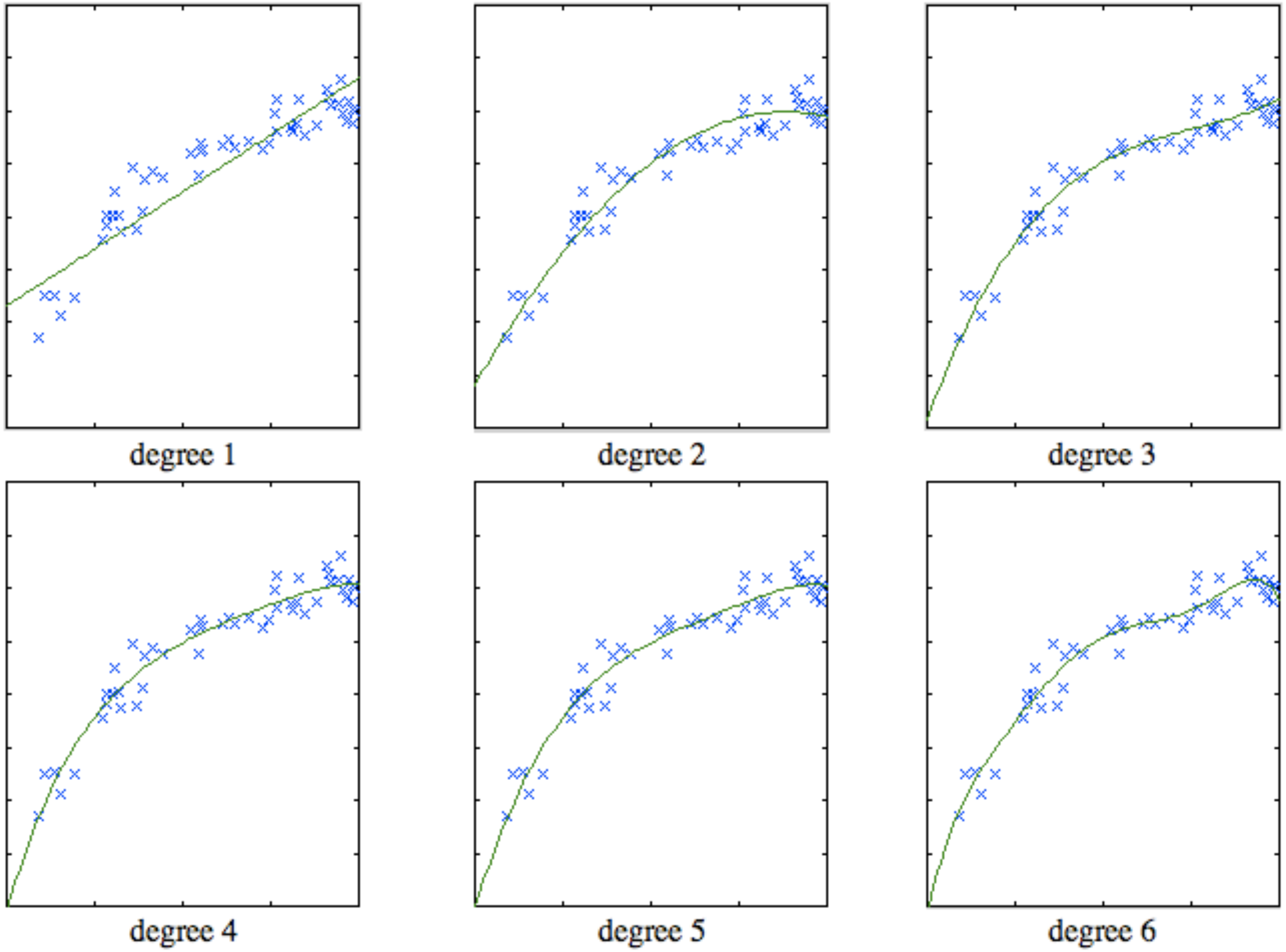
- Data D={x_i,y_i} i=1,...,N
- Parameters Θ=coefficients (e.g. slope, intercept,...), k=1,...,K
- Given D, infer Θ and K

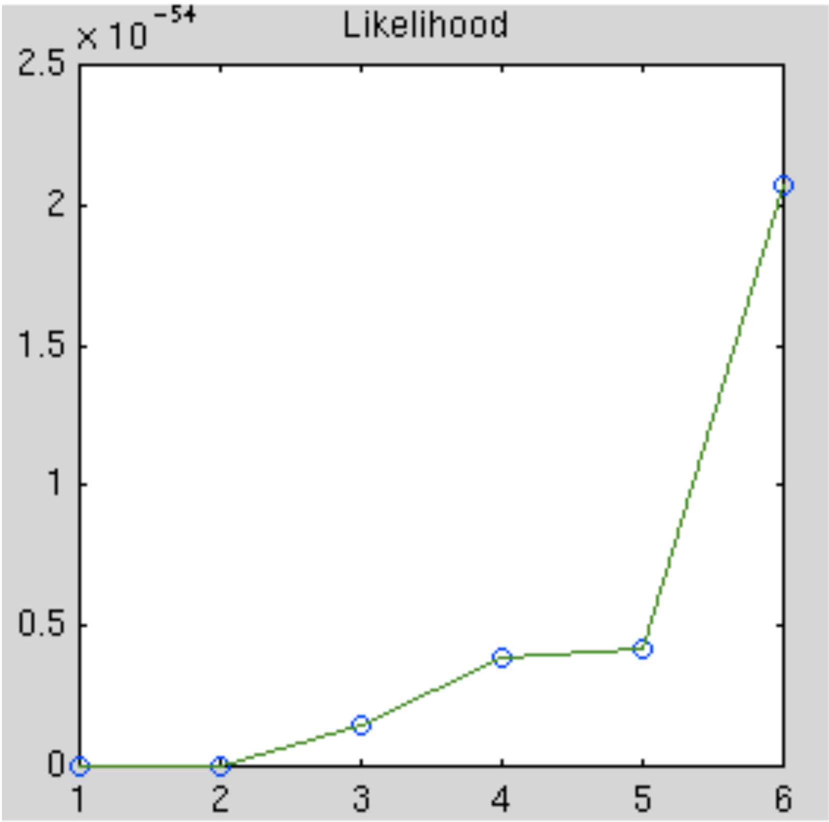


Images from http://research.microsoft.com/~minka/statlearn/demo/

Maximum likelihood regression

- Given D, infer Θ *and* K
- "Best parameters" = most probable: $\Theta^* = \text{argmax}_\Theta \; p(D|\Theta,K)$
- Problem: likelihood, $p(D|\Theta,K)$, increases with K

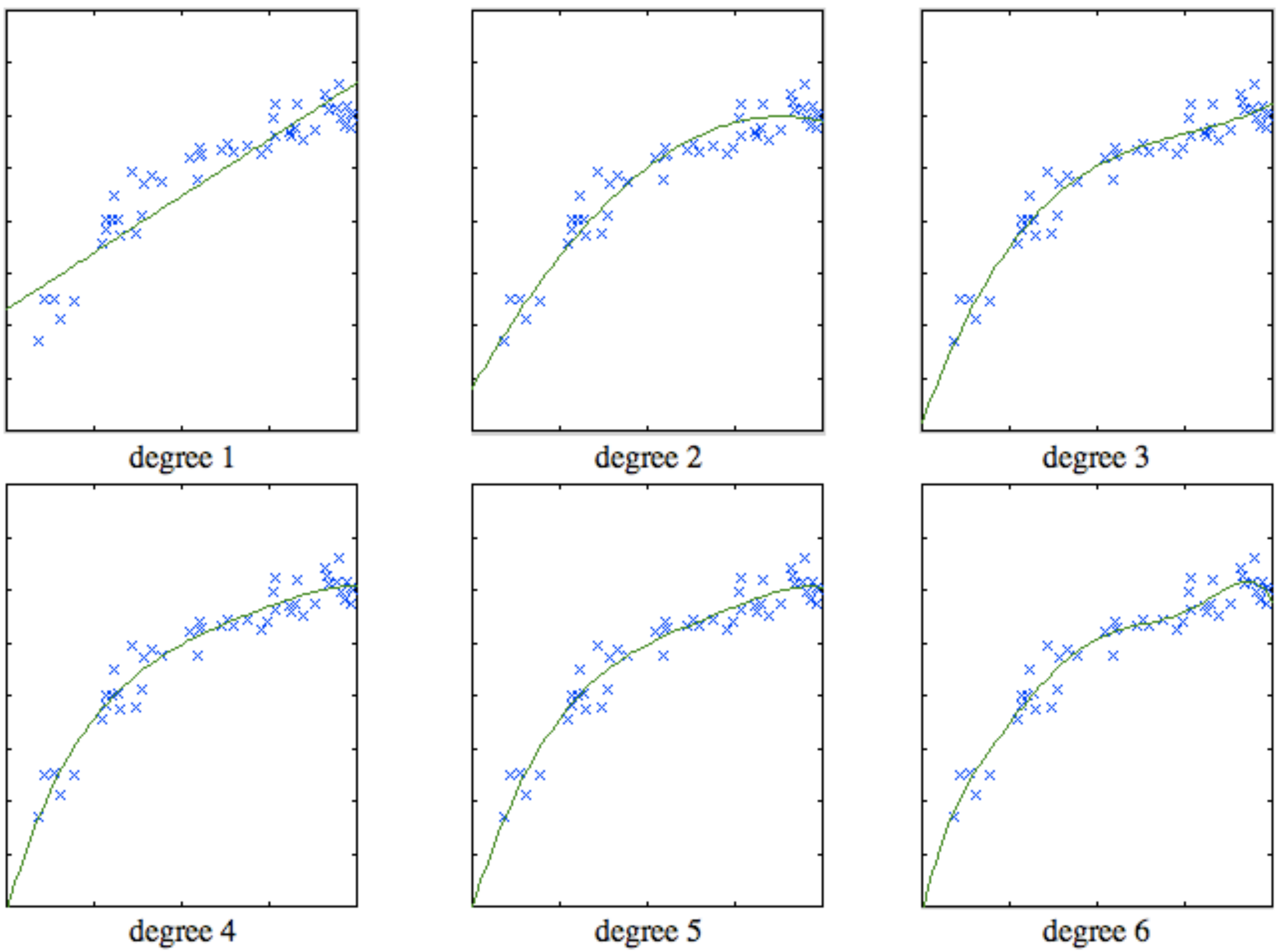


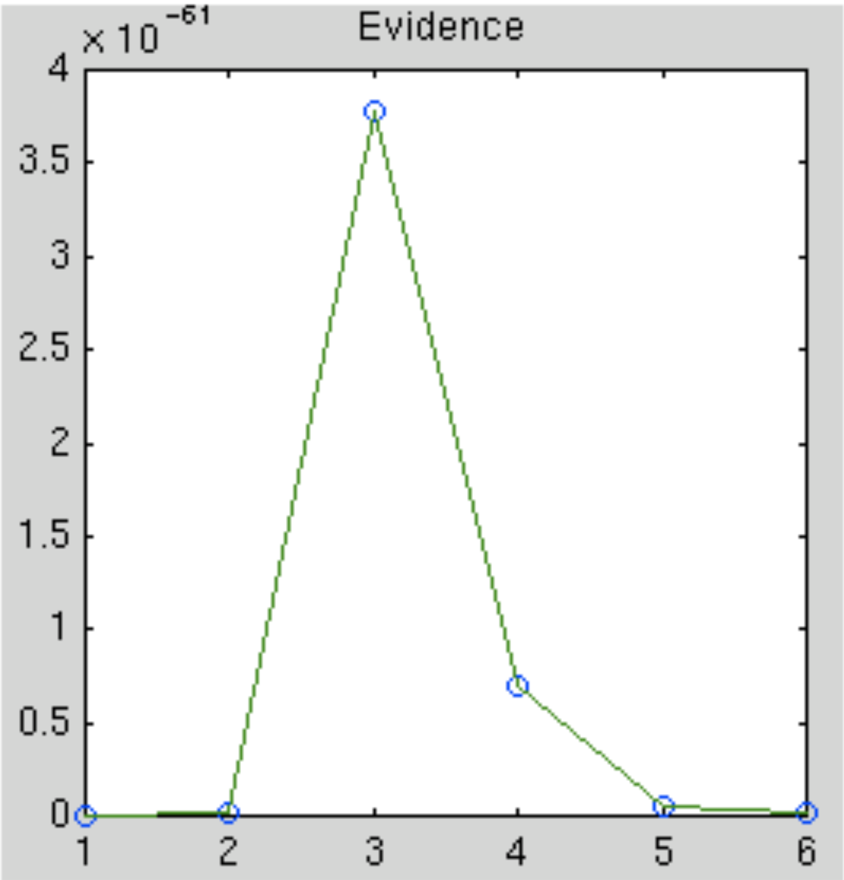


Images from http://research.microsoft.com/~minka/statlearn/demo/

Bayesian regression

- Given D, infer Θ *and* K
- "Best complexity" = most probable: $K^* = \text{argmax}_K\ p(D|K)$





http://research.microsoft.com/~minka/statlearn/demo/

Bayesian inference

- “Best complexity” = most probable: $K^* = \text{argmax}_K\ p(D|K)$
- Avoid choosing best parameters; integrate over parameters

posterior likelihood prior

$$p(\theta|\mathcal{D}, K) = \frac{p(\mathcal{D}|\theta, K)\,p(\theta|K)}{p(\mathcal{D}|K)}$$

evidence

where

$$p(\mathcal{D}|K) = \int d\theta\ p(\mathcal{D}|\theta, K)\,p(\theta|K)$$

Bayes (1763), Jeffreys (1935)

Bayesian inference

- Inference = inverse statistical mechanics
- Calculate and optimize partition function (or free energy)

posterior　　likelihood　　prior

$$p(\Theta|\mathcal{D}, K) = \frac{\mathrm{e}^{-\mathcal{H}}\, p(\Theta|K)}{\mathcal{Z}}$$

evidence

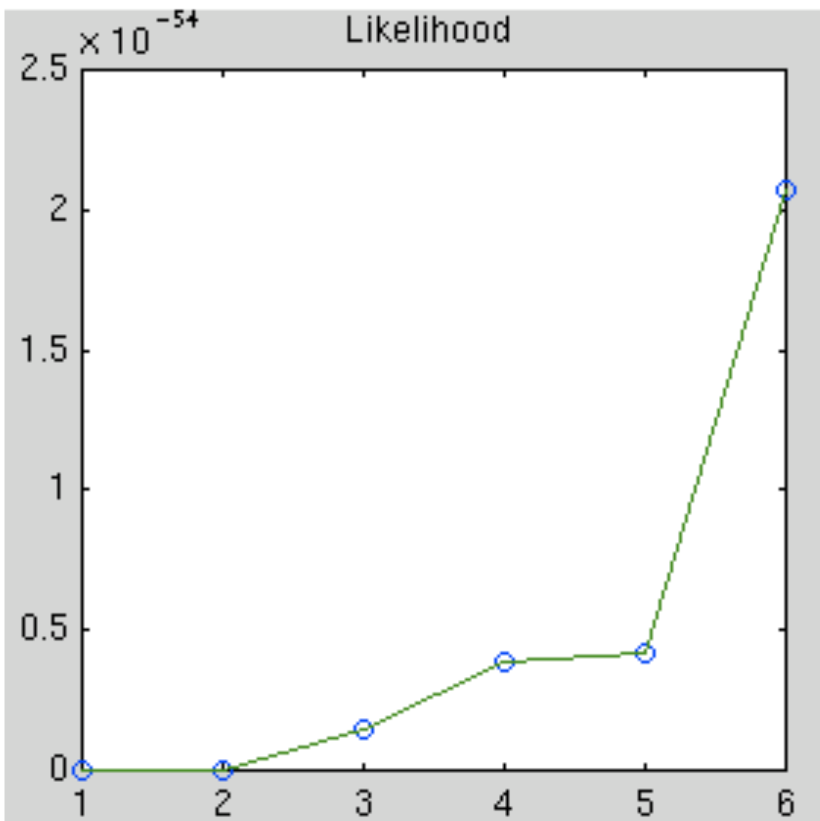
where

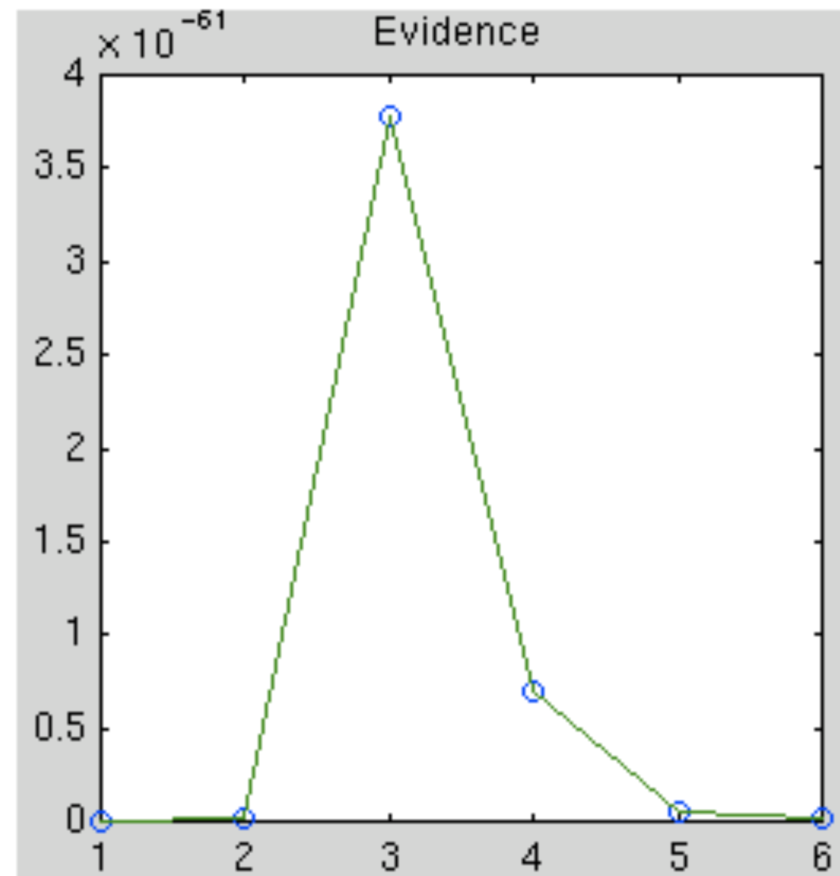
$$\mathcal{H} = -\ln p(\mathcal{D}|\Theta, K)$$

$$\mathcal{Z} = \int d\Theta \;\; \mathrm{e}^{-\mathcal{H}}\, p(\Theta|K)$$

Bayesian complexity control

- Maximize evidence (integrating over unknown parameters and latent variables) to infer most probable model complexity





