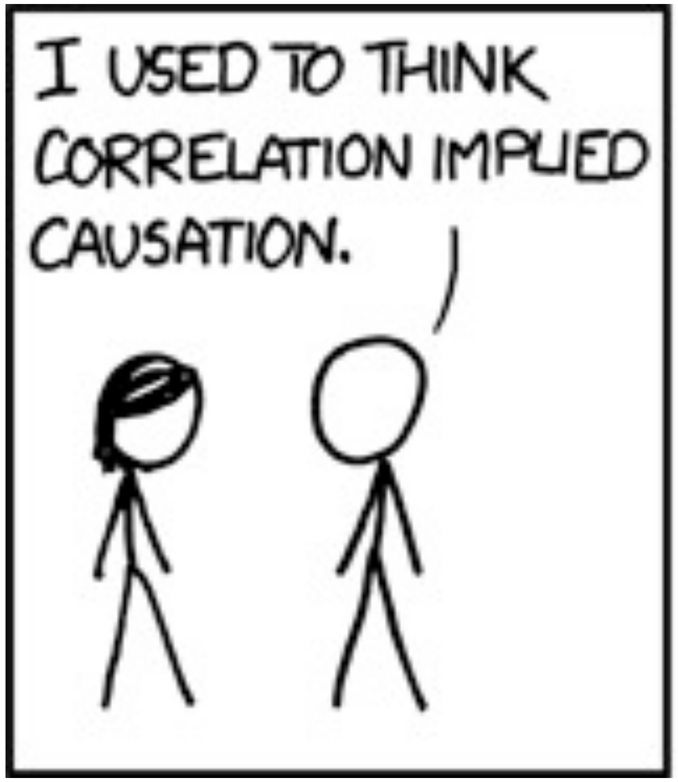
Inference in networks

Cristopher Moore, Santa Fe Institute

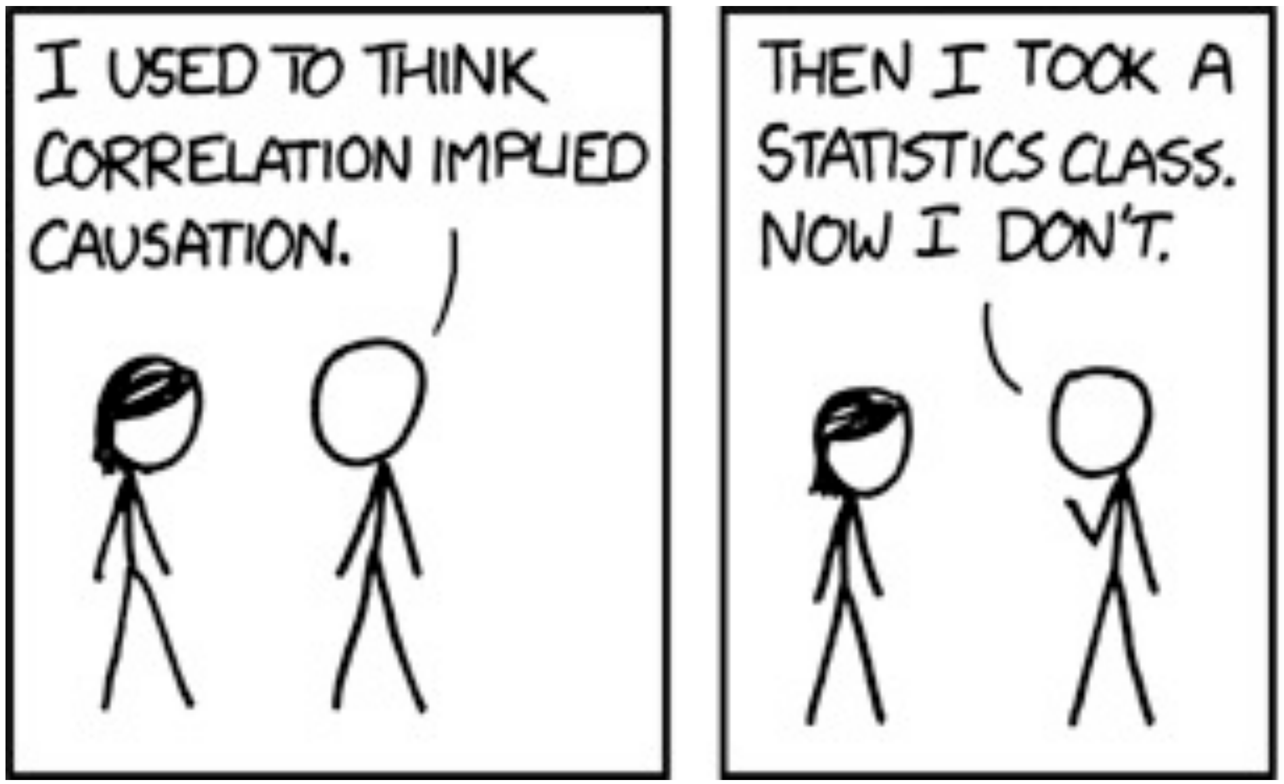
joint work with
Aaron Clauset, Mark Newman,
Xiaoran Yan, Yaojia Zhu,
Lenka Zdeborová, Florent Krzakala,
Aurelien Decelle, Pan Zhang,
Jean-Baptiste Rouquier, and Tiffany Pierce

Learning statistics

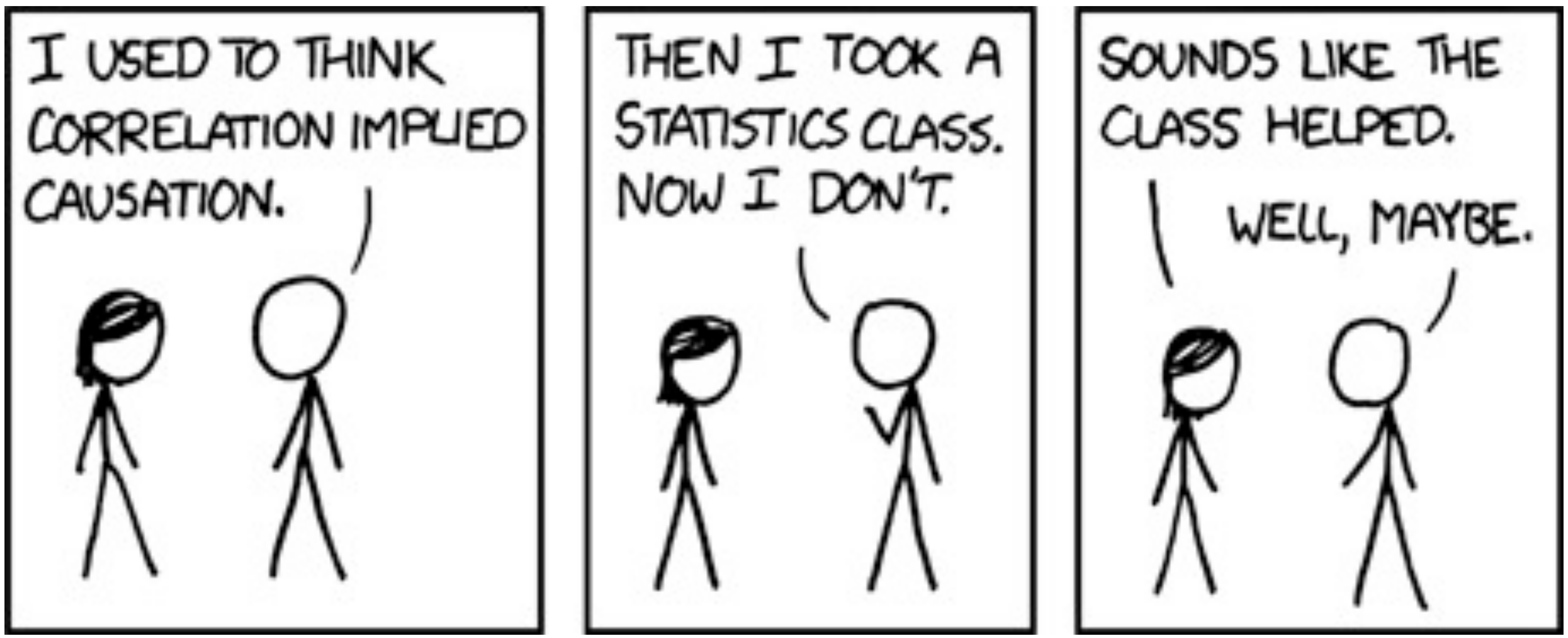
Learning statistics



Learning statistics



Learning statistics



What is structure?