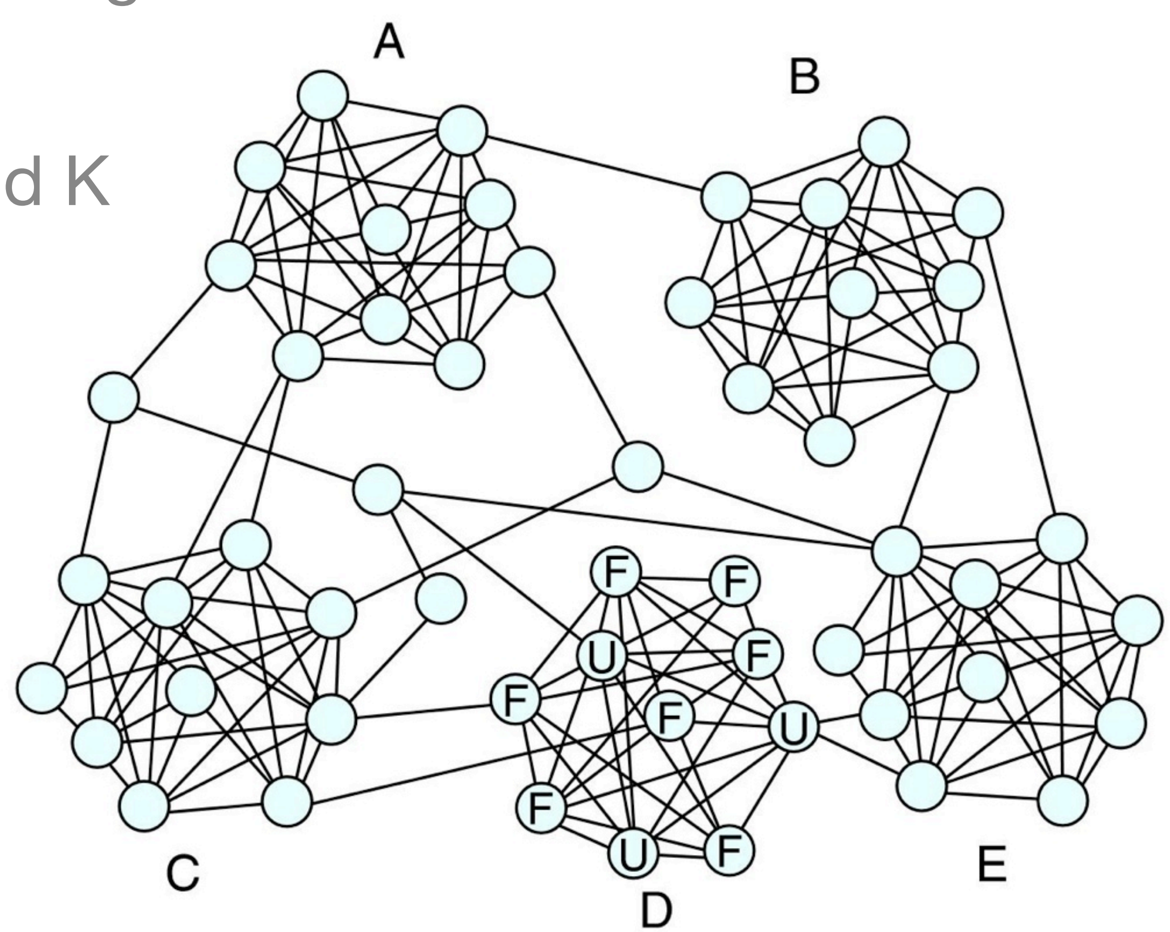
$$\begin{aligned}\hat{\theta} &= \arg\max_{\theta} p(\mathcal{D}|\theta, K) \\ &= \arg\max_{\theta} \sum_{Z} p(\mathcal{D}, Z|\hat{\theta}, K)\end{aligned}$$

$$\begin{aligned}\hat{K} &= \arg\max_{K} p(\mathcal{D}|K) \\ &= \arg\max_{K} \sum_{Z} \int d\theta \; p(\mathcal{D}, Z|\theta, K) p(\theta|K)\end{aligned}$$

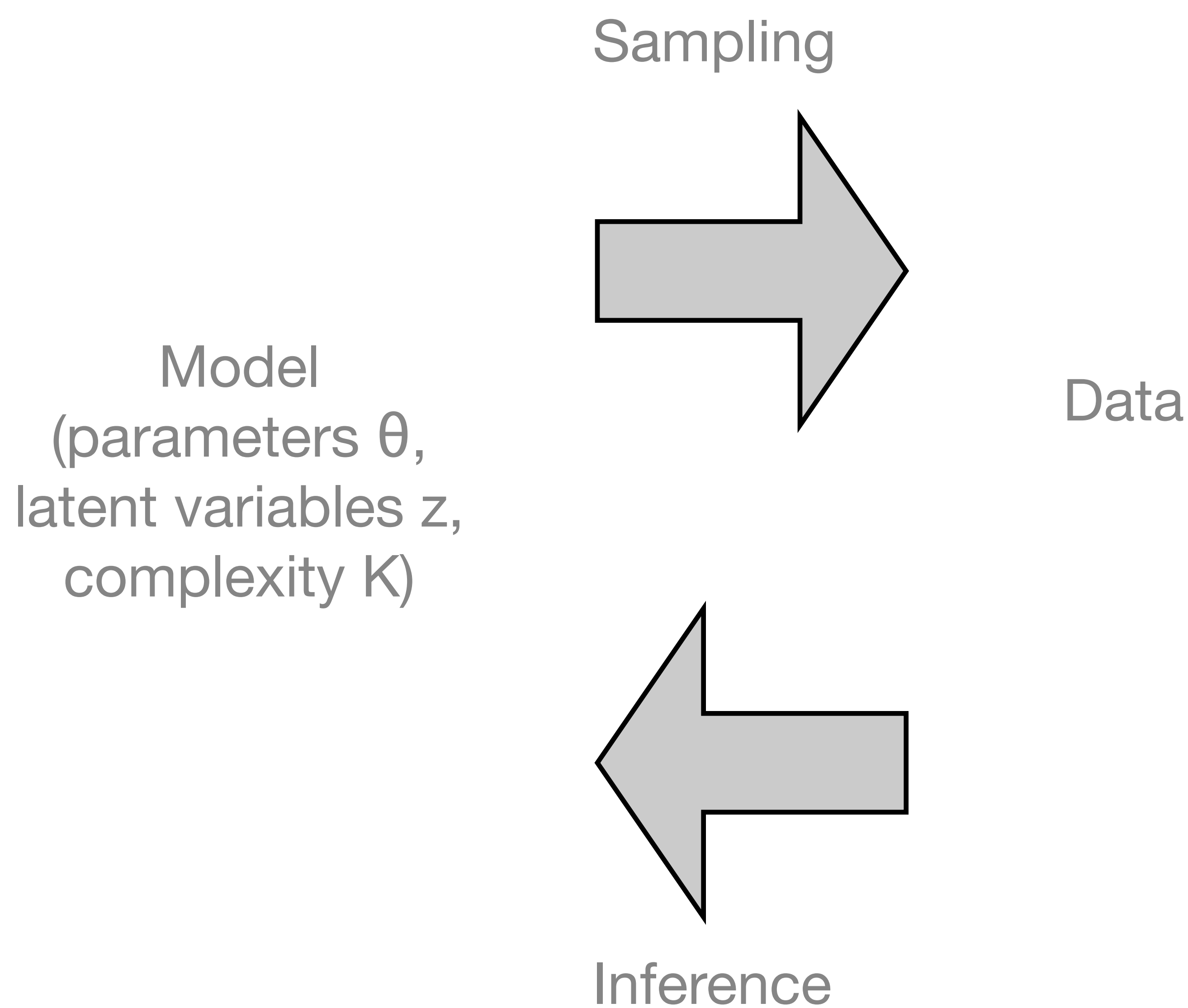
http://research.microsoft.com/~minka/statlearn/demo/

Community detection as inference

- Data D={A_{ij}} i,j=1,...,N; A_{ij}=1 if nodes i and j connected
- Parameters bias of die π, bias of coins θ
- Latent variables {z_i}, assignments of nodes to communities
- Given D, infer z, Θ and K

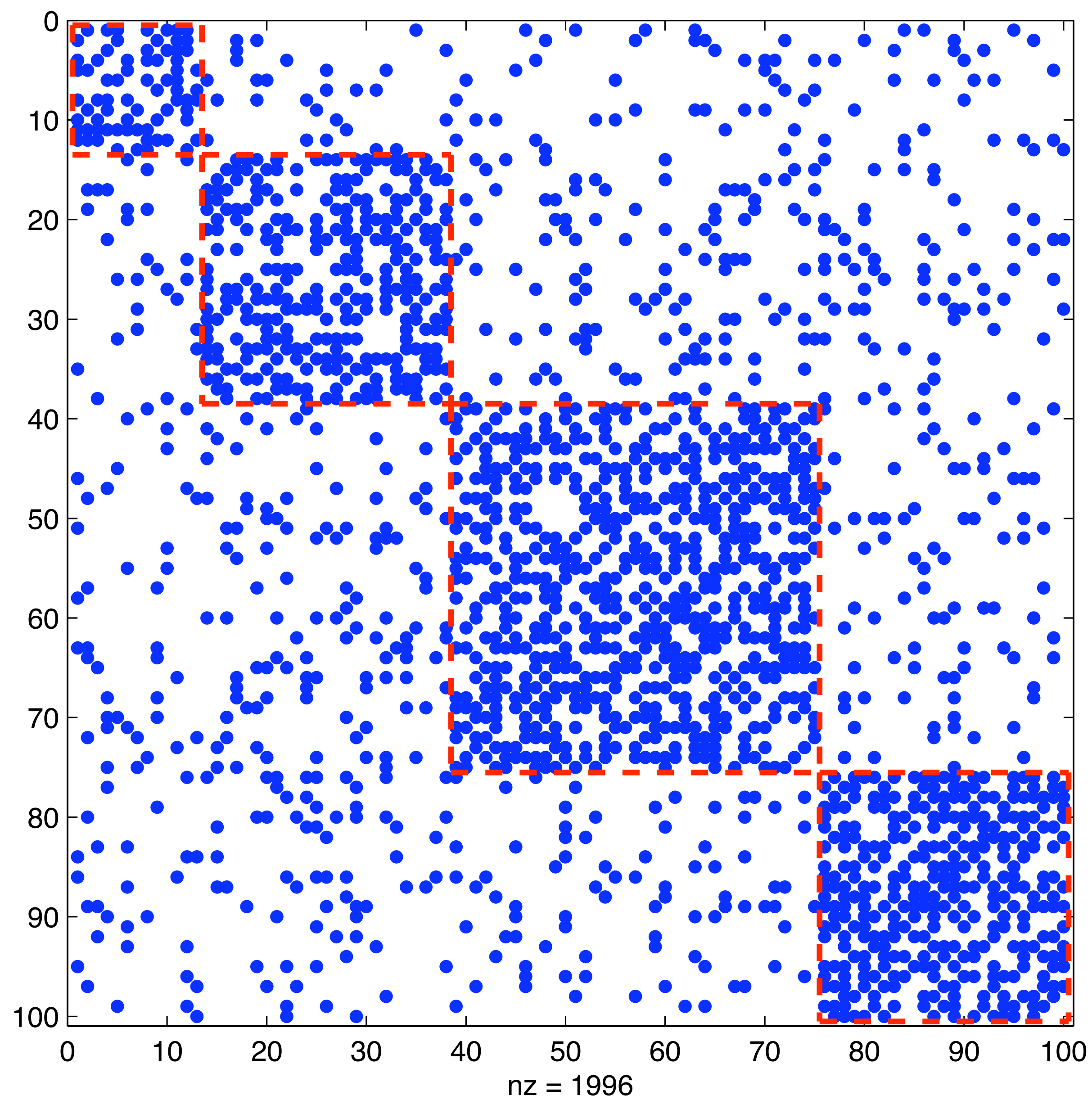


Community detection as inference



Constrained stochastic block model

- **Nodes belong to “blocks”** of varying size
  - Roll die for assignment of nodes to blocks
- Probability of **edge** between two nodes **depends only on block membership**
  - Flip (one of two) coins for edges
- Result: **mixture of Erdos-Renyi** graphs



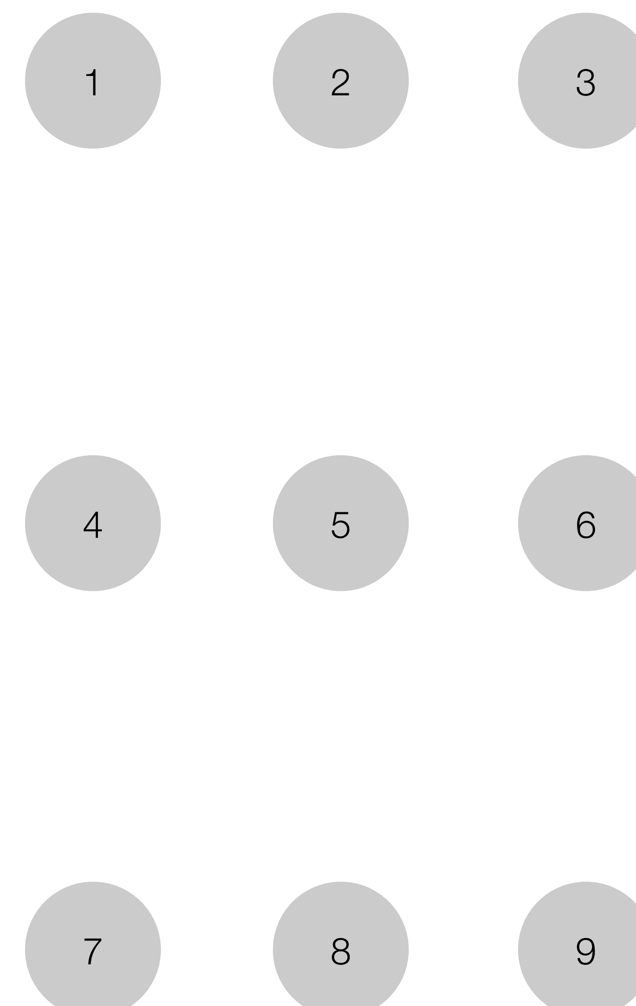
Holland, Laskey, Leinhardt 1983; Wang and Wong, 1987

Generating modular networks

- For each node:
    - **Roll K-sided die** with bias π to determine z_i=1,...,K, the (unobserved) module assignment for i^{th} node
- For each pair of nodes (i,j):
    - If z_i=z_j, **flip "in community" coin** with bias θ_c to determine edge A_{ij}
    - If $z_i \neq z_j$, **flip "between communities" coin** with bias θ_d to determine edge A_{ij}

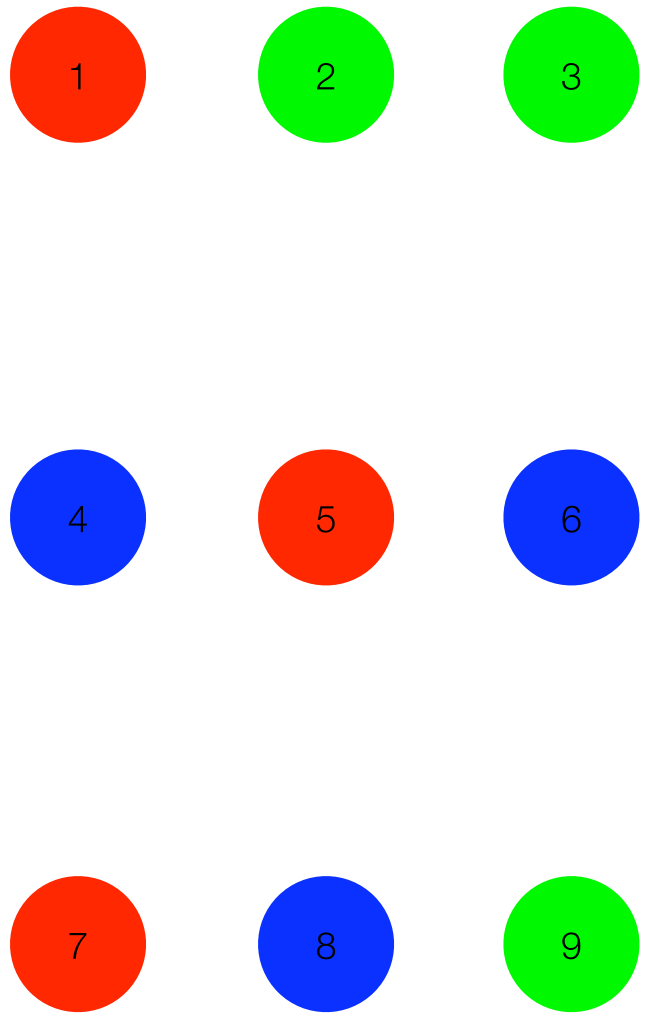
Generating modular networks

- For each node:
    - **Roll K-sided die** with bias π to determine z_i=1,...,K, the (unobserved) module assignment for i^{th} node
- For each pair of nodes (i,j):
    - If z_i=z_j, **flip "in community" coin** with bias θ_c to determine edge A_{ij}
    - If $z_i \neq z_j$, **flip "between communities" coin** with bias θ_d to determine edge A_{ij}



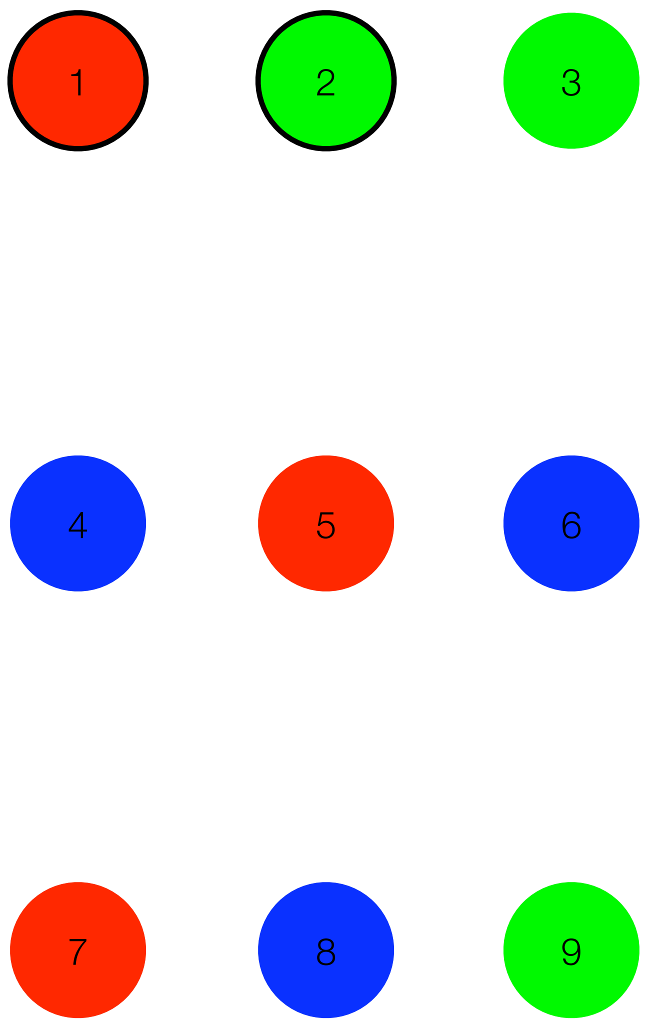
Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine z_i=1,...,K, the (unobserved) module assignment for i^{th} node
- For each pair of nodes (i,j):
  - If $z_i=z_j$, **flip “in community” coin** with bias θ_c to determine edge A_{ij}
  - If $z_i \neq z_j$, **flip “between communities” coin** with bias θ_d to determine edge A_{ij}



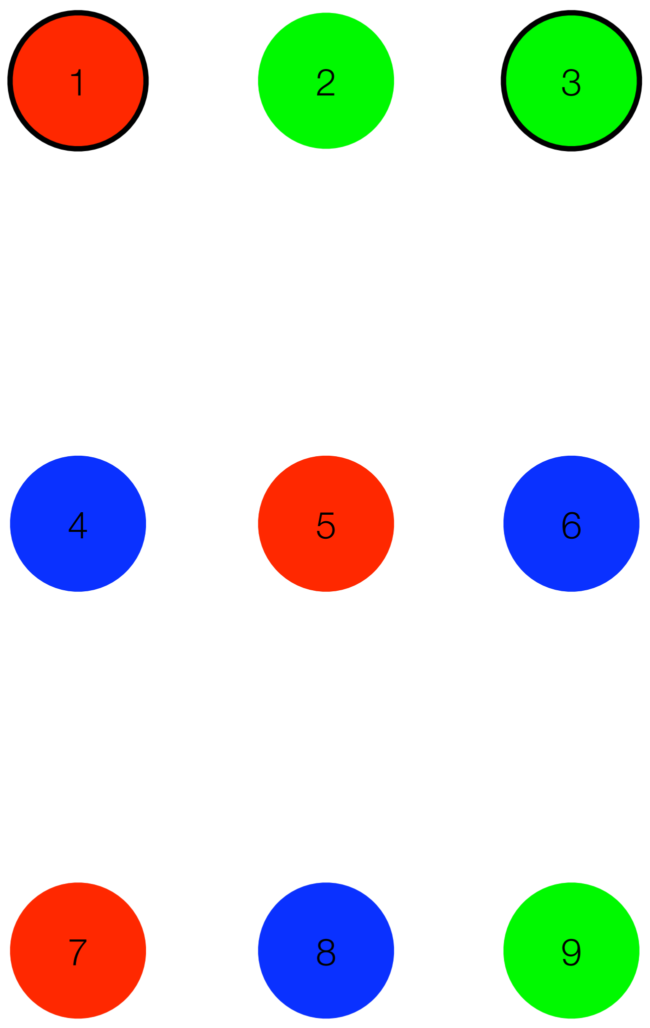
Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine z_i=1,...,K, the (unobserved) module assignment for i^{th} node
- For each pair of nodes (i,j):
  - If $z_i=z_j$, **flip "in community" coin** with bias θ_c to determine edge A_{ij}
  - If $z_i \neq z_j$, **flip "between communities" coin** with bias θ_d to determine edge A_{ij}



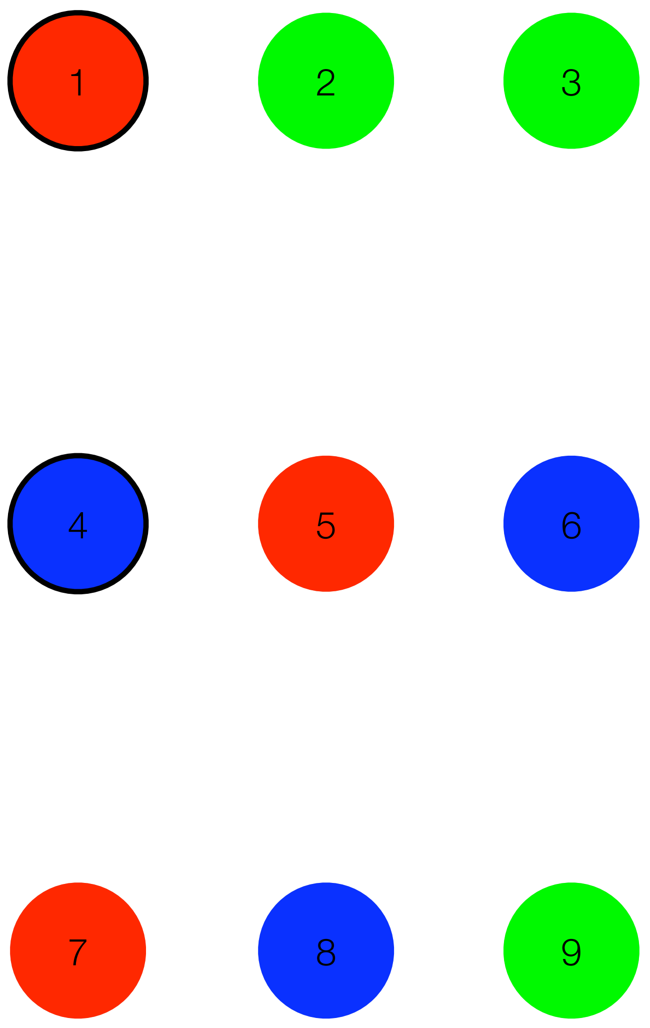
Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine z_i=1,...,K, the (unobserved) module assignment for i^{th} node
- For each pair of nodes (i,j):
  - If z_i=z_j, **flip "in community" coin** with bias θ_c to determine edge A_{ij}
  - If $z_i \neq z_j$, **flip "between communities" coin** with bias θ_d to determine edge A_{ij}



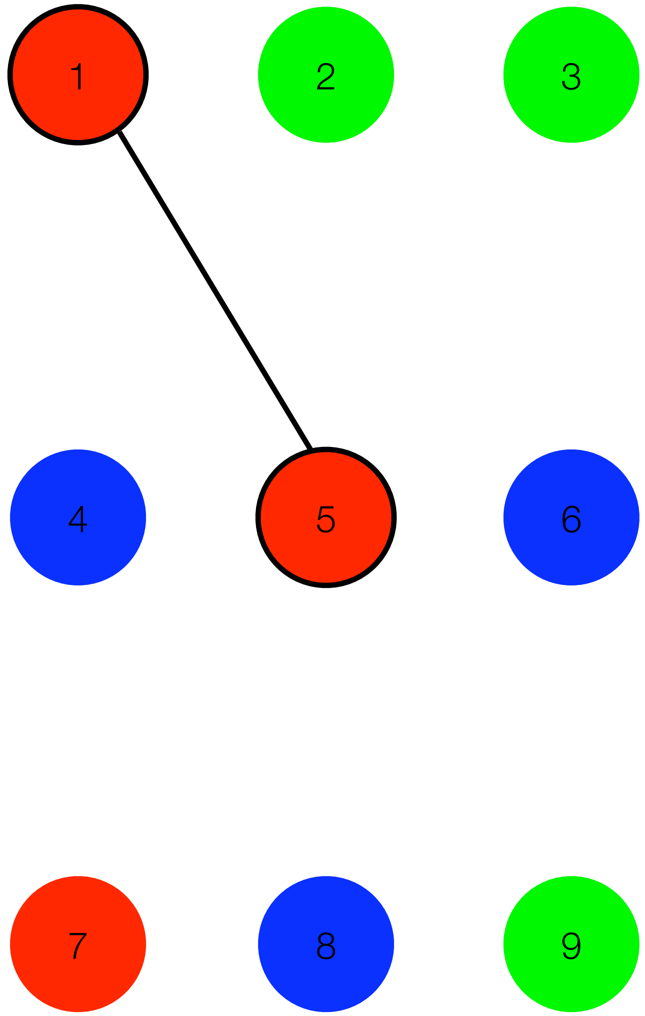
Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine z_i=1,...,K, the (unobserved) module assignment for i^{th} node
- For each pair of nodes (i,j):
  - If $z_i=z_j$, **flip "in community" coin** with bias θ_c to determine edge A_{ij}
  - If $z_i \neq z_j$, **flip "between communities" coin** with bias θ_d to determine edge A_{ij}



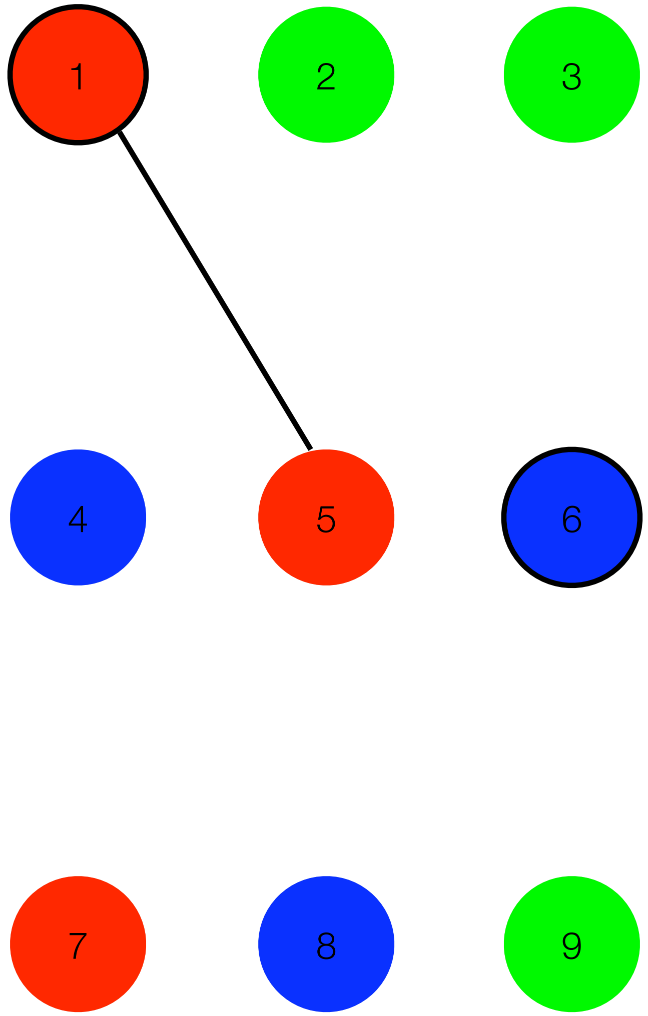
Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine $z_i=1,\ldots,K$, the (unobserved) module assignment for i^{th} node
- For each pair of nodes (i,j):
  - If $z_i=z_j$, **flip "in community" coin** with bias θ_c to determine edge A_{ij}
  - If $z_i \neq z_j$, **flip "between communities" coin** with bias θ_d to determine edge A_{ij}



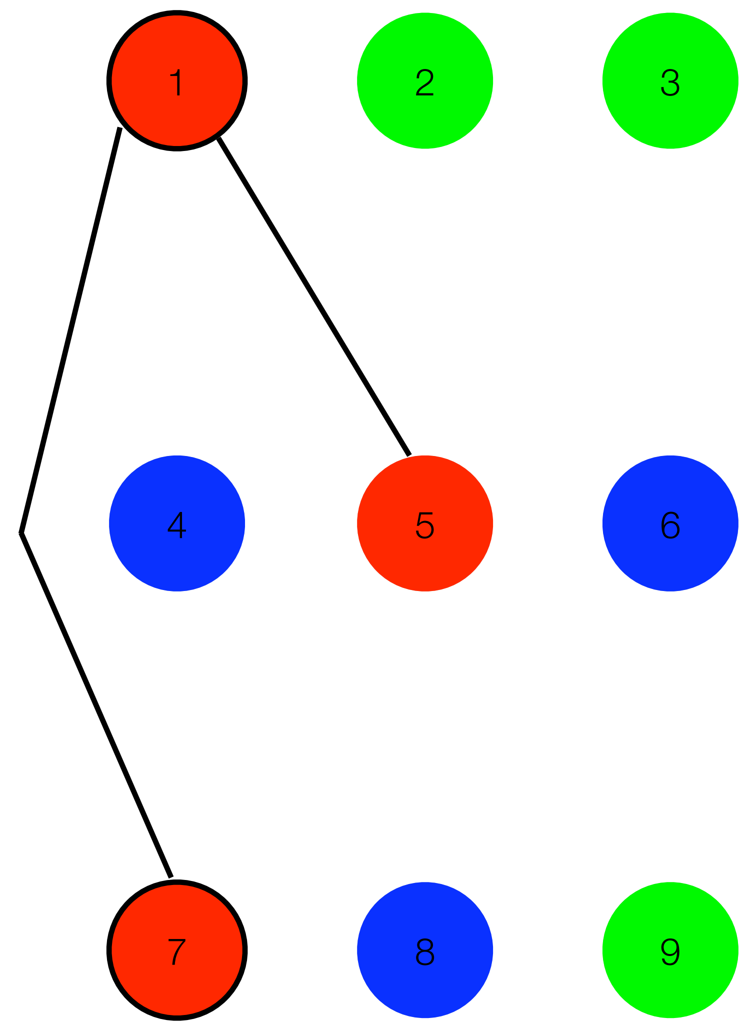
Generating modular networks

- For each node:
    - **Roll K-sided die** with bias π to determine z_i=1,...,K, the (unobserved) module assignment for i^{th} node
- For each pair of nodes (i,j):
    - If $z_i = z_j$, **flip "in community" coin** with bias θ_c to determine edge A_{ij}
    - If $z_i \neq z_j$, **flip "between communities" coin** with bias θ_d to determine edge A_{ij}



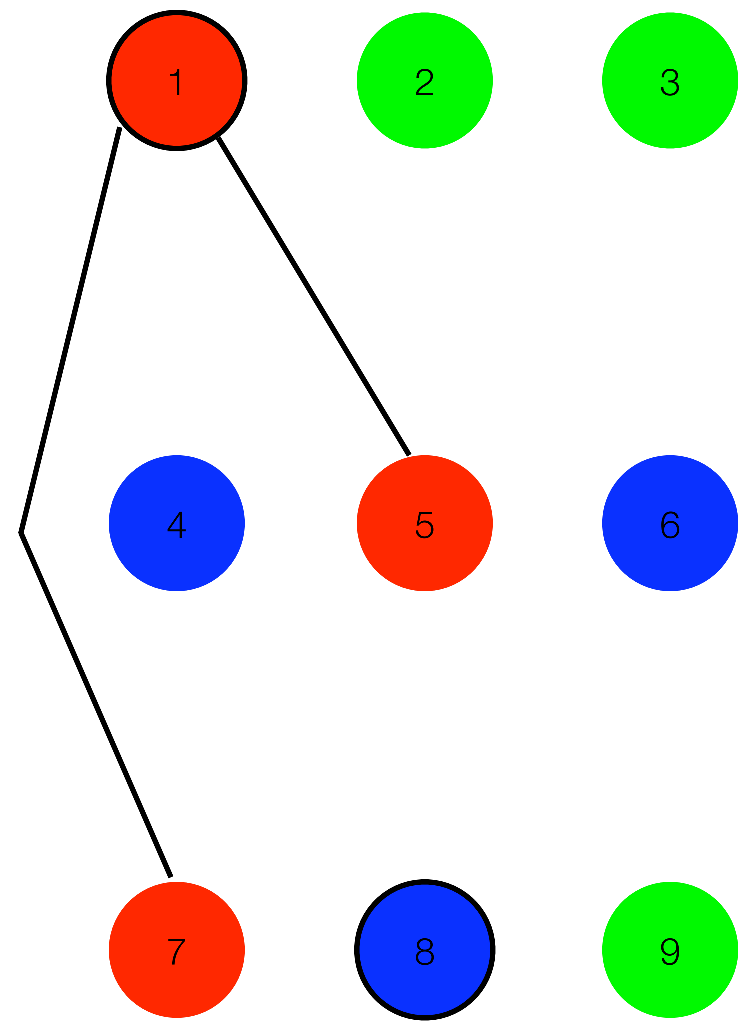
Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine z_i=1,...,K, the (unobserved) module assignment for i^{th} node
- For each pair of nodes (i,j):
  - If $z_i=z_j$, **flip "in community" coin** with bias θ_c to determine edge A_{ij}
  - If $z_i \neq z_j$, **flip "between communities" coin** with bias θ_d to determine edge A_{ij}



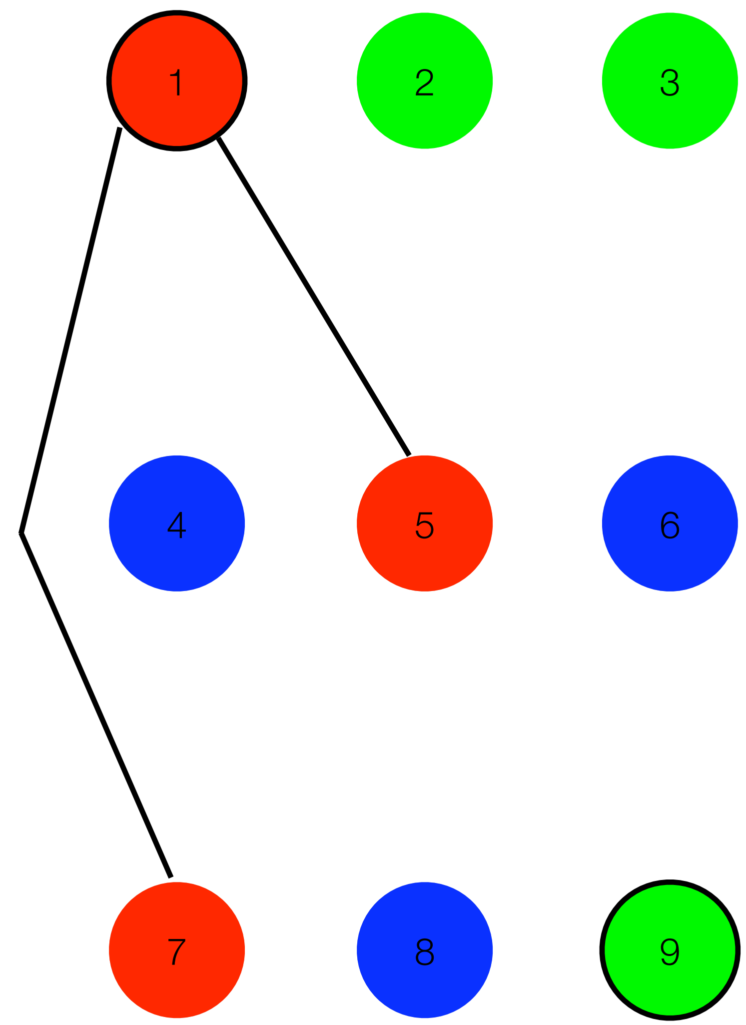
Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine z_i=1,...,K, the (unobserved) module assignment for i^{th} node
- For each pair of nodes (i,j):
  - If $z_i=z_j$, **flip "in community" coin** with bias θ_c to determine edge A_{ij}
  - If $z_i \neq z_j$, **flip "between communities" coin** with bias θ_d to determine edge A_{ij}



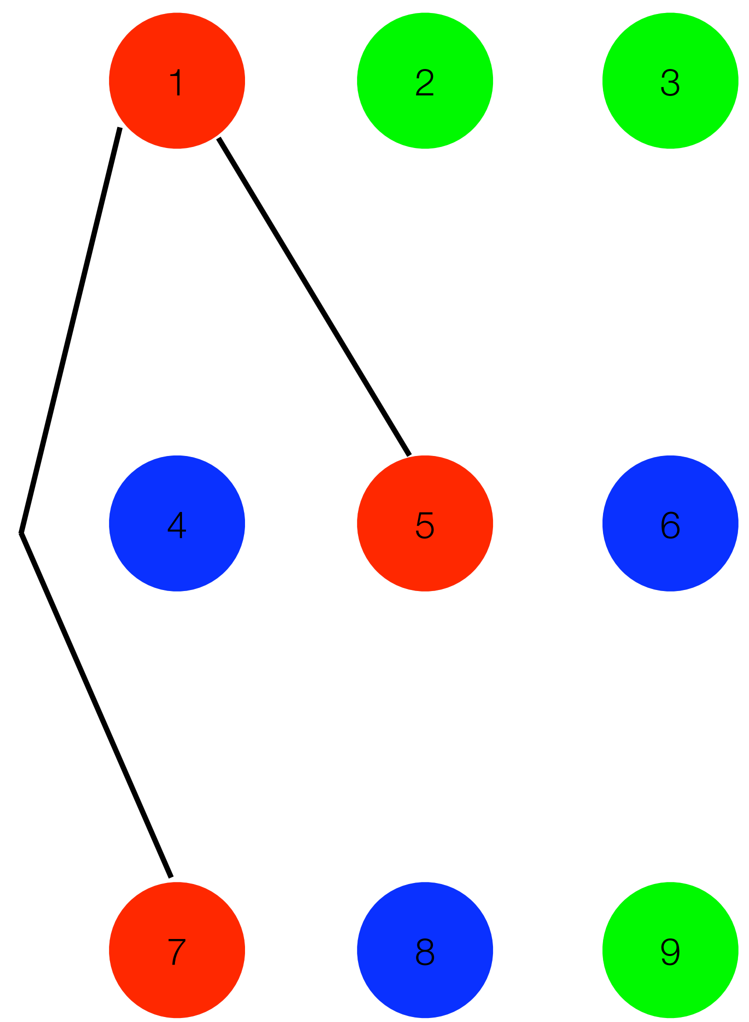
Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine $z_i=1,\ldots,K$, the (unobserved) module assignment for i^{th} node
- For each pair of nodes (i,j):
  - If $z_i=z_j$, **flip "in community" coin** with bias θ_c to determine edge A_{ij}
  - If $z_i \neq z_j$, **flip "between communities" coin** with bias θ_d to determine edge A_{ij}