What is structure?

Structure is that which...

What is structure?

Structure is that which...

makes data different from noise: makes a network different from a random graph

What is structure?

Structure is that which...

makes data different from noise: makes a network different from a random graph

helps us compress the data: describe the network succinctly

What is structure?

Structure is that which...

makes data different from noise: makes a network different from a random graph

helps us compress the data: describe the network succinctly

helps us generalize from data we've seen from data we haven't seen:

What is structure?

Structure is that which...

makes data different from noise: makes a network different from a random graph

helps us compress the data: describe the network succinctly

helps us generalize from data we've seen from data we haven't seen:

from one part of the network to another,

What is structure?

Structure is that which...

makes data different from noise: makes a network different from a random graph

helps us compress the data: describe the network succinctly

helps us generalize from data we’ve seen from data we haven’t seen:

from one part of the network to another,

or from one network to others generated by the same process

What is structure?

Structure is that which...

makes data different from noise: makes a network different from a random graph

helps us compress the data: describe the network succinctly

helps us generalize from data we've seen from data we haven't seen:

from one part of the network to another,

or from one network to others generated by the same process

helps us coarse-grain the dynamics (?)

What is structure?

Structure is that which...

makes data different from noise: makes a network different from a random graph

helps us compress the data: describe the network succinctly

helps us generalize from data we've seen from data we haven't seen:

from one part of the network to another,

or from one network to others generated by the same process

helps us coarse-grain the dynamics (?)

What is structure?

Structure is that which...

makes data different from noise: makes a network different from a random graph

helps us compress the data: describe the network succinctly

helps us generalize from data we've seen from data we haven't seen:

from one part of the network to another,

or from one network to others generated by the same process

helps us coarse-grain the dynamics (?)

Statistical inference

Statistical inference

imagine that our data G is drawn from an ensemble, or "generative model": some probability distribution $P(G|\theta)$ with parameters θ

Statistical inference

imagine that our data G is drawn from an ensemble, or “generative model”: some probability distribution $P(G|\theta)$ with parameters θ

θ can be continuous or discrete: represents the structure of the data

Statistical inference

imagine that our data G is drawn from an ensemble, or “generative model”: some probability distribution $P(G|\theta)$ with parameters θ

θ can be continuous or discrete: represents the structure of the data

given G, find the θ that maximize $P(G|\theta)$

Statistical inference

imagine that our data G is drawn from an ensemble, or “generative model”: some probability distribution $P(G|\theta)$ with parameters θ

θ can be continuous or discrete: represents the structure of the data

given G, find the θ that maximize $P(G|\theta)$

or (Bayes) compute the posterior distribution $P(\theta|G)$

Statistical inference

imagine that our data G is drawn from an ensemble, or "generative model": some probability distribution $P(G|\theta)$ with parameters θ

θ can be continuous or discrete: represents the structure of the data

given G, find the θ that maximize $P(G|\theta)$

or (Bayes) compute the posterior distribution $P(\theta|G)$

The Erdős-Renyí model

The Erdős-Renyí model

every pair of vertices *i, j* is connected independently with probability *p*

The Erdős-Renyí model

every pair of vertices *i, j* is connected independently with probability *p*

average degree $d=np$

The Erdős-Renyí model

every pair of vertices *i, j* is connected independently with probability *p*

average degree $d=np$

degree distribution is Poisson with mean d

The Erdős-Renyí model

every pair of vertices *i, j* is connected independently with probability *p*

average degree $d=np$

degree distribution is Poisson with mean *d*

if $d<1$, almost all components are trees, and max component has size O(log n)

The Erdős-Renyí model

every pair of vertices *i, j* is connected independently with probability *p*

average degree *d=np*

degree distribution is Poisson with mean *d*

if *d*<1, almost all components are trees, and max component has size O(log n)

if *d*>1, a unique giant component appears

The Erdős-Renyí model

every pair of vertices *i, j* is connected independently with probability *p*

average degree $d=np$

degree distribution is Poisson with mean *d*

if $d<1$, almost all components are trees, and max component has size O(log n)

if $d>1$, a unique giant component appears

at $d=\ln n$, completely conected

The Erdős-Renyí model

every pair of vertices *i, j* is connected independently with probability *p*

average degree $d=np$

degree distribution is Poisson with mean *d*

if $d<1$, almost all components are trees, and max component has size O(log n)

if $d>1$, a unique giant component appears

at $d=\ln n$, completely conected

ring + Erdős-Renyí = Watts-Strogatz

The Erdős-Renyí model

every pair of vertices *i, j* is connected independently with probability *p*

average degree $d=np$

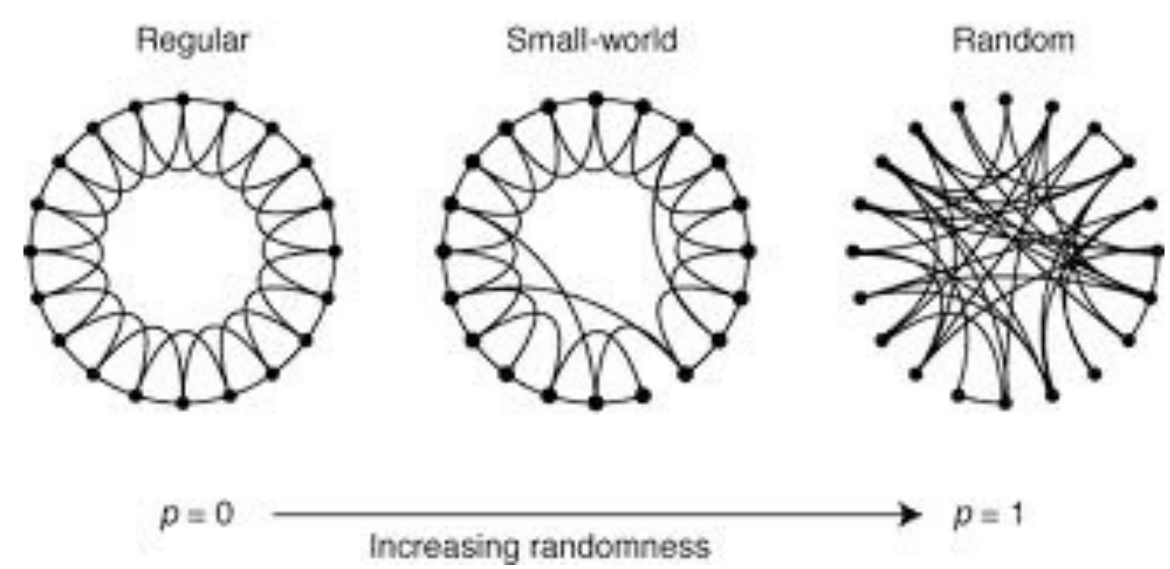
degree distribution is Poisson with mean *d*

if $d<1$, almost all components are trees, and max component has size O(log n)

if $d>1$, a unique giant component appears

at $d=\ln n$, completely conected

ring + Erdős-Renyí = Watts-Strogatz



The stochastic block model

The stochastic block model

take a discrete attribute into account:

The stochastic block model

take a discrete attribute into account:

each vertex i has a type $t_i \in \{1,...,k\}$, with prior distribution $q_1,...,q_k$

The stochastic block model

take a discrete attribute into account:

each vertex *i* has a type $t_i \in \{1,...,k\}$, with prior distribution $q_1,...,q_k$

$k \times k$ matrix p

The stochastic block model

take a discrete attribute into account:

each vertex i has a type $t_i \in \{1,...,k\}$, with prior distribution $q_1,...,q_k$

$k \times k$ matrix p

if $t_i = r$ and $t_j = s$, there is an edge $i \rightarrow j$ with probability p_{rs}

The stochastic block model

take a discrete attribute into account:

each vertex i has a type $t_i \in \{1,...,k\}$, with prior distribution $q_1,...,q_k$

$k \times k$ matrix p

if $t_i = r$ and $t_j = s$, there is an edge $i \rightarrow j$ with probability p_{rs}

p is not necessarily symmetric, and we don't assume that $p_{rr} > p_{rs}$

The stochastic block model

take a discrete attribute into account:

each vertex i has a type $t_i \in \{1,...,k\}$, with prior distribution $q_1,...,q_k$

$k \times k$ matrix p

if $t_i = r$ and $t_j = s$, there is an edge $i \rightarrow j$ with probability p_{rs}

p is not necessarily symmetric, and we don't assume that $p_{rr} > p_{rs}$

given a graph G, we want to simultaneously

The stochastic block model

take a discrete attribute into account:

each vertex i has a type $t_i \in \{1,...,k\}$, with prior distribution $q_1,...,q_k$