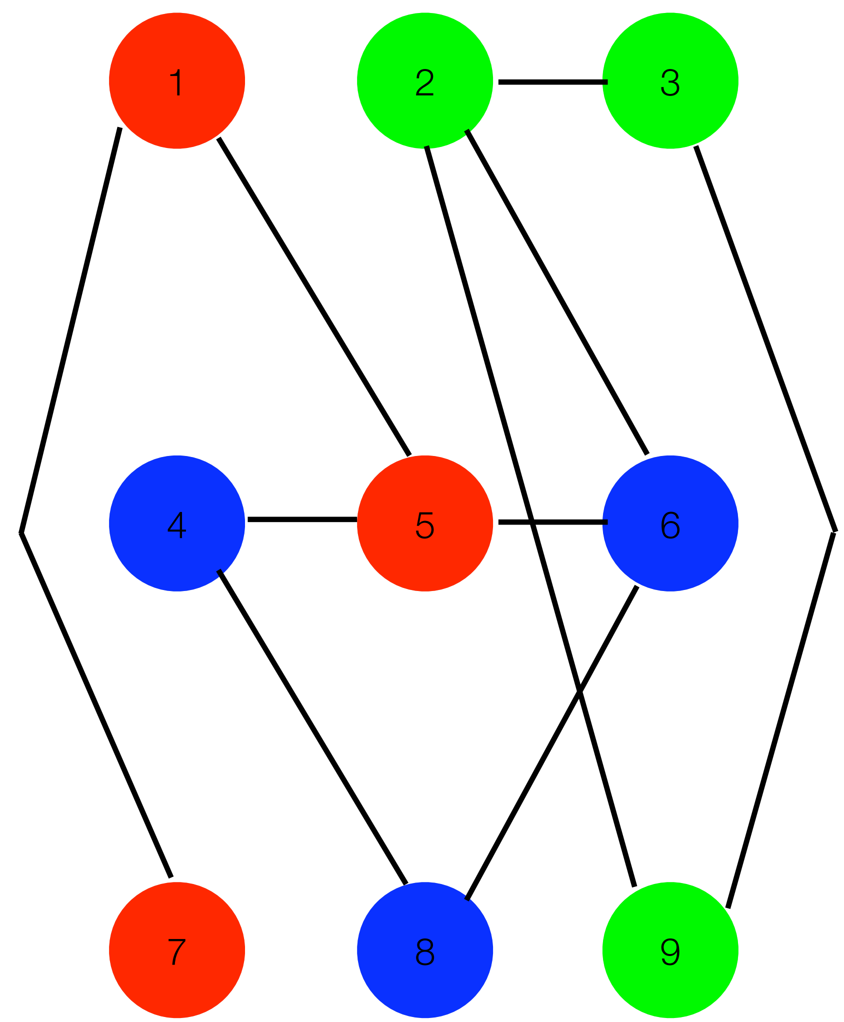
Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine z_i=1,...,K, the (unobserved) module assignment for i^{th} node
- For each pair of nodes (i,j):
  - If $z_i=z_j$, **flip "in community" coin** with bias θ_c to determine edge A_{ij}
  - If $z_i \neq z_j$, **flip "between communities" coin** with bias θ_d to determine edge A_{ij}



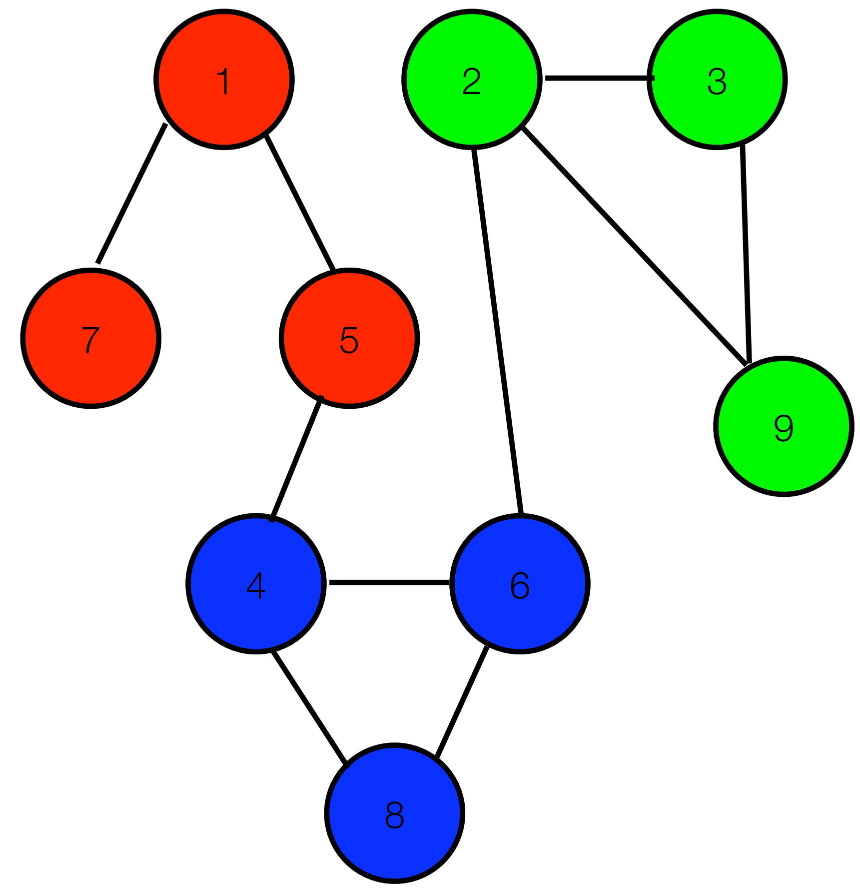
Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine z_i=1,...,K, the (unobserved) module assignment for i^{th} node
- For each pair of nodes (i,j):
  - If $z_i=z_j$, **flip "in community" coin** with bias θ_c to determine edge A_{ij}
  - If $z_i \neq z_j$, **flip "between communities" coin** with bias θ_d to determine edge A_{ij}

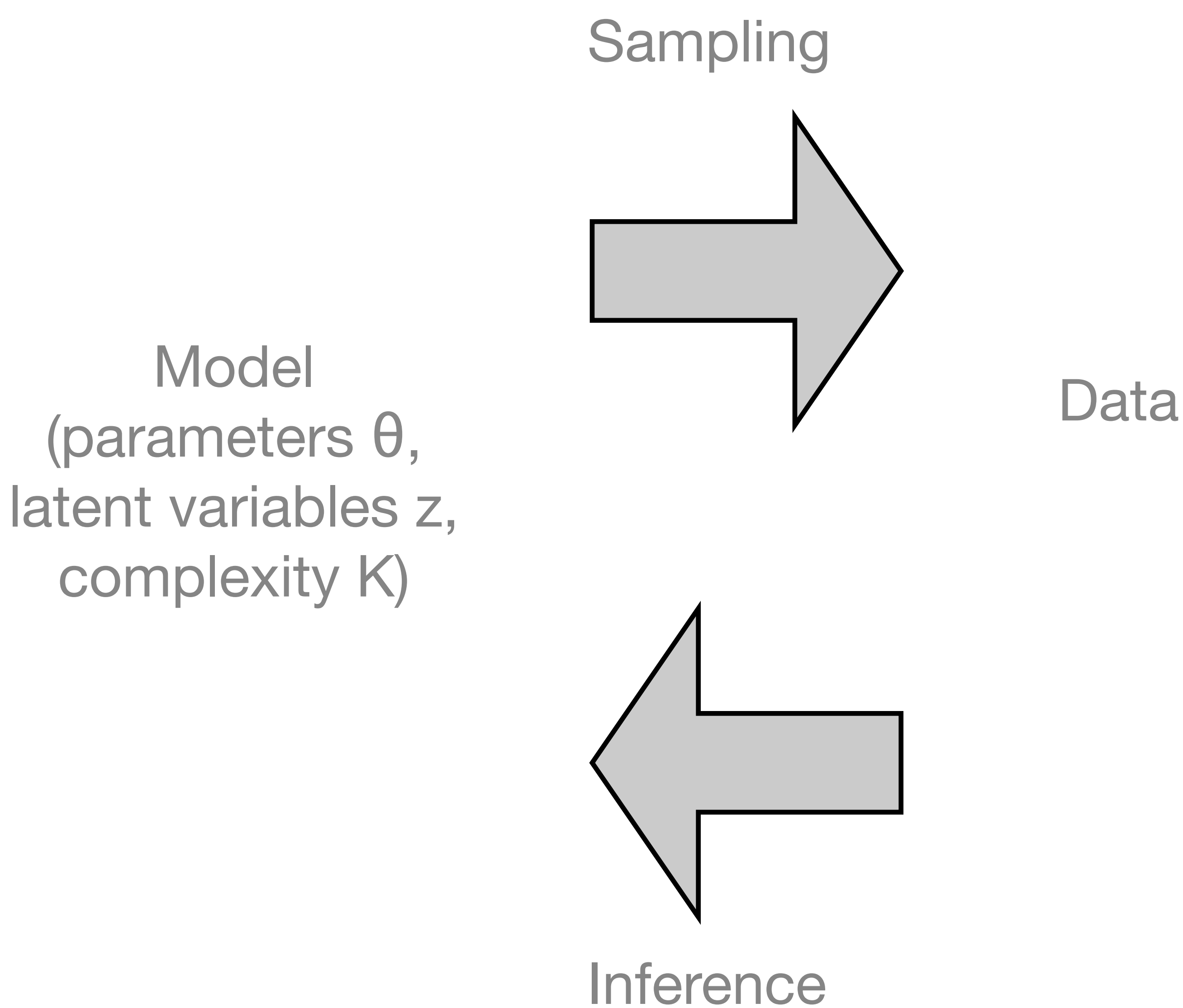


Generating modular networks

- For each node:
  - **Roll K-sided die** with bias π to determine z_i=1,...,K, the (unobserved) module assignment for i^{th} node
- For each pair of nodes (i,j):
  - If $z_i = z_j$, **flip "in community" coin** with bias θ_c to determine edge A_{ij}
  - If $z_i \neq z_j$, **flip "between communities" coin** with bias θ_d to determine edge A_{ij}



Community detection as inference

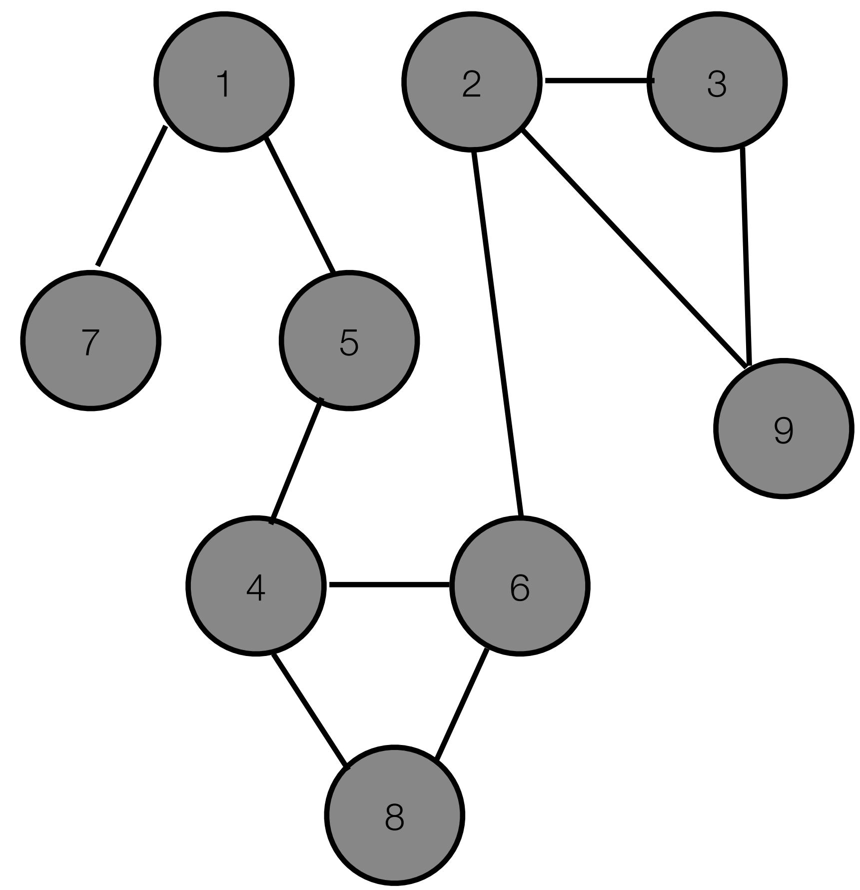


Community detection as inference

- From observed graph structure, infer distributions over module assignments, model parameters, and model complexity

$$p(\vec{\pi}, \vec{\theta} | \mathbf{A}, K) = \frac{p(\mathbf{A} | \vec{\pi}, \vec{\theta}, K) p(\vec{\pi}, \vec{\theta} | K)}{p(\mathbf{A} | K)}$$

$$p(\vec{z} | \mathbf{A}, K) = \frac{p(\mathbf{A} | \vec{z}, K) p(\vec{z} | K)}{p(\mathbf{A} | K)}$$



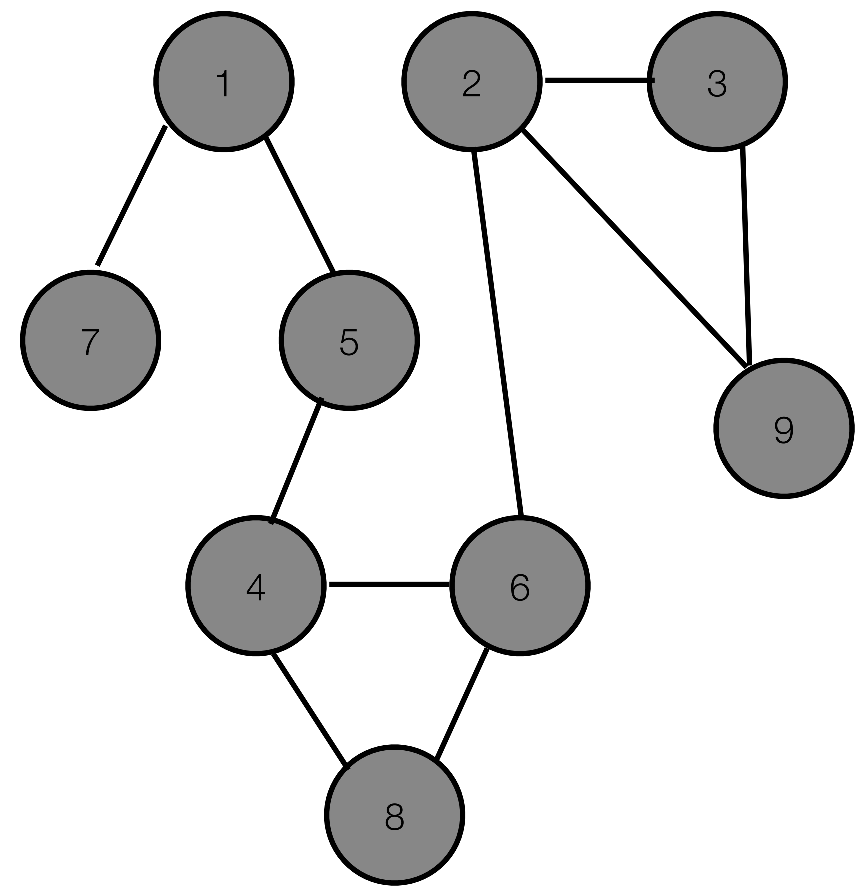
Community detection as inference

- From observed graph structure, infer distributions over module assignments, model parameters, and model complexity

$$p(\vec{\pi},\vec{\theta}|\mathbf{A},K) = \frac{p(\mathbf{A}|\vec{\pi},\vec{\theta},K)p(\vec{\pi},\vec{\theta}|K)}{p(\mathbf{A}|K)}$$

$$p(\vec{z}|\mathbf{A},K) = \frac{p(\mathbf{A}|\vec{z},K)p(\vec{z}|K)}{p(\mathbf{A}|K)}$$

Multiplication is easy, but normalization is intractable $O(K^N)$; use mean-field variational approach



Approximate inference for modular networks

- Iteratively optimize F{q;A} by updating distributions over parameters {π, θ} and latent variables {z}

Algorithm 2 Variational Bayes for maximum evidence inference

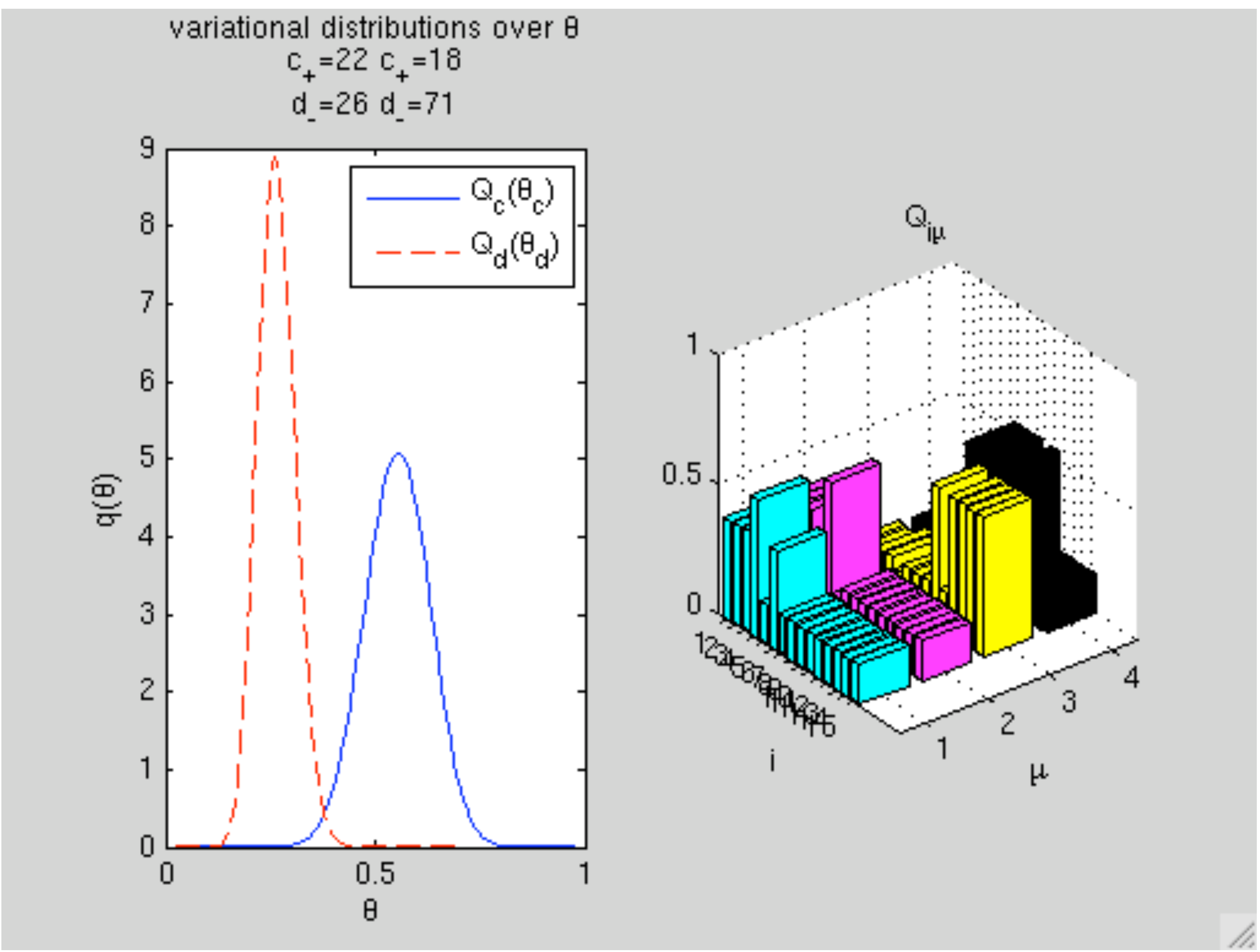
t=0
choose initial distributions $q^{(0)}(Z), q^{(0)}(\Theta)$
repeat
    E-step: calculate $\ln q^{(t+1)}(Z) \propto \langle \ln p(\mathcal{D}, Z|\Theta, K) p(\Theta|K) \rangle_{q^{(t)}(\Theta)}$
    M-step: calculate $\ln q^{(t+1)}(\Theta) \propto \langle \ln p(\mathcal{D}, Z|\Theta, K) p(\Theta|K) \rangle_{q^{(t+1)}(Z)}$
    $t \leftarrow t + 1$
until $\mathcal{F}[q^{(t+1)}(Z), q^{(t+1)}(\Theta)] - \mathcal{F}[q^{(t)}(Z), q^{(t)}(\Theta)] \leq \delta$ or $t = T_{max}$

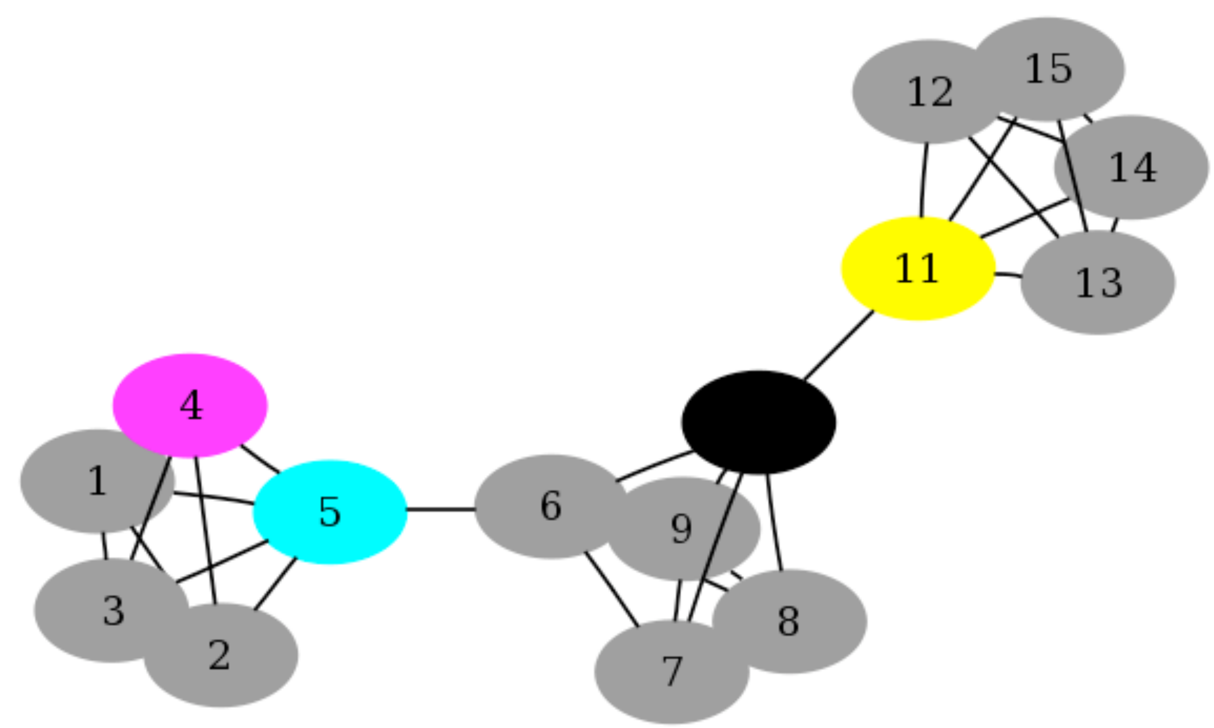
Validation: complexity control

- Automatic complexity control: probability of occupation for extraneous modules goes to zero

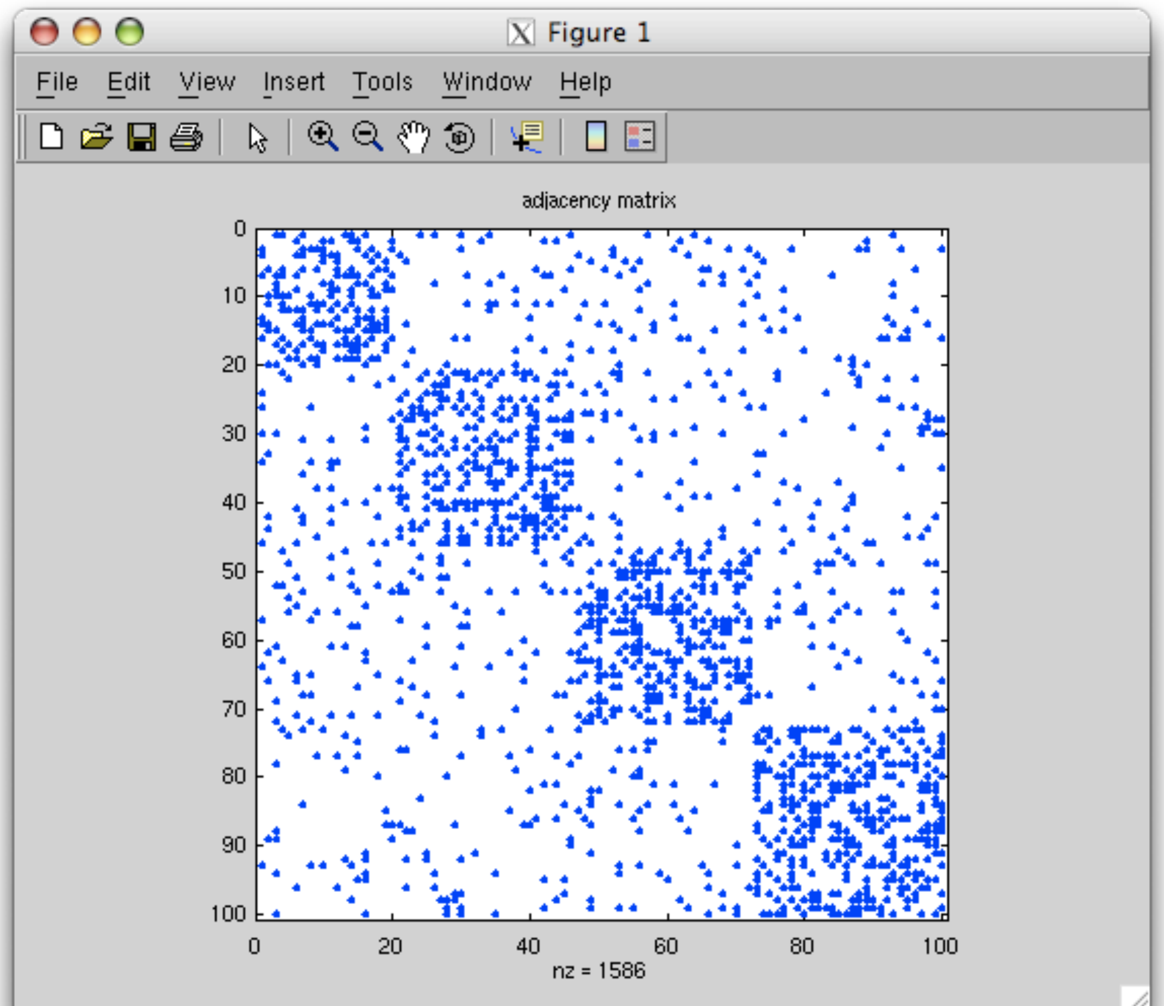
Validation: complexity control

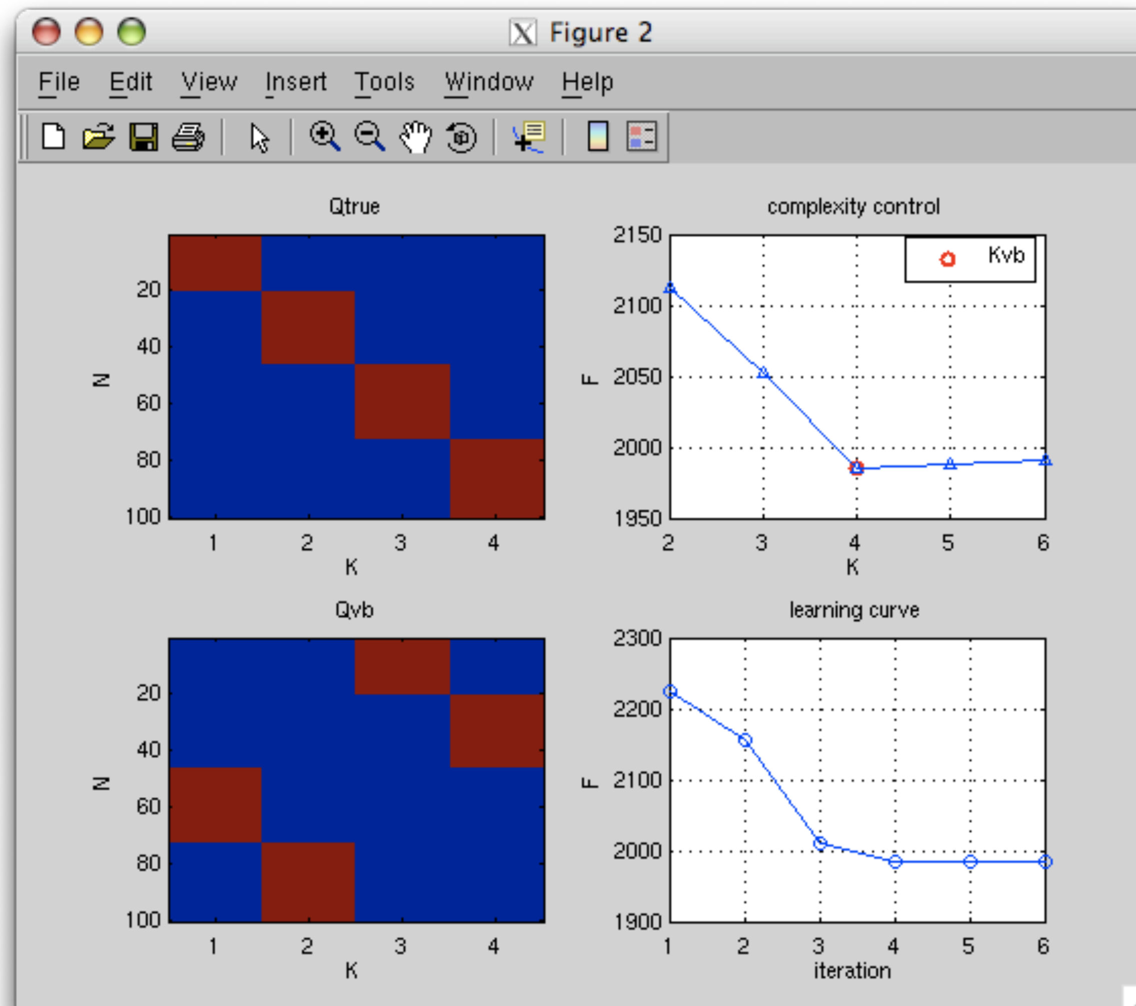
- Automatic complexity control: probability of occupation for extraneous modules goes to zero