$k \times k$ matrix p

if $t_i = r$ and $t_j = s$, there is an edge $i \rightarrow j$ with probability p_{rs}

p is not necessarily symmetric, and we don't assume that $p_{rr} > p_{rs}$

given a graph G, we want to simultaneously

label the nodes, i.e., infer the type assignment $t : V \rightarrow \{1,...,k\}$

The stochastic block model

take a discrete attribute into account:

each vertex i has a type $t_i \in \{1,...,k\}$, with prior distribution $q_1,...,q_k$

$k \times k$ matrix p

if $t_i = r$ and $t_j = s$, there is an edge $i \rightarrow j$ with probability p_{rs}

p is not necessarily symmetric, and we don't assume that $p_{rr} > p_{rs}$

given a graph G, we want to simultaneously

- label the nodes, i.e., infer the type assignment $t : V \rightarrow \{1,...,k\}$
- learn how nodes affect connections, i.e., infer the matrix p

The stochastic block model

take a discrete attribute into account:

each vertex i has a type $t_i \in \{1,...,k\}$, with prior distribution $q_1,...,q_k$

$k \times k$ matrix p

if $t_i = r$ and $t_j = s$, there is an edge $i \to j$ with probability p_{rs}

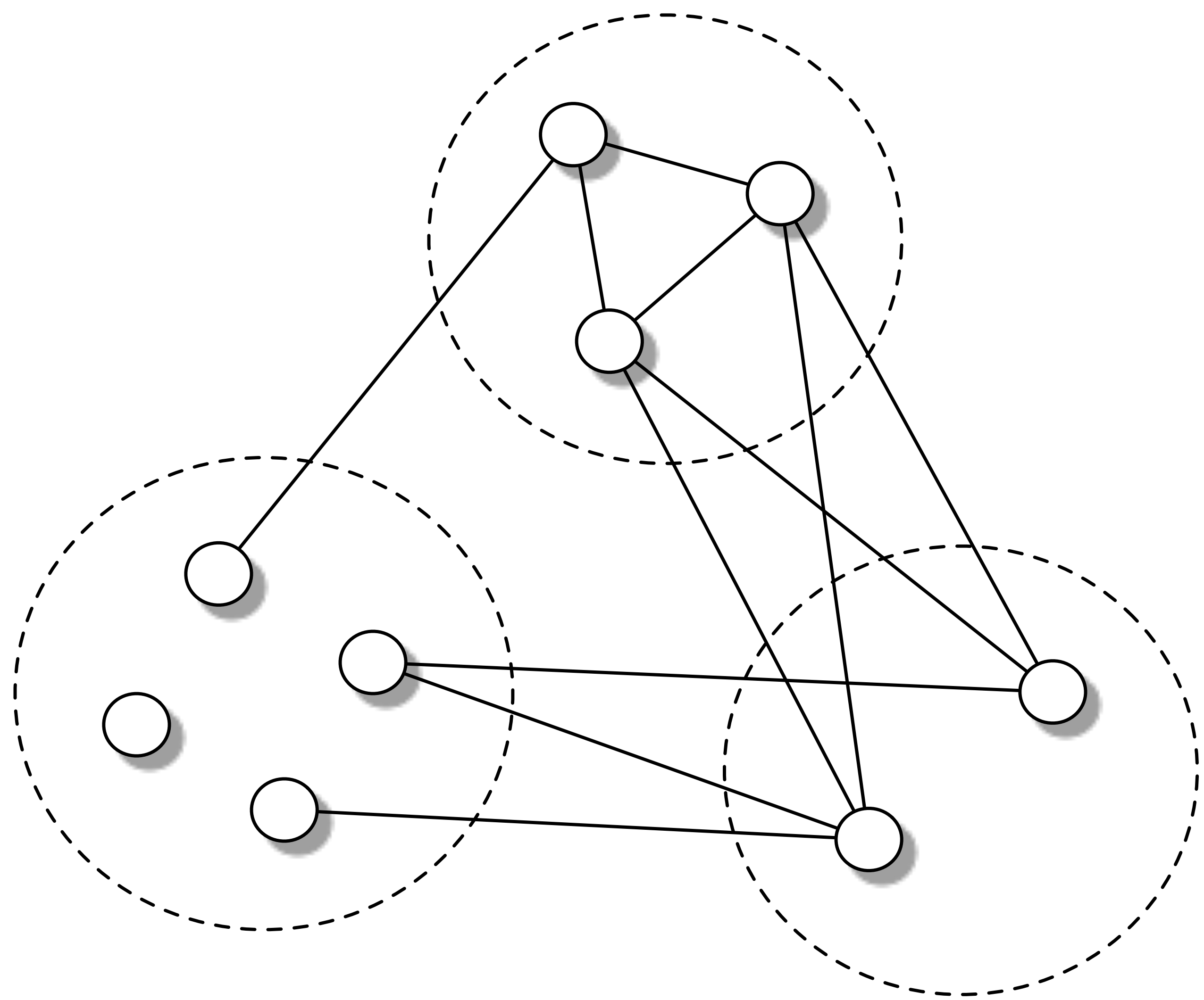
p is not necessarily symmetric, and we don't assume that $p_{rr} > p_{rs}$

given a graph G, we want to simultaneously

- label the nodes, i.e., infer the type assignment $t : V \to \{1,...,k\}$
- learn how nodes affect connections, i.e., infer the matrix p

how do we get off the ground?

Assortative and disassortative



The likelihood

The likelihood

the probability of *G* given the types *t* and parameters θ=(*p*,*q*) is

$$P(G \mid t, \theta) = \prod_{(i,j)\in E} p_{t_i,t_j} \prod_{(i,j)\notin E} (1 - p_{t_i,t_j})$$

The likelihood

the probability of *G* given the types *t* and parameters θ=(*p*,*q*) is

$$P(G \mid t, \theta) = \prod_{(i,j)\in E} p_{t_i,t_j} \prod_{(i,j)\notin E} (1 - p_{t_i,t_j})$$

so the probability of *t* given *G* is

$$P(t \mid G, \theta) = \frac{P(t \mid \theta)\, P(G \mid t, \theta)}{\sum_{t' \in \{1,\ldots,k\}^n} P(G \mid t', \theta)}$$

$$\propto \prod_{i\in V} q_{t_i} \prod_{(i,j)\in E} p_{t_i,t_j} \prod_{(i,j)\notin E} (1 - p_{t_i,t_j})$$

The likelihood

the probability of *G* given the types *t* and parameters θ=(*p*,*q*) is

$$P(G \mid t, \theta) = \prod_{(i,j)\in E} p_{t_i,t_j} \prod_{(i,j)\notin E} (1 - p_{t_i,t_j})$$

so the probability of *t* given *G* is

$$P(t \mid G, \theta) = \frac{P(t \mid \theta)\, P(G \mid t, \theta)}{\sum_{t' \in \{1,\ldots,k\}^n} P(G \mid t', \theta)}$$
$$\propto \prod_{i \in V} q_{t_i} \prod_{(i,j)\in E} p_{t_i,t_j} \prod_{(i,j)\notin E} (1 - p_{t_i,t_j})$$

call this the Gibbs distribution on *t*. How do we maximize it, or sample from it?