http://vbmod.sourceforge.net

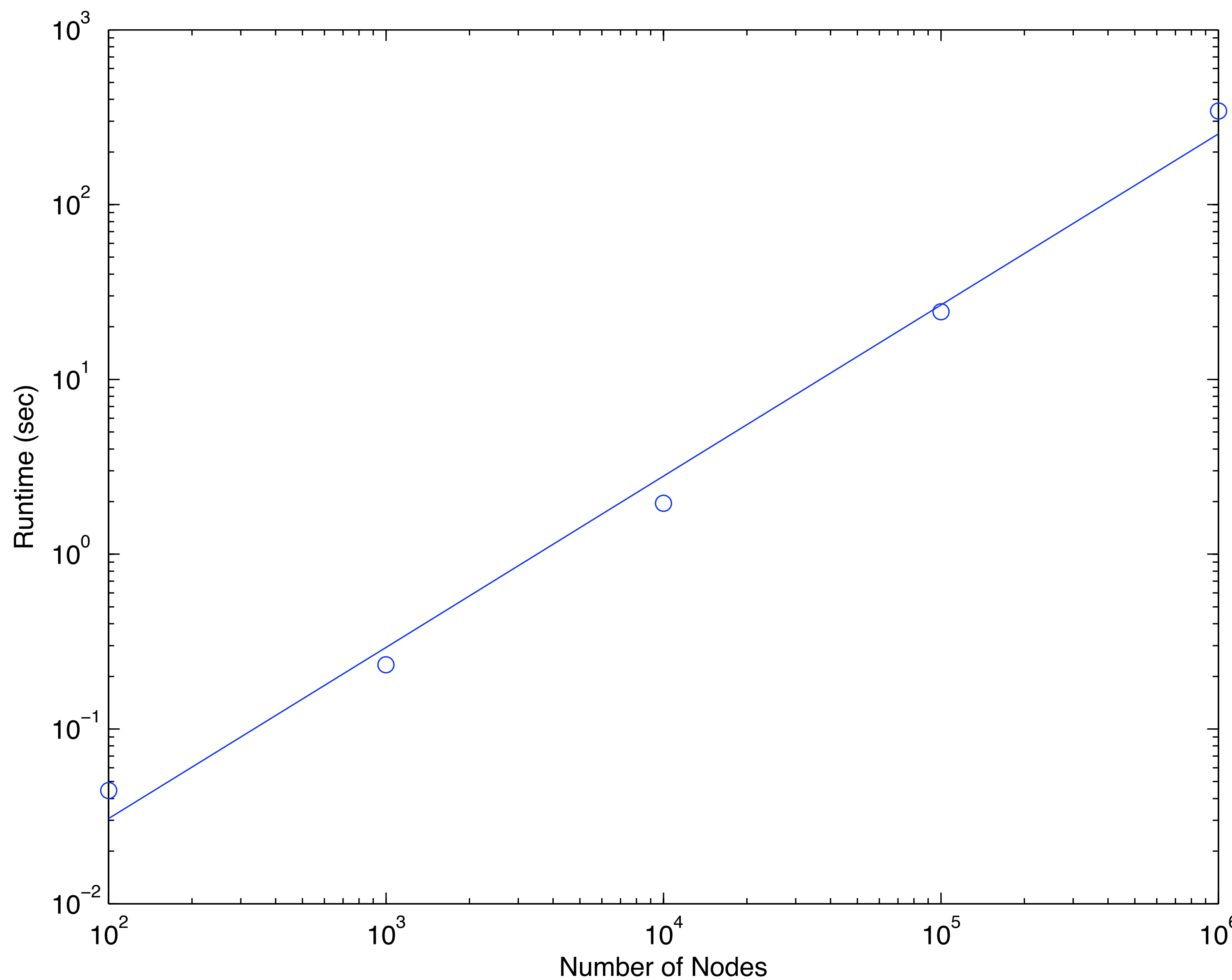




Phys. Rev. Lett. 2008 vol. 100

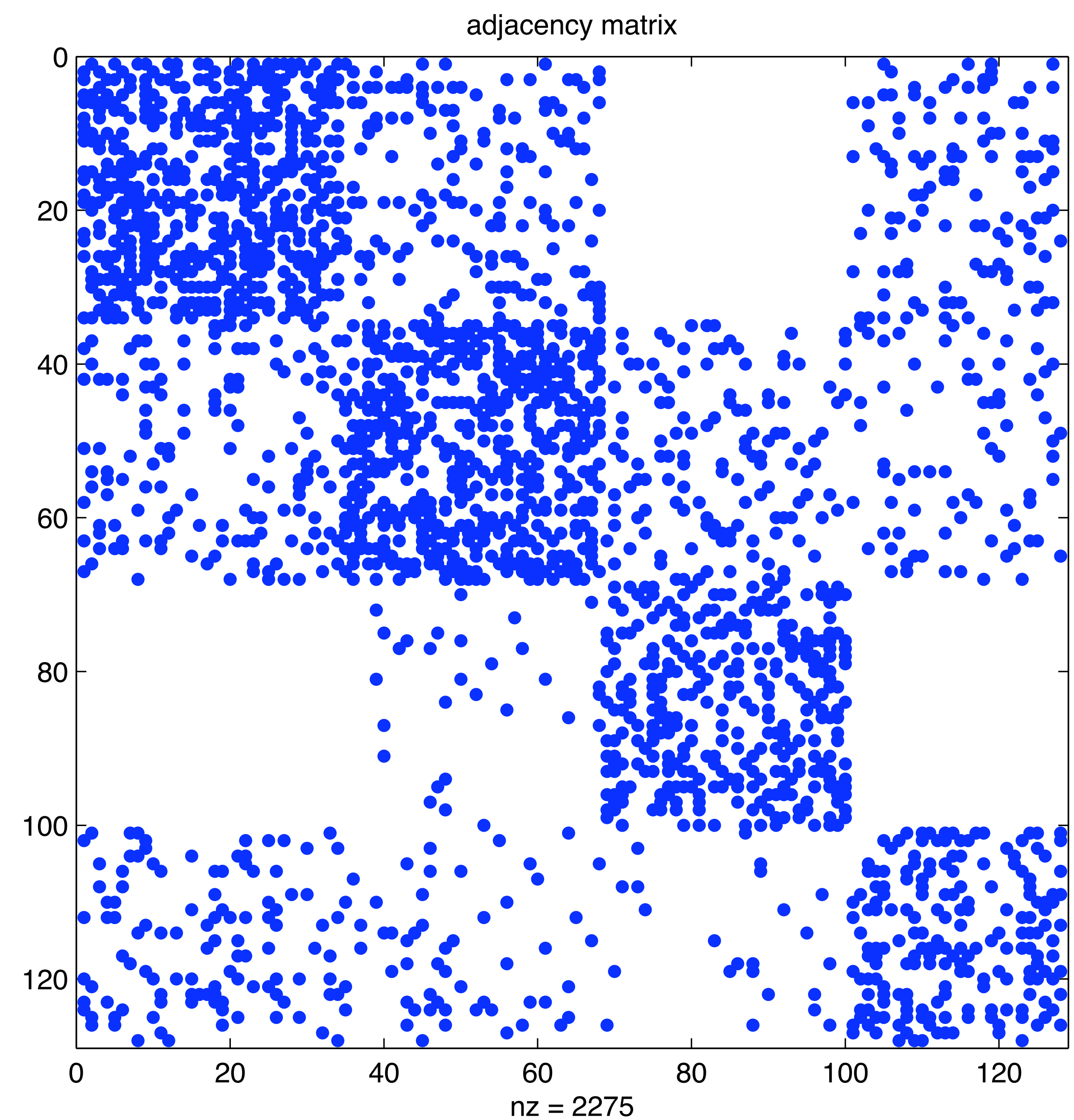
Validation: Runtime

- O(MK) runtime; ~400 sec for N=10^6 nodes, K=4 modules, average node degree 16



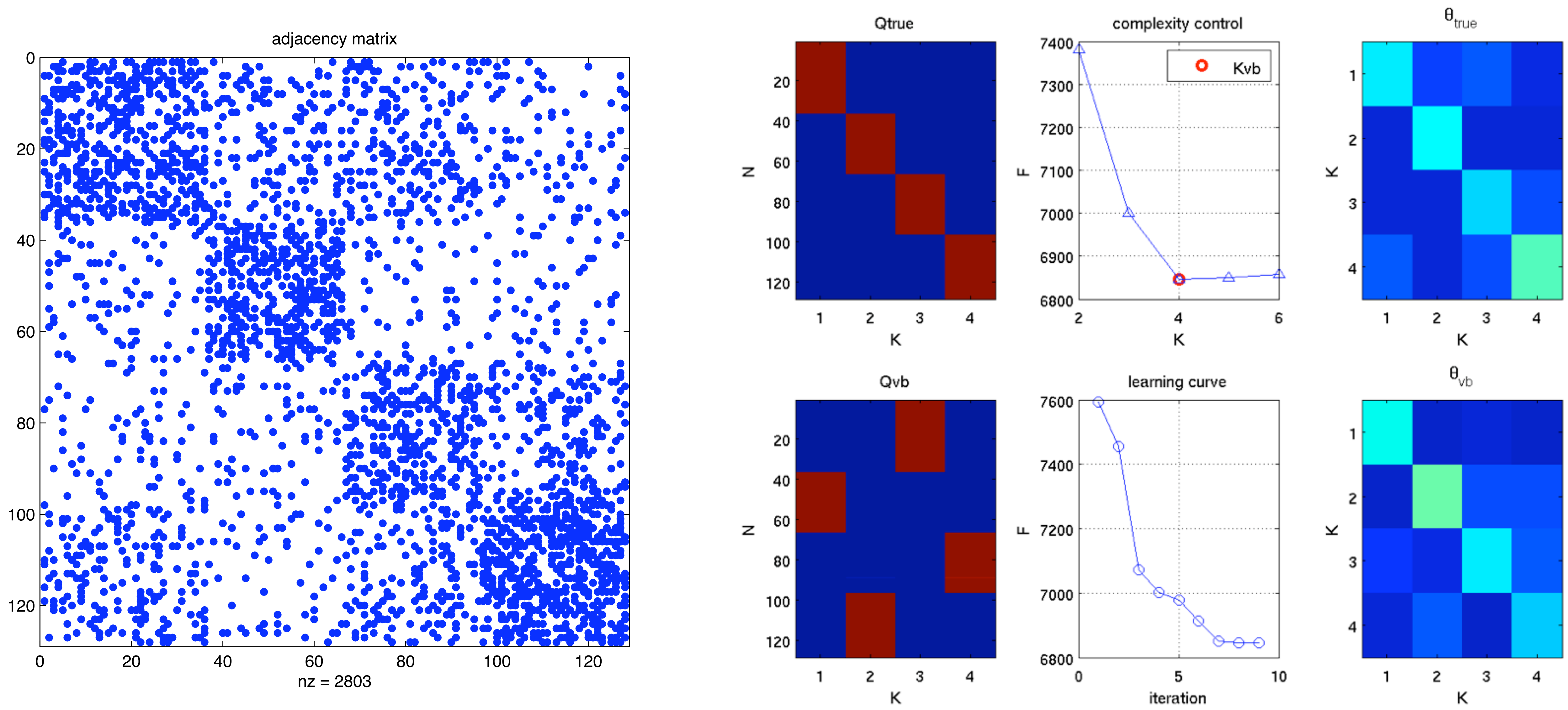
Full stochastic block models

- **Nodes belong to "blocks"** of varying size
    - Roll die for assignment of nodes to blocks
- Probability of **edge** between two nodes **depends only on block membership**
    - Flip (**one of K^2**) coins for edges
- Result: **mixture of Erdos-Renyi** graphs



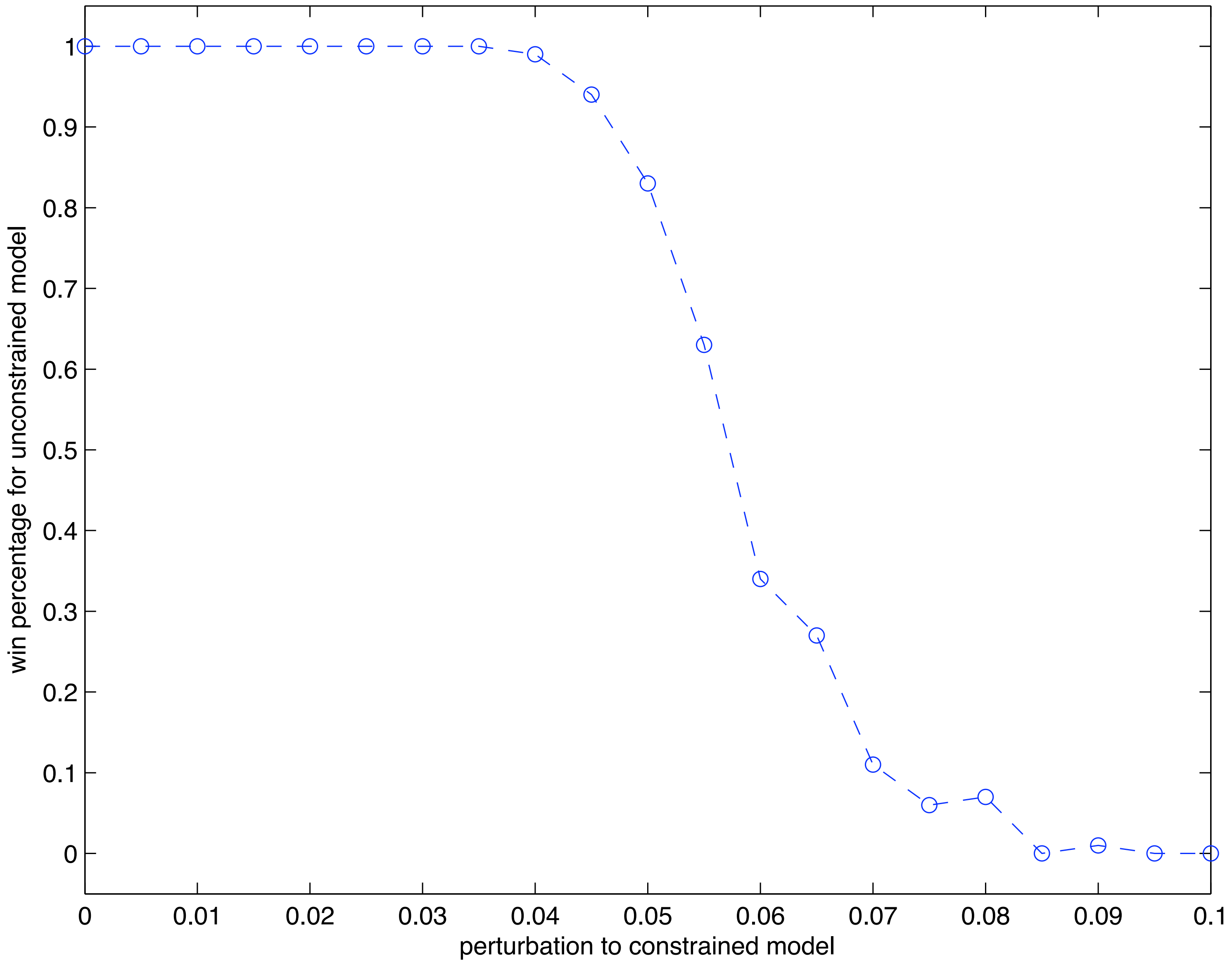
Holland, Laskey, Leinhardt 1983; Wang and Wong, 1987

Variational Bayes for stochastic block models



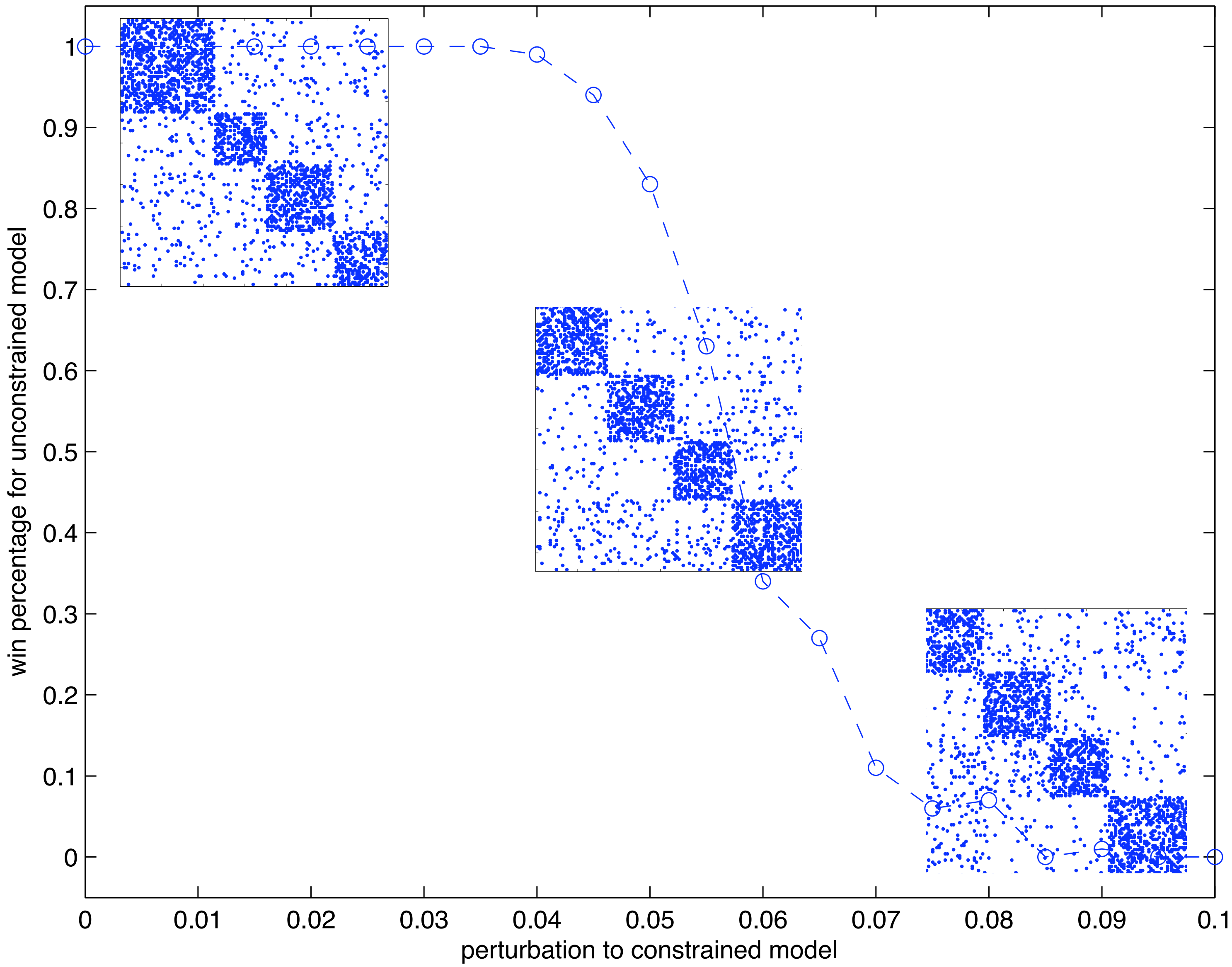
Model comparison

- Using *same framework* we can compare the constrained and full stochastic block models via p(D|M,K*)



Model comparison

- Using *same framework* we can compare the constrained and full stochastic block models via p(D|M,K*)



Application: APS March Meeting 2008 co-authorship

APS Meetings

- Login
- Create Account
- Meeting Home
- APS Home
  - Meeting Announcement
- Invited Speakers
- Author Index
- Session Index
- Epitome
- Session Chairs
- Word Search
- Affiliation Search
- Using the Scheduler
- BAPS PDFs

2008 APS March Meeting

Monday–Friday, March 10–14, 2008; New Orleans, Louisiana

Session P39: Applications of Complex Networks

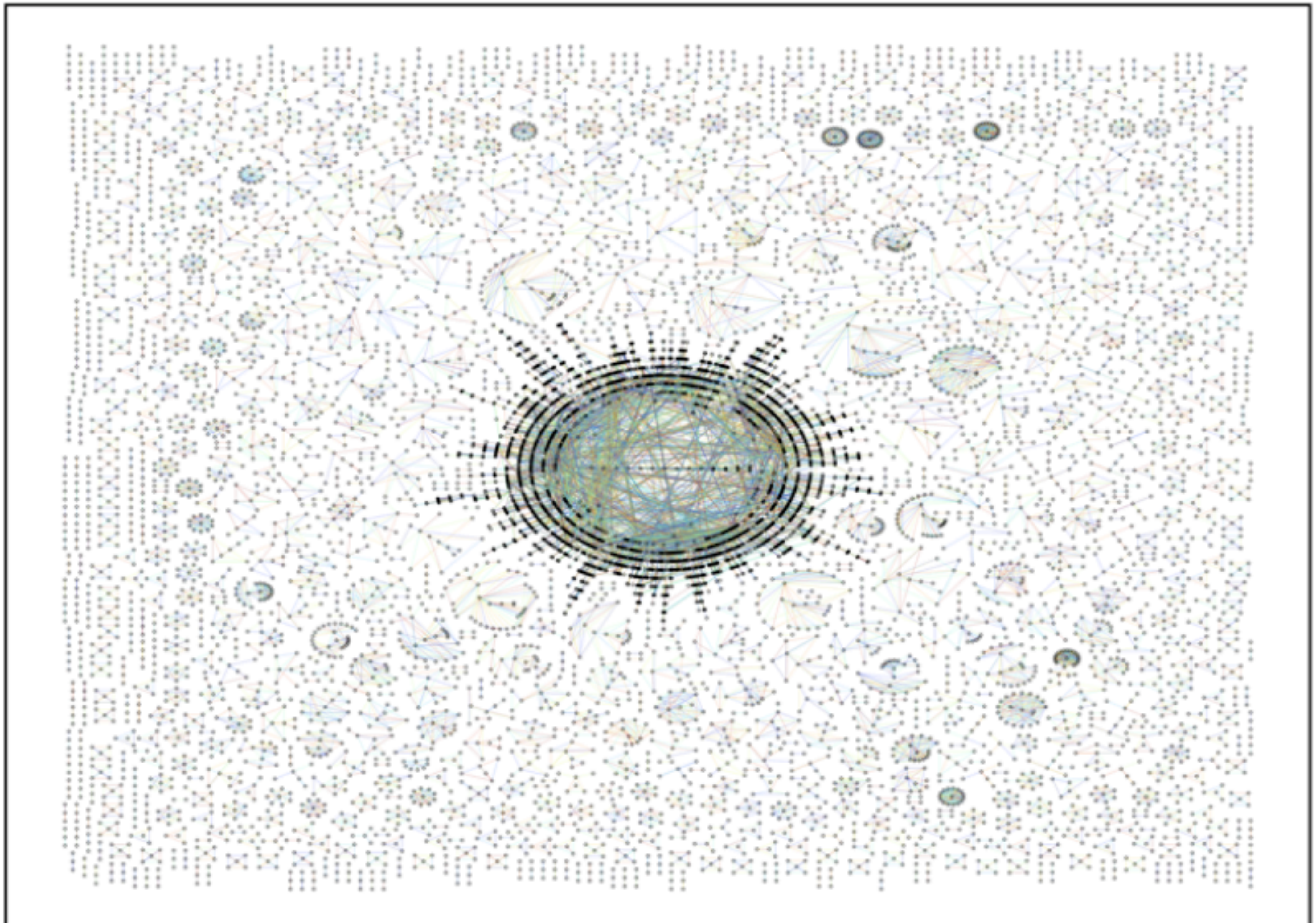
Show Abstracts

Sponsoring Units: GSNP
Chair: Narayan Menon, University of Massachusetts, Amherst
Morial Convention Center - 231

Wednesday, March 12, 2008 8:00AM - 8:12AM	P39.00001: Effects of quenched randomness on predator-prey interactions in a stochastic Lotka-Volterra lattice model Uwe C. Tauber , Ulrich Dobramysl Preview Abstract
Wednesday, March 12, 2008 8:12AM - 8:24AM	P39.00002: Dynamical Clustering in Reaction-Dispersal Processes on Complex Networks Vincent David , Marc Timme , Theo Geisel , Dirk Brockmann Preview Abstract
Wednesday, March 12, 2008 8:24AM - 8:36AM	P39.00003: Fluctuations and Food-web Structures in Individual-based Models of Biological Coevolution Per Arne Rikvold , Volkan Sevim Preview Abstract
Wednesday, March 12, 2008 8:36AM - 8:48AM	P39.00004: Metabolic disease network and its implication for disease comorbidity Deok-Sun Lee , Zoltan Oltvai , Nicholas Christakis , Albert-Laszlo Barabasi Preview Abstract
Wednesday, March 12, 2008 8:48AM - 9:00AM	P39.00005: The Human Phenotypic Disease Network Cesar Hidalgo , Nicholas Blumm , Albert-Laszlo Barabasi , Nicholas Christakis Preview Abstract

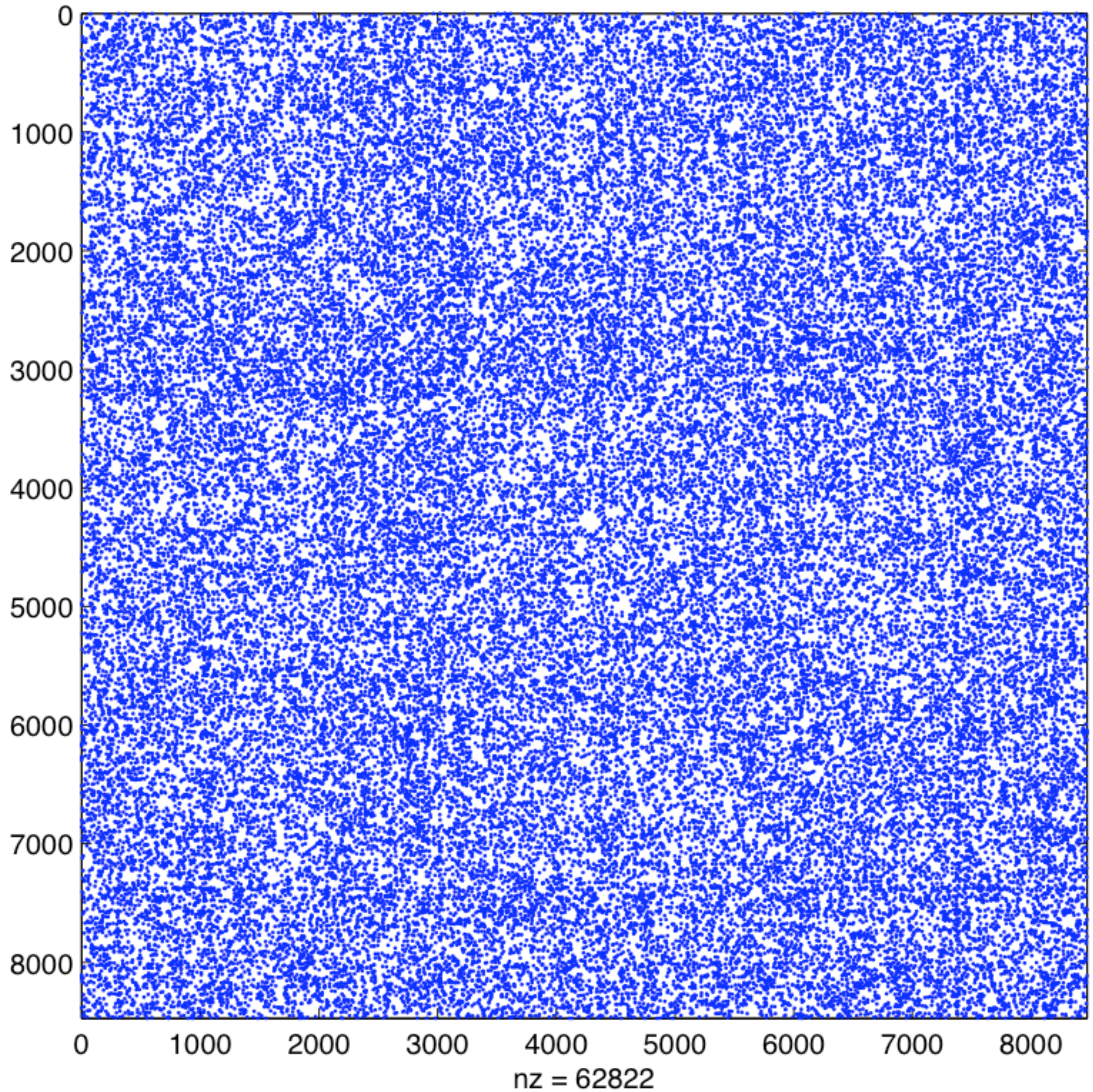
http://meetings.aps.org/Meeting/MAR08

Application: APS March Meeting 2008 co-authorship

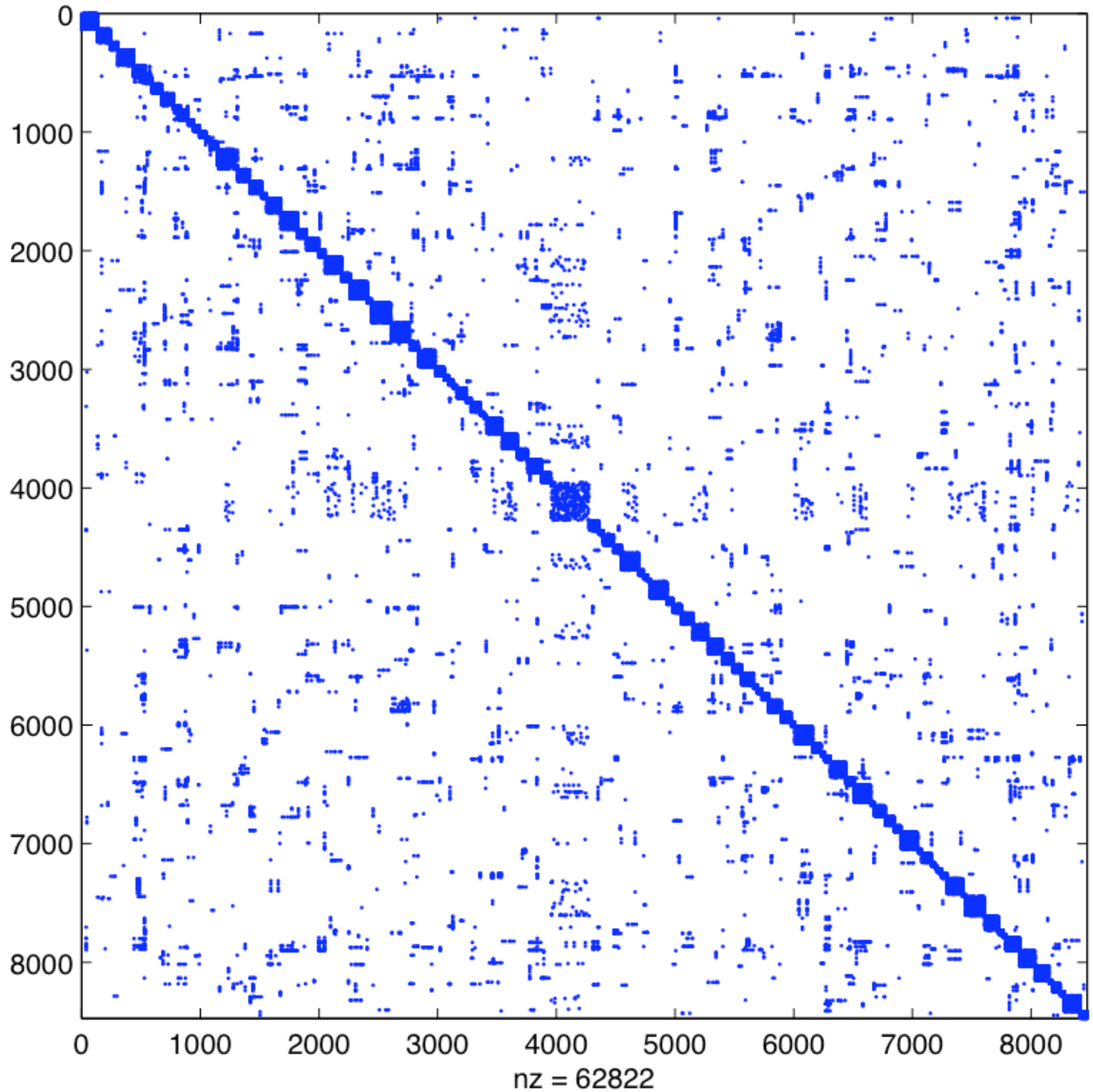


nodes: authors
edges: co-authored papers

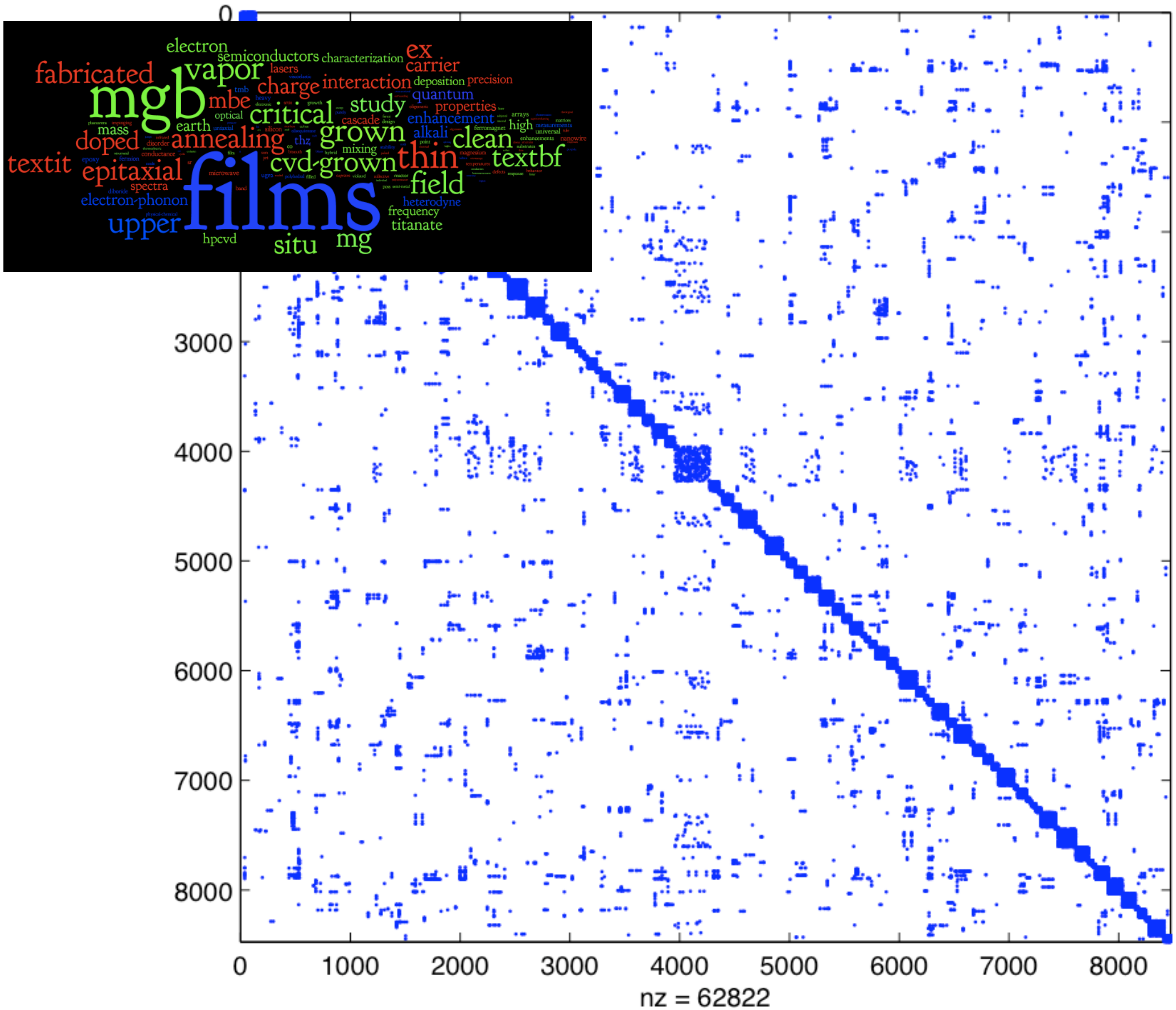
APS March Meeting 2008 co-authorship network



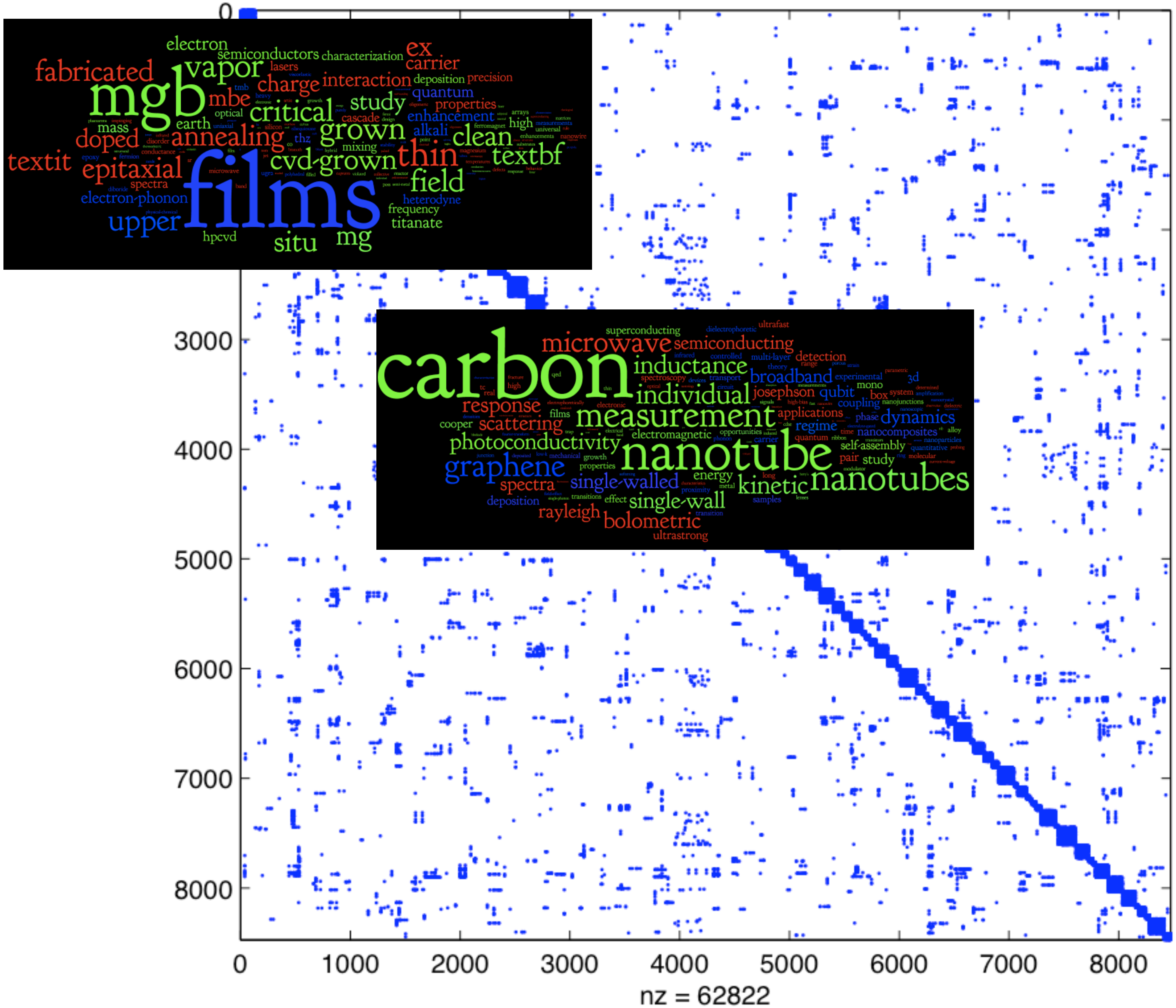
APS March Meeting 2008 co-authorship network



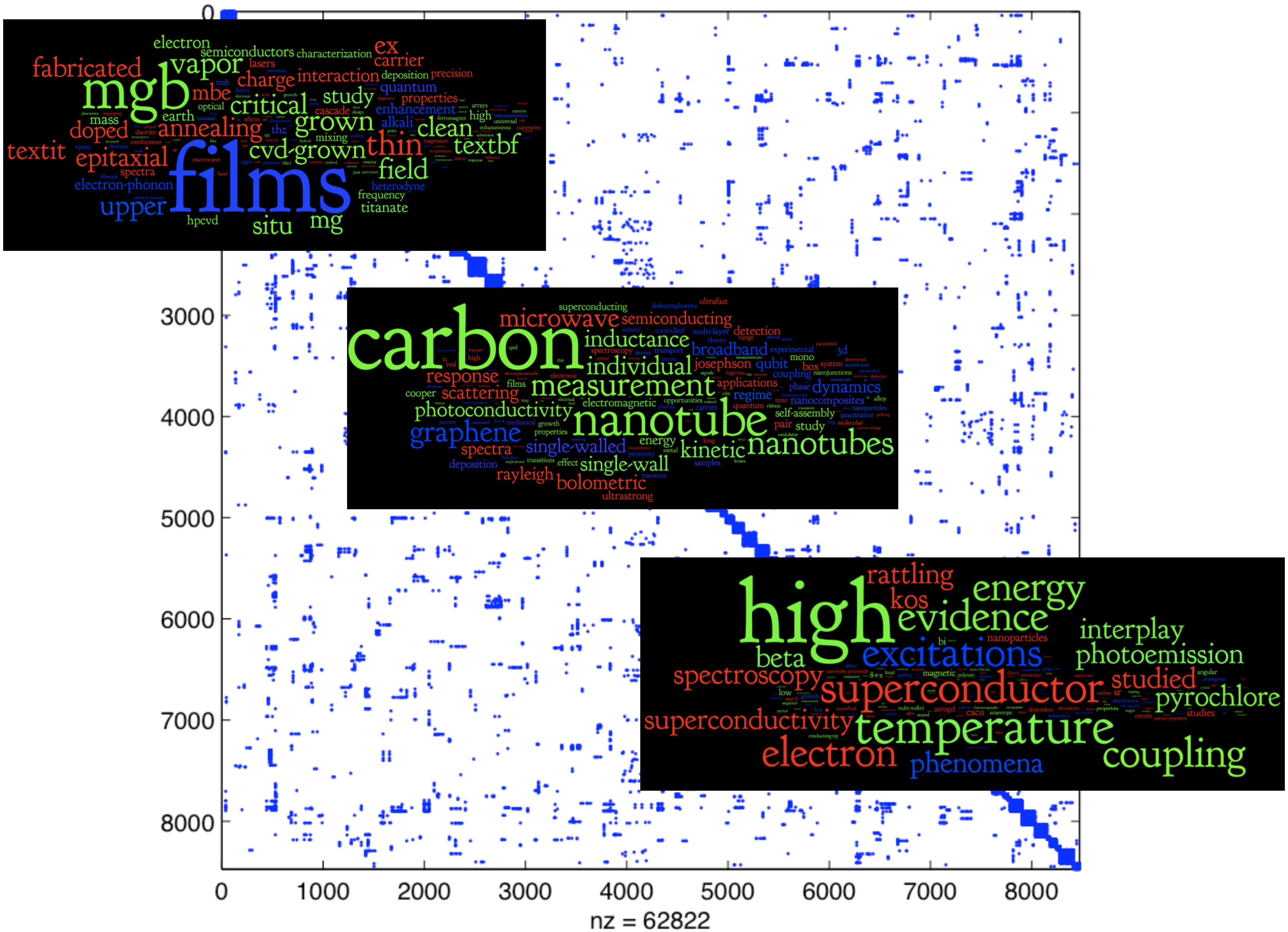
APS March Meeting 2008 co-authorship network



APS March Meeting 2008 co-authorship network

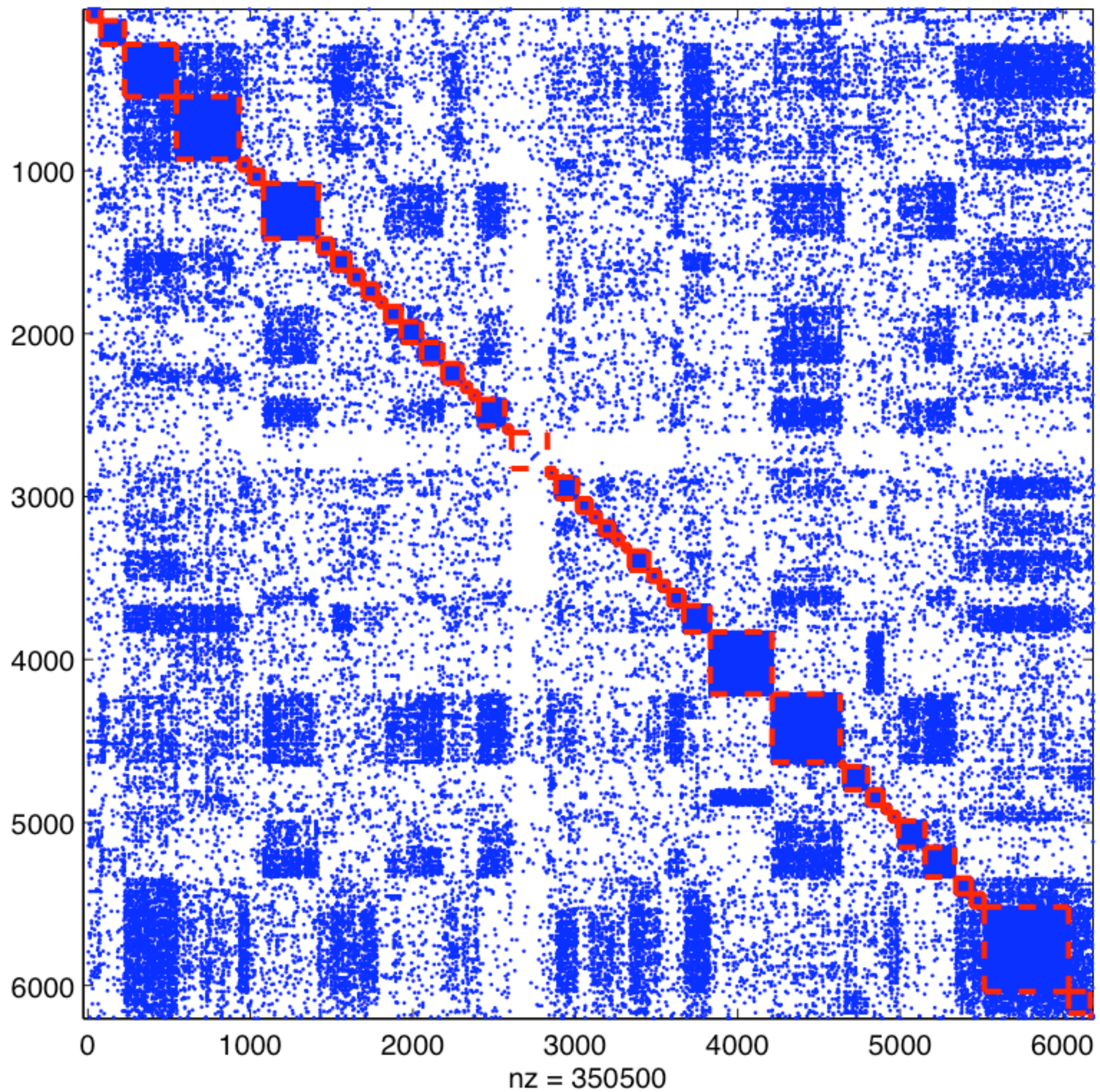


APS March Meeting 2008 co-authorship network



University email data set

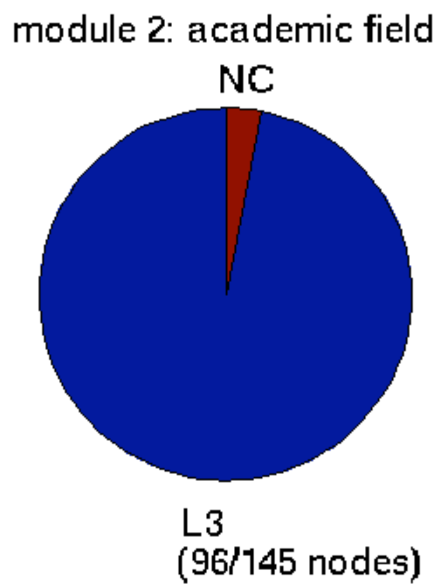
- How does topology (who emails whom) correspond to attributes (age, gender, academic affiliation, etc.)?