Maximizing the likelihood

Maximizing the likelihood

single-site heat-bath dynamics: choose a random vertex and update its type

Maximizing the likelihood

single-site heat-bath dynamics: choose a random vertex and update its type

if we like, we can jointly maximize $P(G|t,\theta)$ as a function of t and p by setting

$$p_{rs} = \frac{e_{rs}}{n_r n_s}\,, \quad q_r = \frac{n_r}{n}$$

Maximizing the likelihood

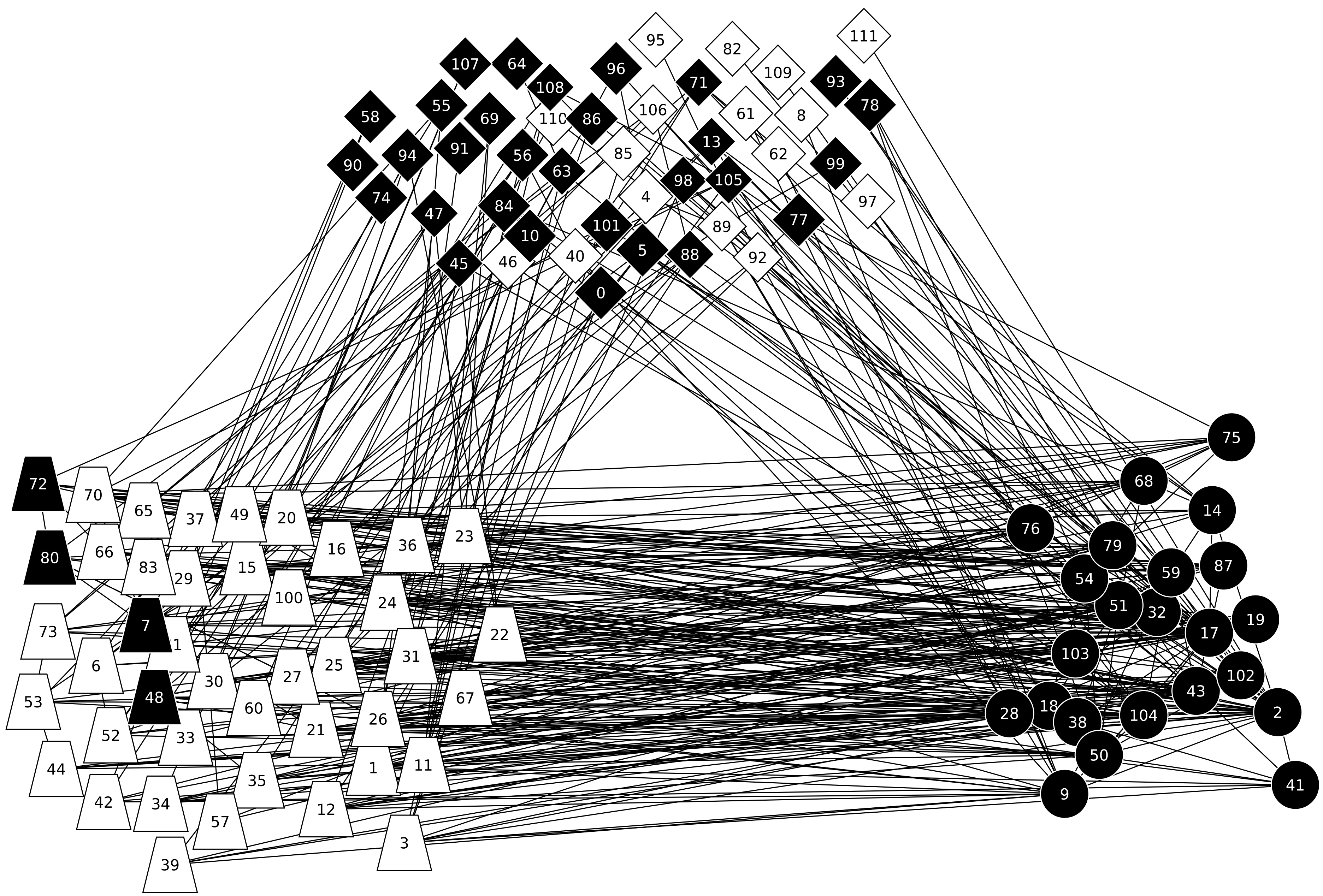
single-site heat-bath dynamics: choose a random vertex and update its type

if we like, we can jointly maximize $P(G|t,\theta)$ as a function of t and p by setting

$$p_{rs} = \frac{e_{rs}}{n_r n_s}, \quad q_r = \frac{n_r}{n}$$

this works reasonably well on small networks...

I record that I was born on a Friday



Maximizing the likelihood

Maximizing the likelihood

single-site heat-bath dynamics: choose a random vertex and update its type

if we like, we can jointly maximize $P(G|t,p)$ as a function of t and p by setting

$$p_{rs} = \frac{e_{rs}}{n_r n_s}, \quad q_r = \frac{n_r}{n}$$

this works reasonably well on small networks... but it isn't really what we want

Maximizing the likelihood

single-site heat-bath dynamics: choose a random vertex and update its type

if we like, we can jointly maximize $P(G|t,p)$ as a function of t and p by setting

$$p_{rs} = \frac{e_{rs}}{n_r n_s}, \quad q_r = \frac{n_r}{n}$$

this works reasonably well on small networks... but it isn't really what we want

the probability of θ given G is a proportional to a partition function

$$P(G\,|\,\theta) = \sum_{t \in \{1,\ldots,k\}^n} P(G\,|\,t,\theta)$$

Maximizing the likelihood

single-site heat-bath dynamics: choose a random vertex and update its type

if we like, we can jointly maximize $P(G|t,p)$ as a function of t and p by setting

$$p_{rs} = \frac{e_{rs}}{n_r n_s}\,, \quad q_r = \frac{n_r}{n}$$

this works reasonably well on small networks... but it isn't really what we want

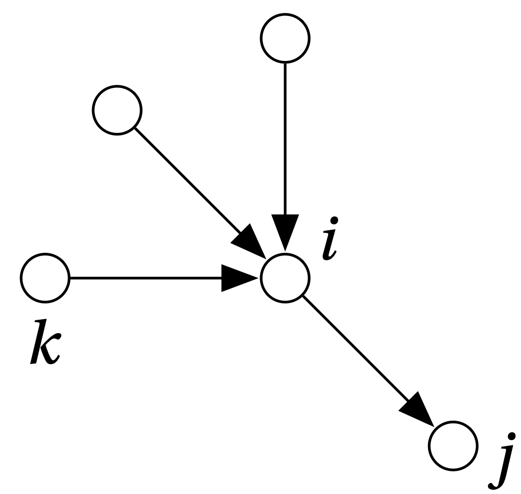
the probability of θ given G is a proportional to a partition function

$$P(G\,|\,\theta) = \sum_{t \in \{1,\ldots,k\}^n} P(G\,|\,t,\theta)$$

and $-\log P(G|\theta)$ is a free energy, not a ground state energy

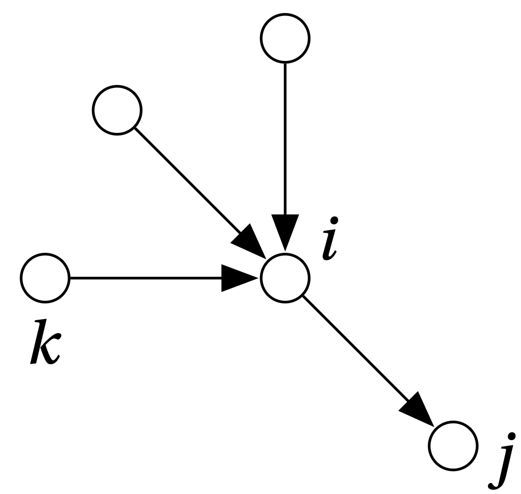
Belief propagation (a.k.a. the cavity method)

Belief propagation (a.k.a. the cavity method)



each vertex *i* sends a “message” to each of its neighbors j, giving *i*’s marginal distribution based on its other neighbors *k*

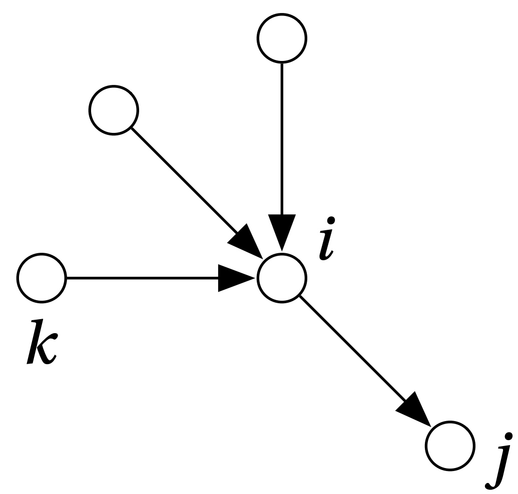
Belief propagation (a.k.a. the cavity method)



each vertex *i* sends a “message” to each of its neighbors j, giving *i*’s marginal distribution based on its other neighbors *k*

denote this message $\mu_r^{i\to j} = \text{estimate of } \Pr[t_i = r] \text{ if } j \text{ were absent}$

Belief propagation (a.k.a. the cavity method)

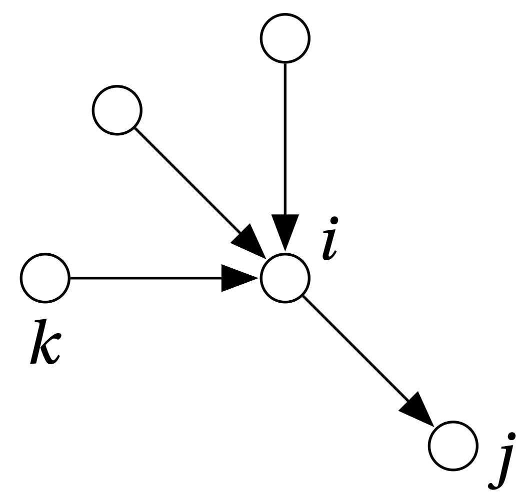


each vertex *i* sends a “message” to each of its neighbors j, giving *i*’s marginal distribution based on its other neighbors *k*