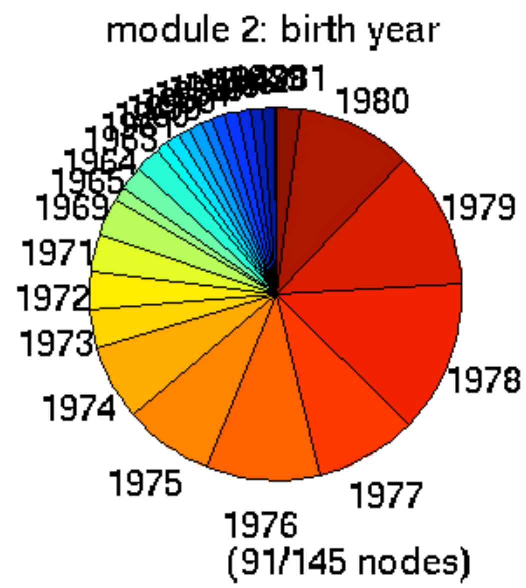


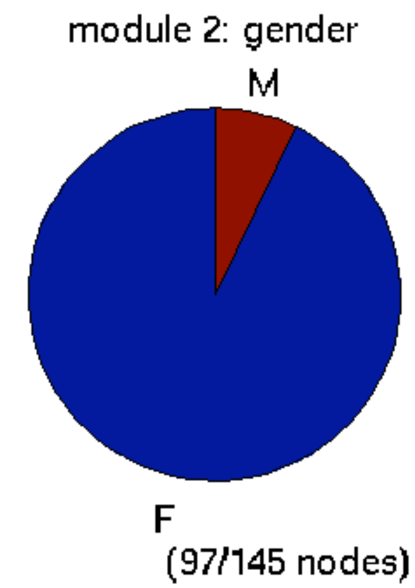
nodes: individuals
edges: reciprocated emails

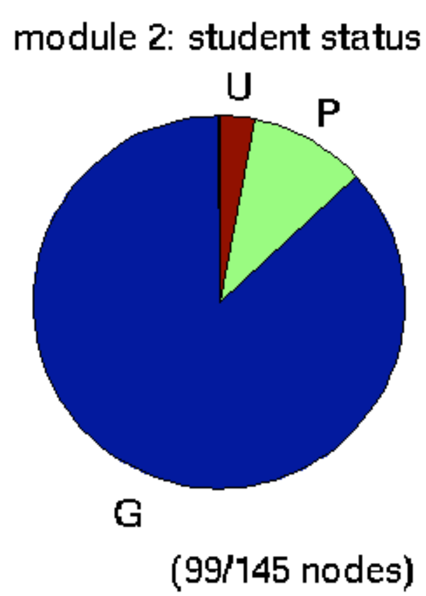
University email data set

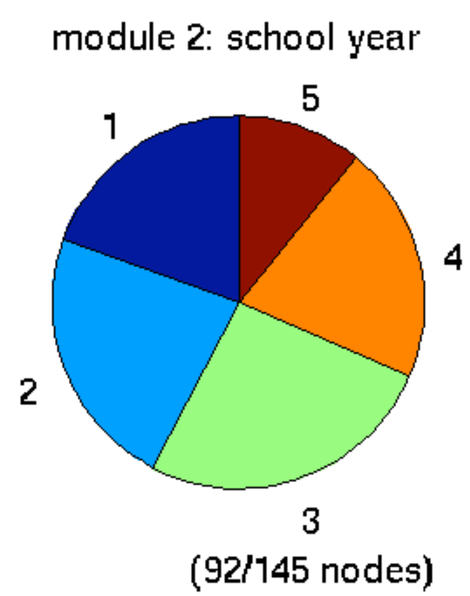
- Module 2: Female graduate students in same major (code L3), across all years







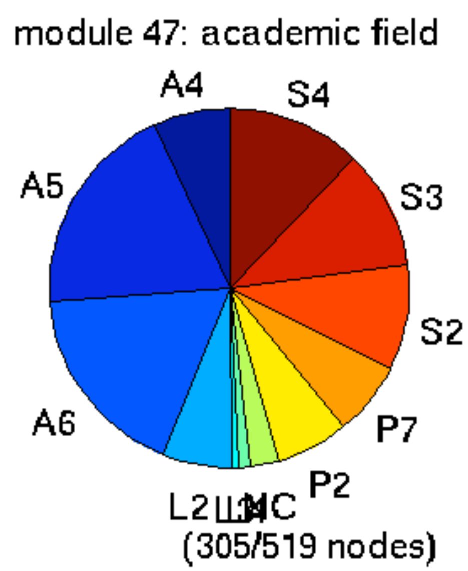


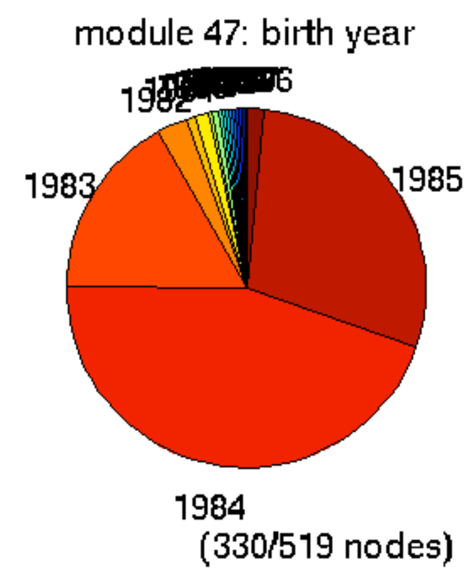


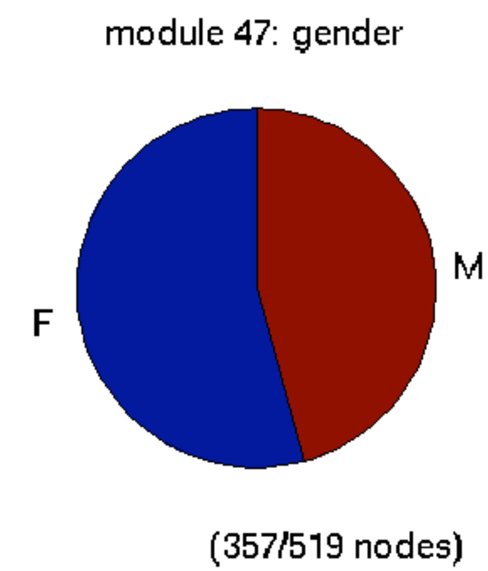
module 2 stats:
nodes: 145
edges: 917
edge density: 0.0878352

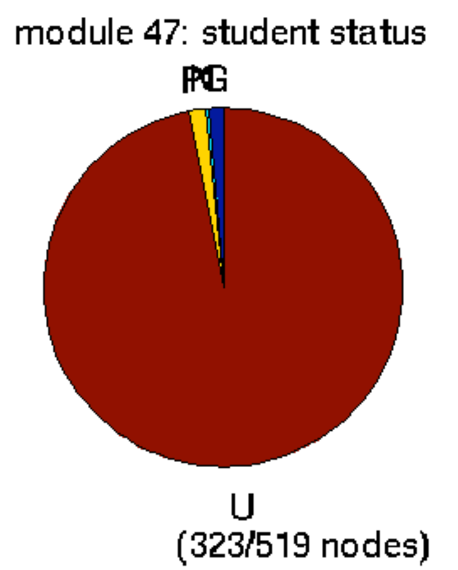
University email data set

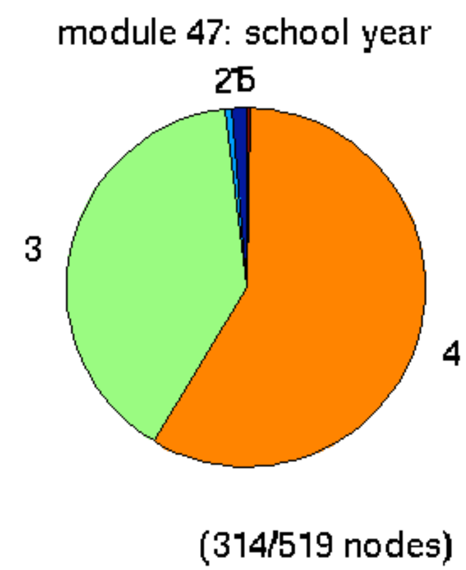
- Module 47: Junior and senior undergraduates of various majors







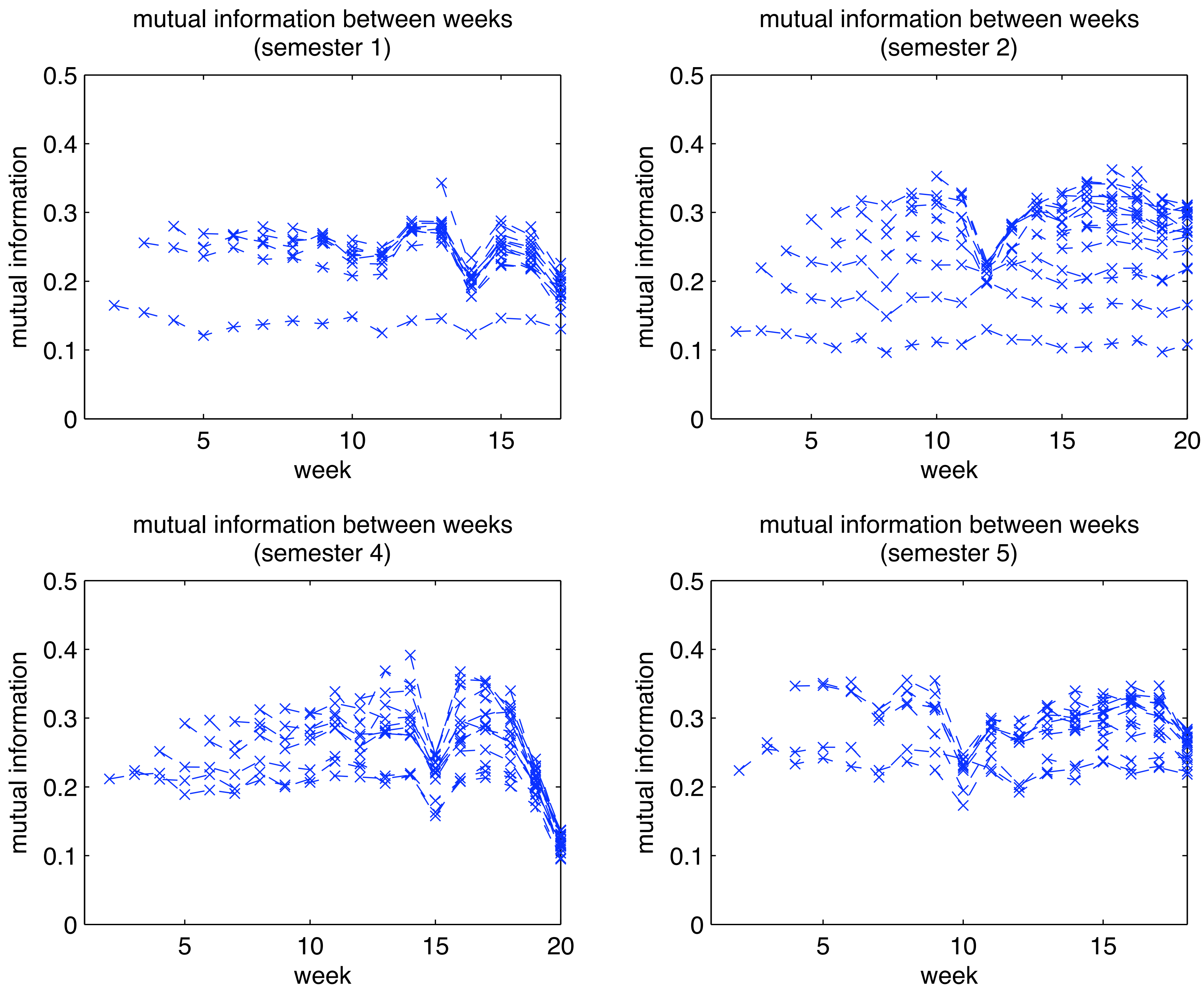




module 47 stats:
nodes: 519
edges: 6474
edge density: 0.0481621

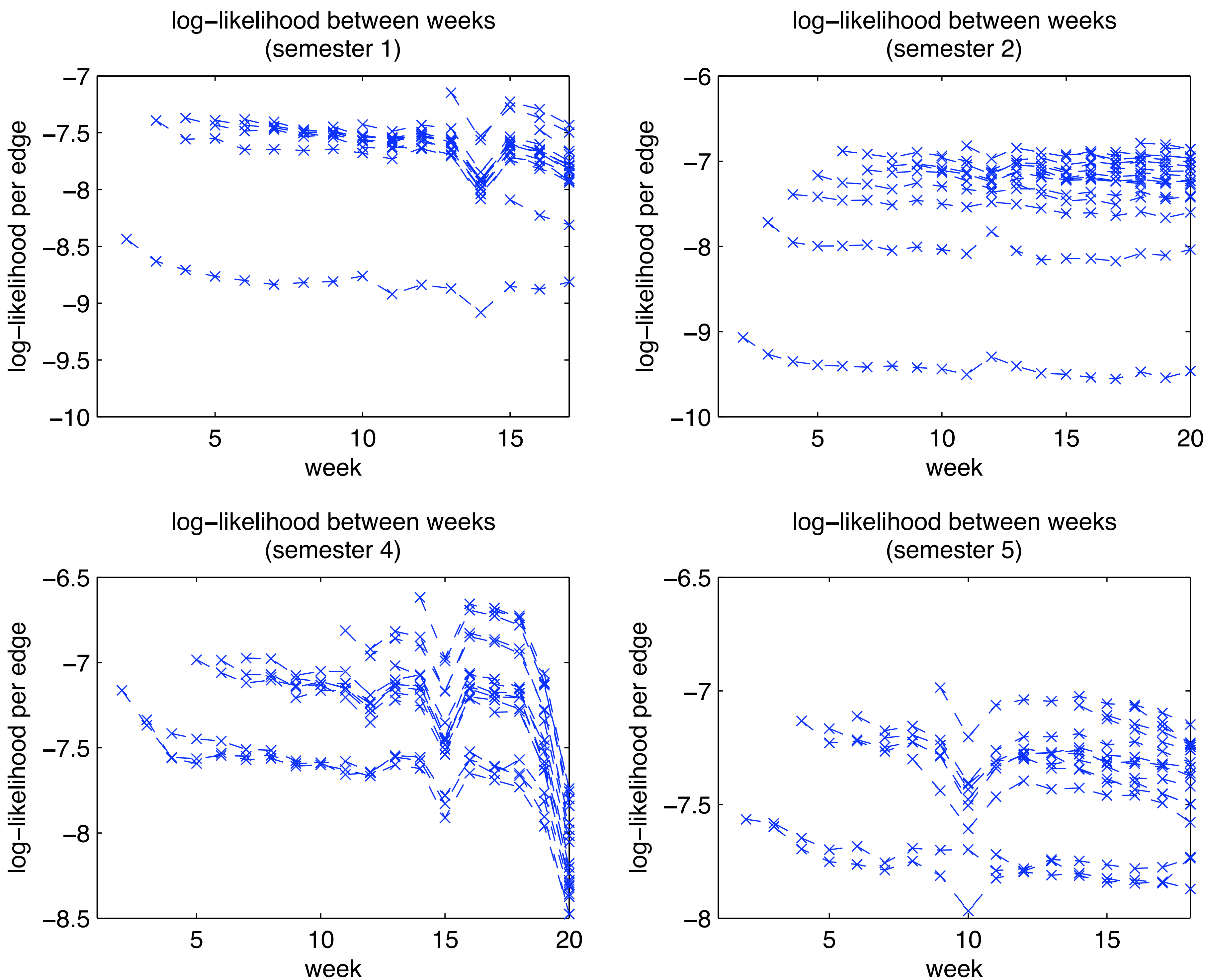
University email data set

- Compare cluster assignments between weeks



University email data set

- Compare model performance as trained/tested on different weeks



Conclusions

- Phrased community detection as Bayesian inference
- Implemented model selection and comparison for constrained and full stochastic block models in variational framework
- Some correlation between topology and attributes, but unclear without additional information
- Non-trivial dynamic evolution of community structure
- References: http://vbmod.sf.net, Phys. Rev. Lett. Vol 100 (2008)

Acknowledgements

- **Wiggins Lab**
  - Andrew Mugler
  - Anil Raj
  - Chris Wiggins
- **Yahoo! Research**
  - Duncan Watts
- **Useful discussions**
  - Edo Airoldi (Princeton)
  - Joel Bader (John Hopkins)
  - David Blei (Princeton)
  - Aaron Clauset (SFI)
  - Jonathan Goodman (NYU)
  - Matt Hastings (LANL)