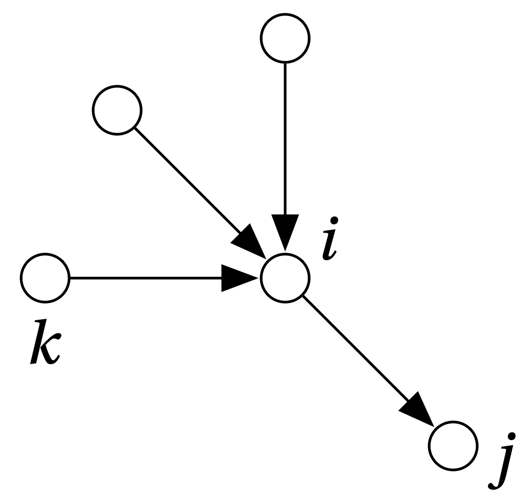
denote this message $\mu_r^{i \to j} = \text{estimate of } \Pr[t_i = r] \text{ if } j \text{ were absent}$

how do we update it?

Belief propagation (a.k.a. the cavity method)

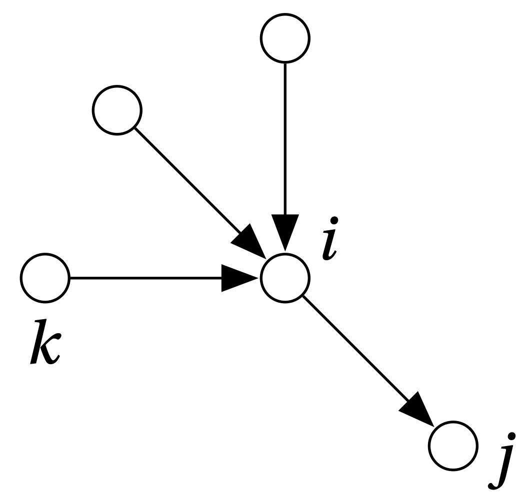


Belief propagation (a.k.a. the cavity method)



$$\mu_s^{i\to j} = \frac{1}{Z^{i\to j}}\, q_s \prod_{\substack{k\neq j\\(i,k)\in E}} \sum_r \mu_r^{k\to i} p_{rs} \times \prod_{\substack{k\neq j\\(i,k)\notin E}} \sum_r \mu_r^{k\to i}(1-p_{rs})$$

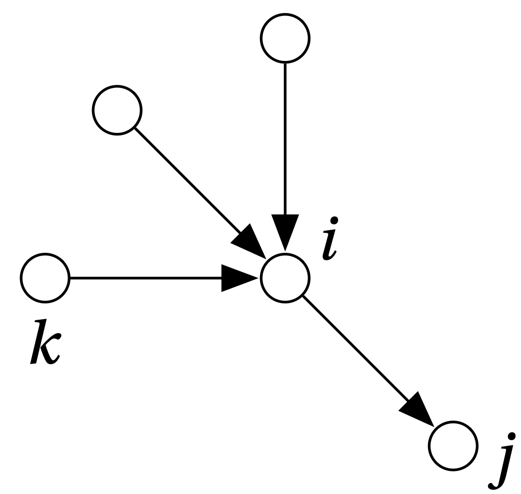
Belief propagation (a.k.a. the cavity method)



conditional independence

$$\mu_s^{i\to j} = \frac{1}{Z^{i\to j}}\, q_s \prod_{\substack{k\neq j \\ (i,k)\in E}} \sum_r \mu_r^{k\to i}\, p_{rs} \times \prod_{\substack{k\neq j \\ (i,k)\notin E}} \sum_r \mu_r^{k\to i} (1-p_{rs})$$

Belief propagation (a.k.a. the cavity method)

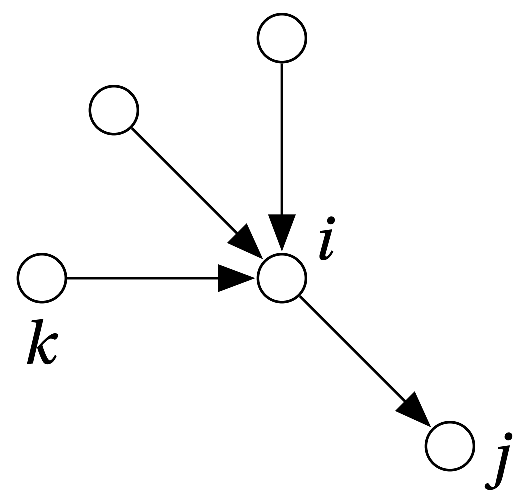


conditional independence

$$\mu_s^{i\to j} = \frac{1}{Z^{i\to j}}\, q_s \prod_{\substack{k\neq j\\(i,k)\in E}} \sum_r \mu_r^{k\to i} p_{rs} \times \prod_{\substack{k\neq j\\(i,k)\notin E}} \sum_r \mu_r^{k\to i} (1-p_{rs})$$

BP on a complete graph — takes O(n^2) time to update

Belief propagation (a.k.a. the cavity method)



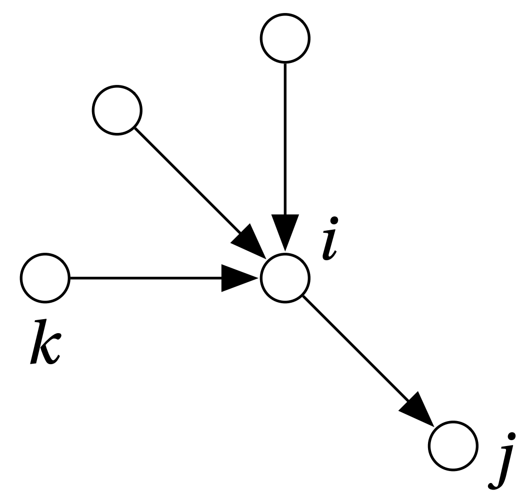
conditional independence

$$\mu_s^{i\to j} = \frac{1}{Z^{i\to j}}\, q_s \prod_{\substack{k\neq j \\ (i,k)\in E}} \sum_r \mu_r^{k\to i} p_{rs} \times \prod_{\substack{k\neq j \\ (i,k)\notin E}} \sum_r \mu_r^{k\to i}(1-p_{rs})$$

BP on a complete graph — takes O(n^2) time to update

can simplify by assuming that $\mu_r^{k\to i} = \mu_r^k$ for all non-neighbors *i*

Belief propagation (a.k.a. the cavity method)



conditional independence

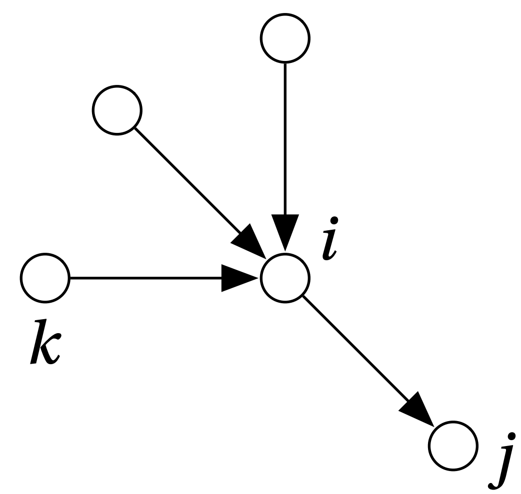
$$\mu_s^{i\to j} = \frac{1}{Z^{i\to j}}\, q_s \prod_{\substack{k\neq j \\ (i,k)\in E}} \sum_r \mu_r^{k\to i} p_{rs} \times \prod_{\substack{k\neq j \\ (i,k)\notin E}} \sum_r \mu_r^{k\to i}(1-p_{rs})$$

BP on a complete graph — takes O(n^2) time to update

can simplify by assuming that $\mu_r^{k\to i} = \mu_r^k$ for all non-neighbors *i*

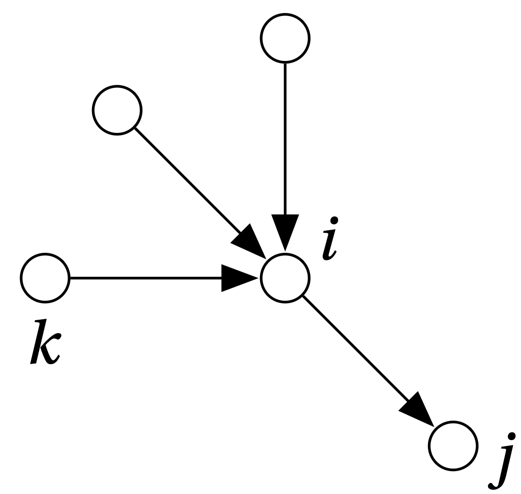
each vertex *k* applies an “external field” $\sum_r \mu_r^k (1-p_{rs})$ to all vertices of type *s*

Belief propagation (a.k.a. the cavity method)



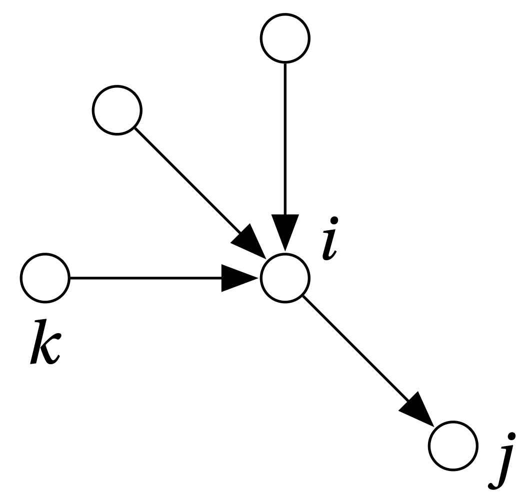
$$\mu_s^{i\to j} = \frac{1}{Z^{i\to j}}\, q_s \prod_{\substack{k\neq j \\ (i,k)\in E}} \sum_r \mu_r^{k\to i} p_{rs} \times \prod_{\substack{k\neq j \\ (i,k)\notin E}} \sum_r \mu_r^{k\to i}(1-p_{rs})$$

Belief propagation (a.k.a. the cavity method)



$$\mu_s^{i\to j} = \frac{1}{Z^{i\to j}}\, q_s \prod_{\substack{k\neq j \\ (i,k)\in E}} \sum_r \mu_r^{k\to i} p_{rs} \times \frac{\prod_k \sum_r \mu_r^k (1-p_{rs})}{\prod_{k:(i,k)\in E} \sum_r \mu_r^k (1-p_{rs})}$$

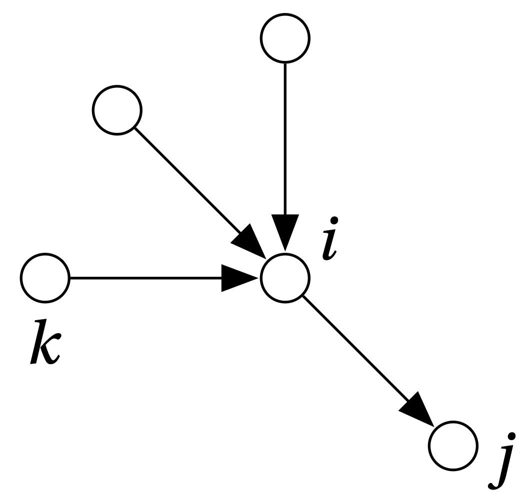
Belief propagation (a.k.a. the cavity method)



$$\mu_s^{i\to j} = \frac{1}{Z^{i\to j}} q_s \prod_{\substack{k\neq j \\ (i,k)\in E}} \sum_r \mu_r^{k\to i} p_{rs} \times \frac{\prod_k \sum_r \mu_r^k (1-p_{rs})}{\prod_{k:(i,k)\in E} \sum_r \mu_r^k (1-p_{rs})}$$

each update now takes O(*n*+*m*) time

Belief propagation (a.k.a. the cavity method)

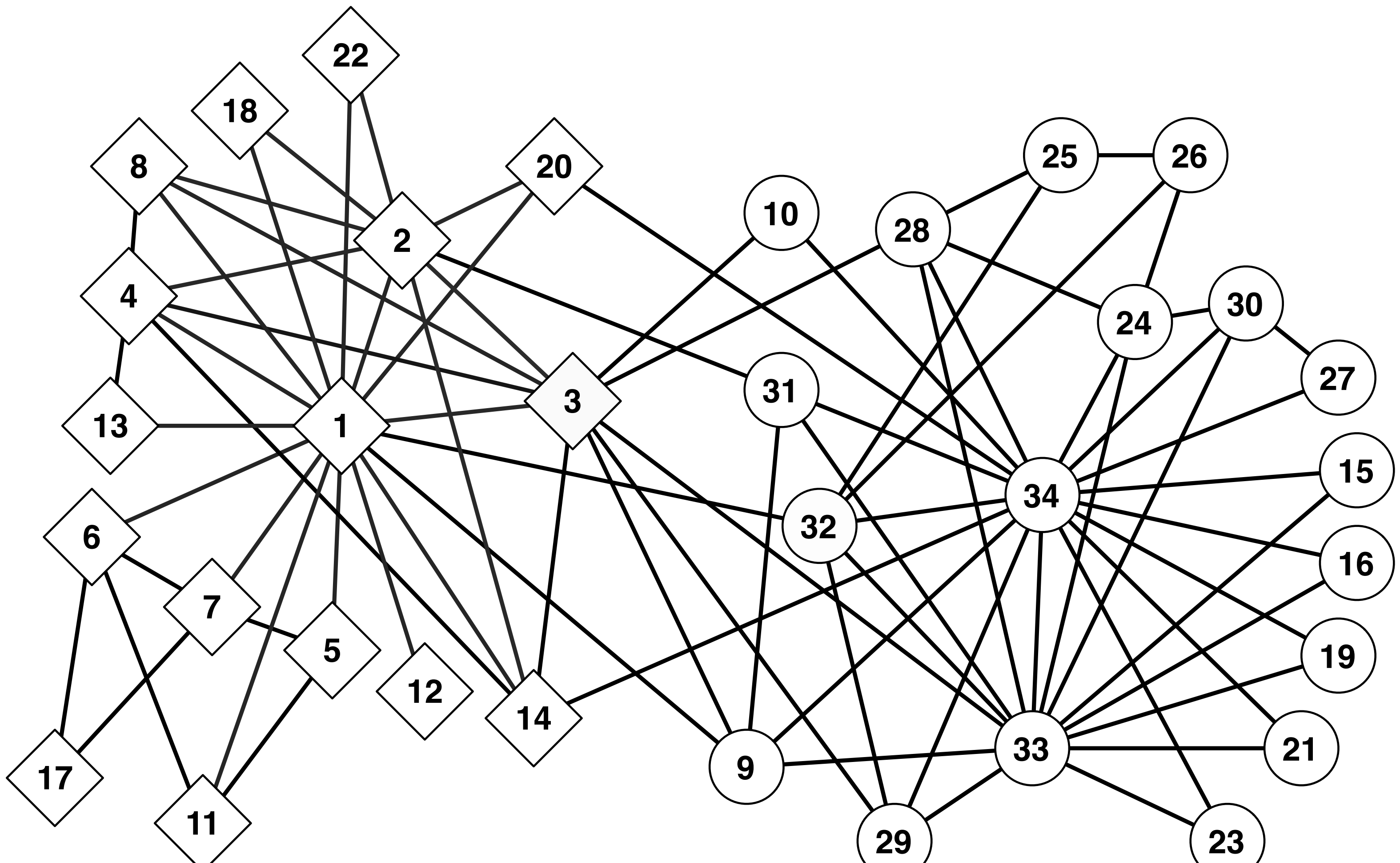


$$\mu_s^{i\to j} = \frac{1}{Z^{i\to j}} q_s \prod_{\substack{k\neq j \\ (i,k)\in E}} \sum_r \mu_r^{k\to i} p_{rs} \times \frac{\prod_k \sum_r \mu_r^k (1-p_{rs})}{\prod_{k:(i,k)\in E} \sum_r \mu_r^k (1-p_{rs})}$$

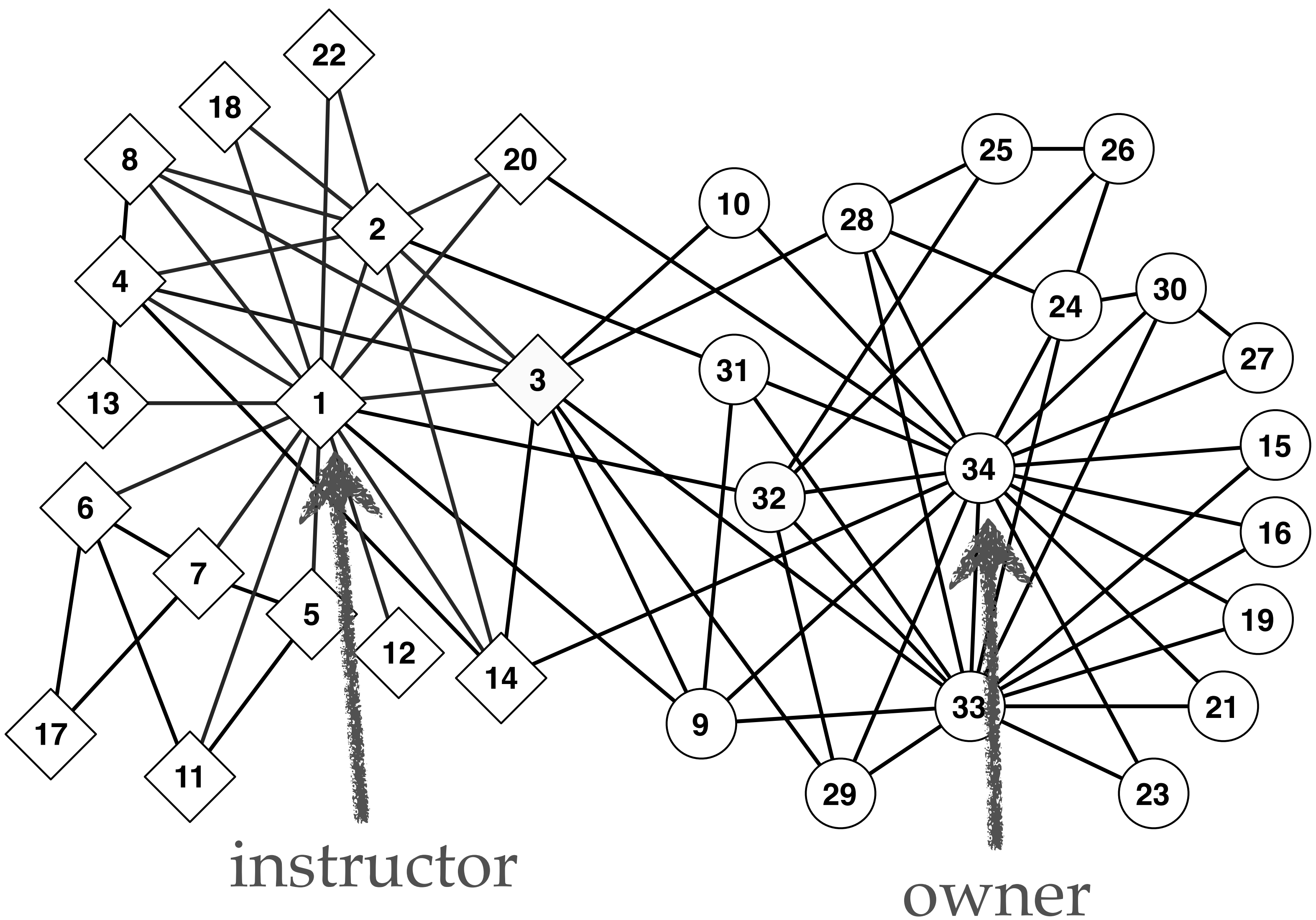
each update now takes O(*n*+*m*) time

update until the messages reach a fixed point

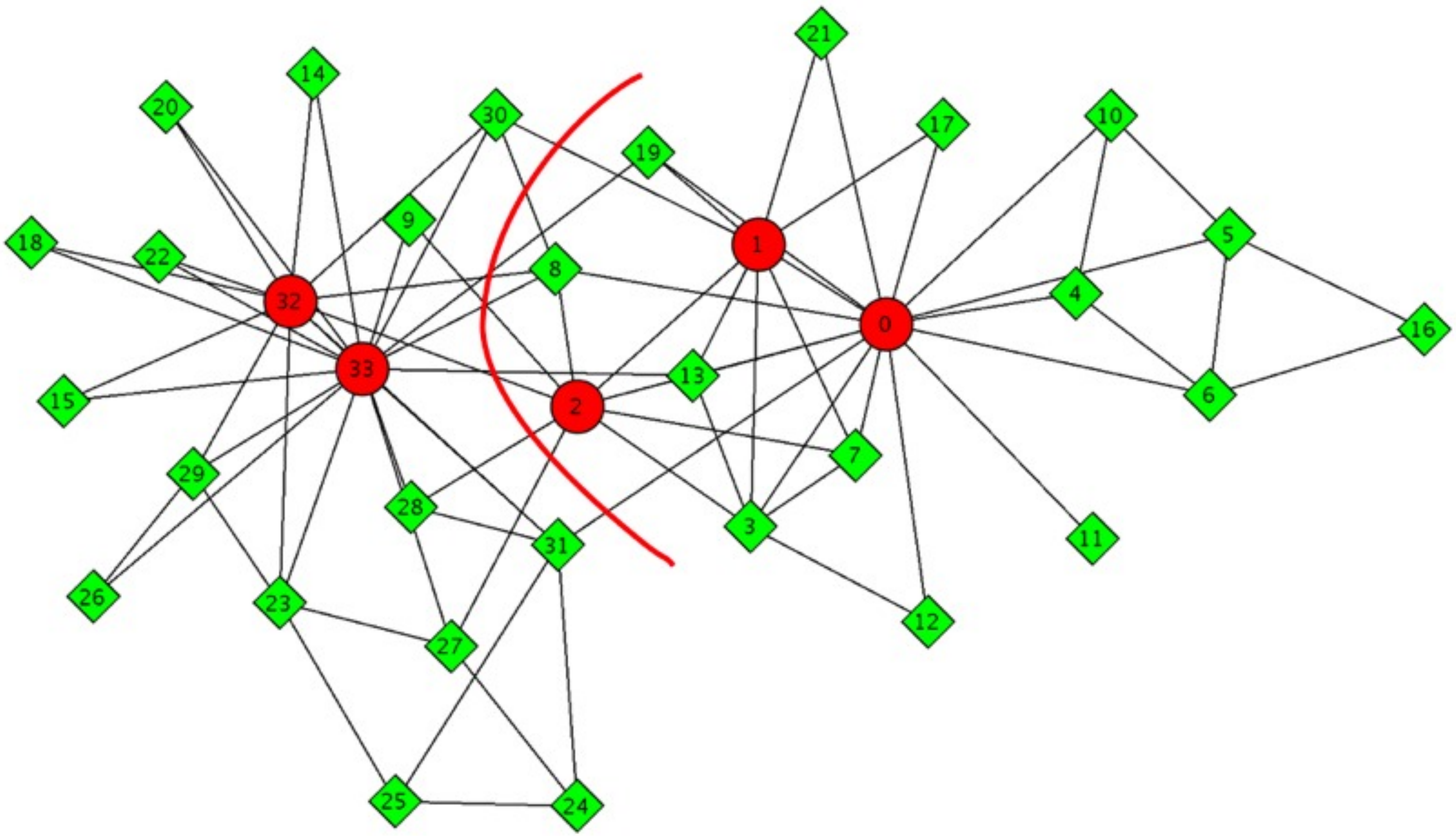
The karate club again



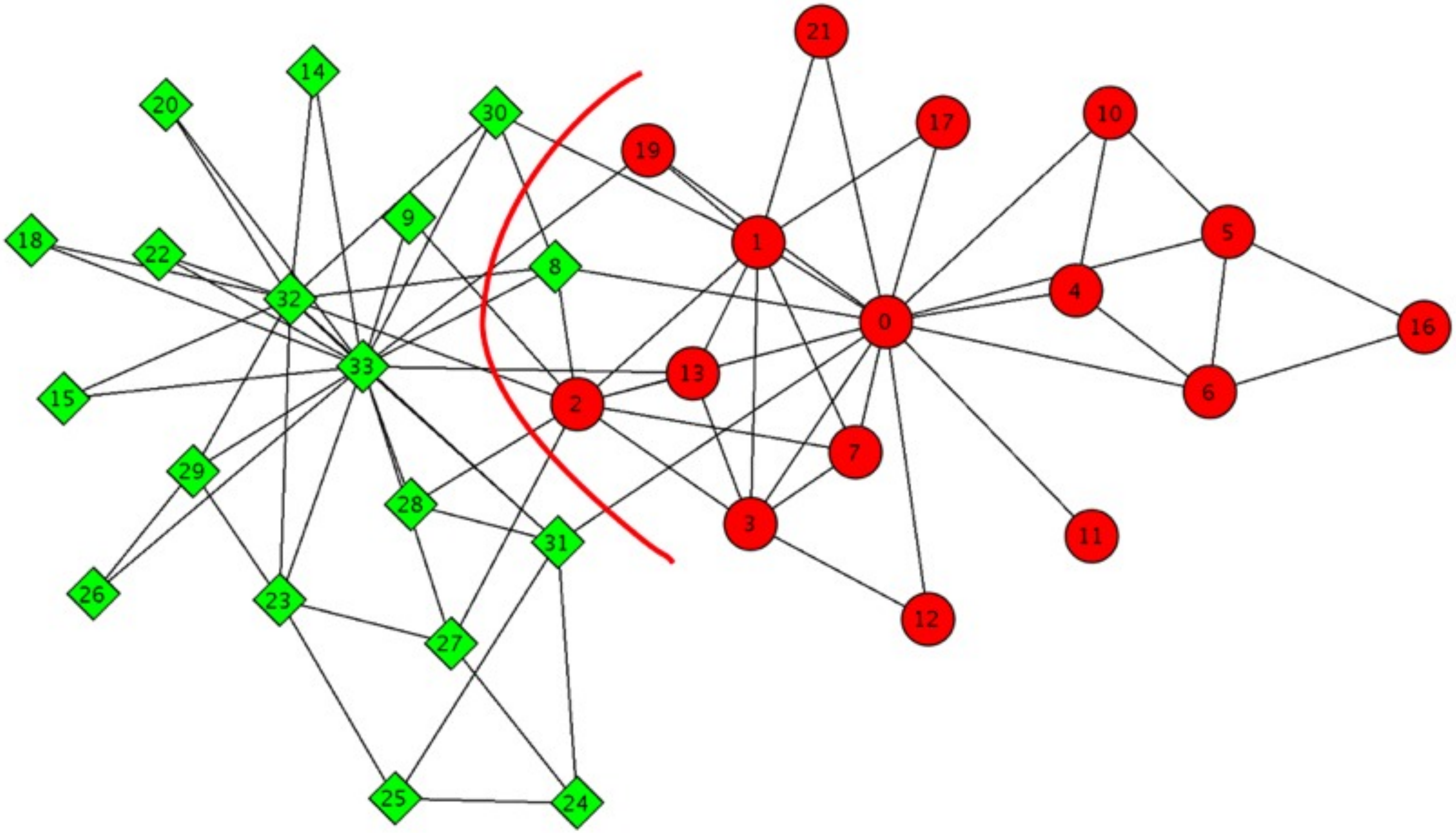
The karate club again



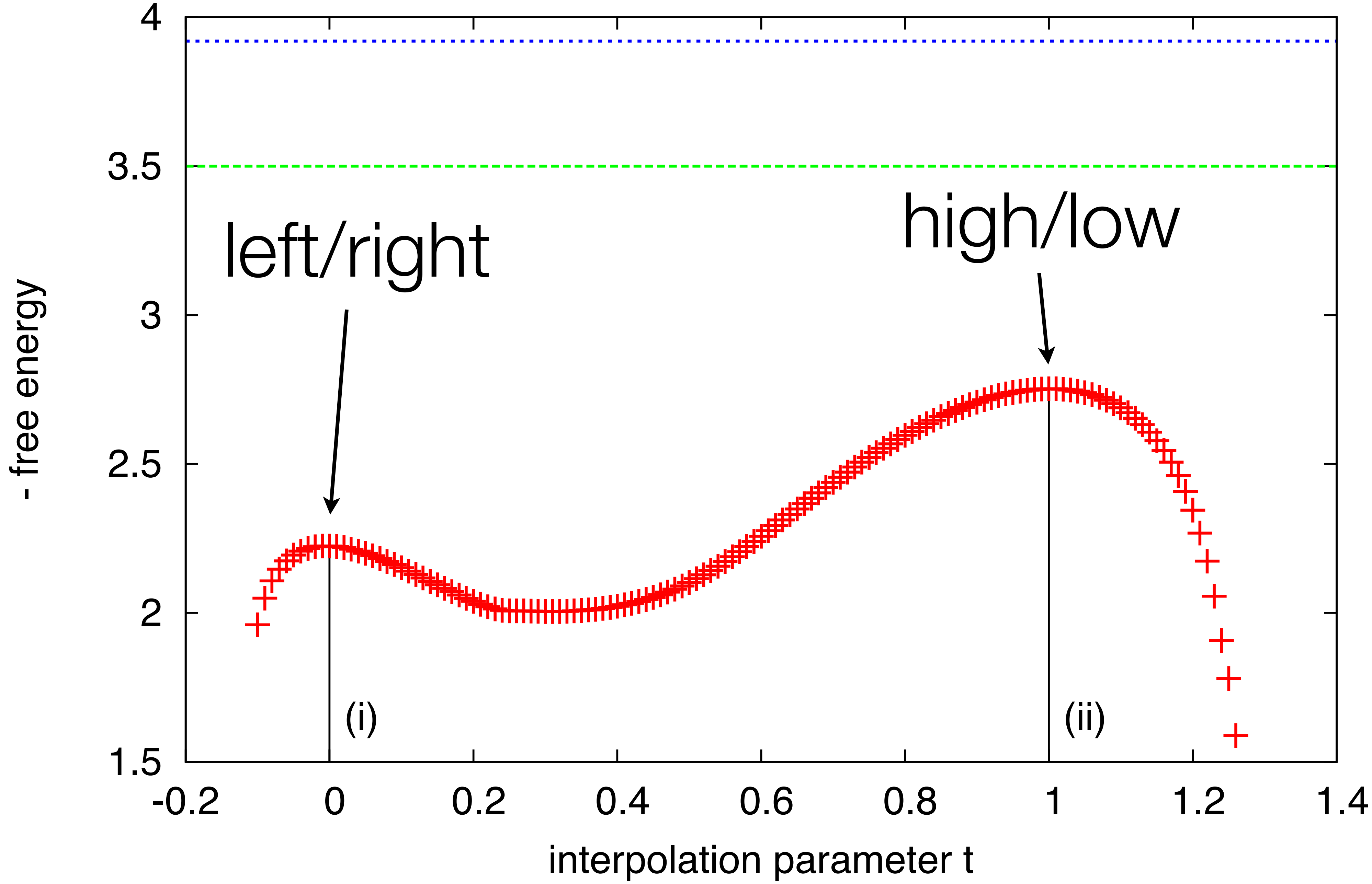
Which kind of community do you want?



Which kind of community do you want?



Two local optima



Degree-corrected block models

Degree-corrected block models

the “vanilla” block model expects vertices of the same type to have roughly the same degree

Degree-corrected block models

the “vanilla” block model expects vertices of the same type to have roughly the same degree

a random multigraph [Karrer & Newman, 2010]

Degree-corrected block models

the “vanilla” block model expects vertices of the same type to have roughly the same degree

a random multigraph [Karrer & Newman, 2010]

each vertex i has an expected degree d_i

Degree-corrected block models

the "vanilla" block model expects vertices of the same type to have roughly the same degree

a random multigraph [Karrer & Newman, 2010]

each vertex i has an expected degree d_i

analogous to p_{ij}, a $k \times k$ matrix w_{ij}

Degree-corrected block models

the "vanilla" block model expects vertices of the same type to have roughly the same degree

a random multigraph [Karrer & Newman, 2010]

each vertex i has an expected degree d_i

analogous to p_{ij}, a $k \times k$ matrix w_{ij}

for each pair i, j with $t_i = r$ and $t_j = s$, the number of edges between them is

$$m_{ij} \sim \mathrm{Poi}(d_i d_j w_{rs})$$

Degree-corrected block models

the "vanilla" block model expects vertices of the same type to have roughly the same degree

a random multigraph [Karrer & Newman, 2010]

each vertex *i* has an expected degree d_i

analogous to p_{ij}, a $k \times k$ matrix w_{ij}

for each pair *i*, *j* with $t_i = r$ and $t_j = s$, the number of edges between them is

$$m_{ij} \sim \mathrm{Poi}(d_i d_j w_{rs})$$

now the degrees are parameters, not data to be explained

Degree-corrected block models

the “vanilla” block model expects vertices of the same type to have roughly the same degree

a random multigraph [Karrer & Newman, 2010]

each vertex i has an expected degree d_i

analogous to p_{ij}, a $k \times k$ matrix w_{ij}

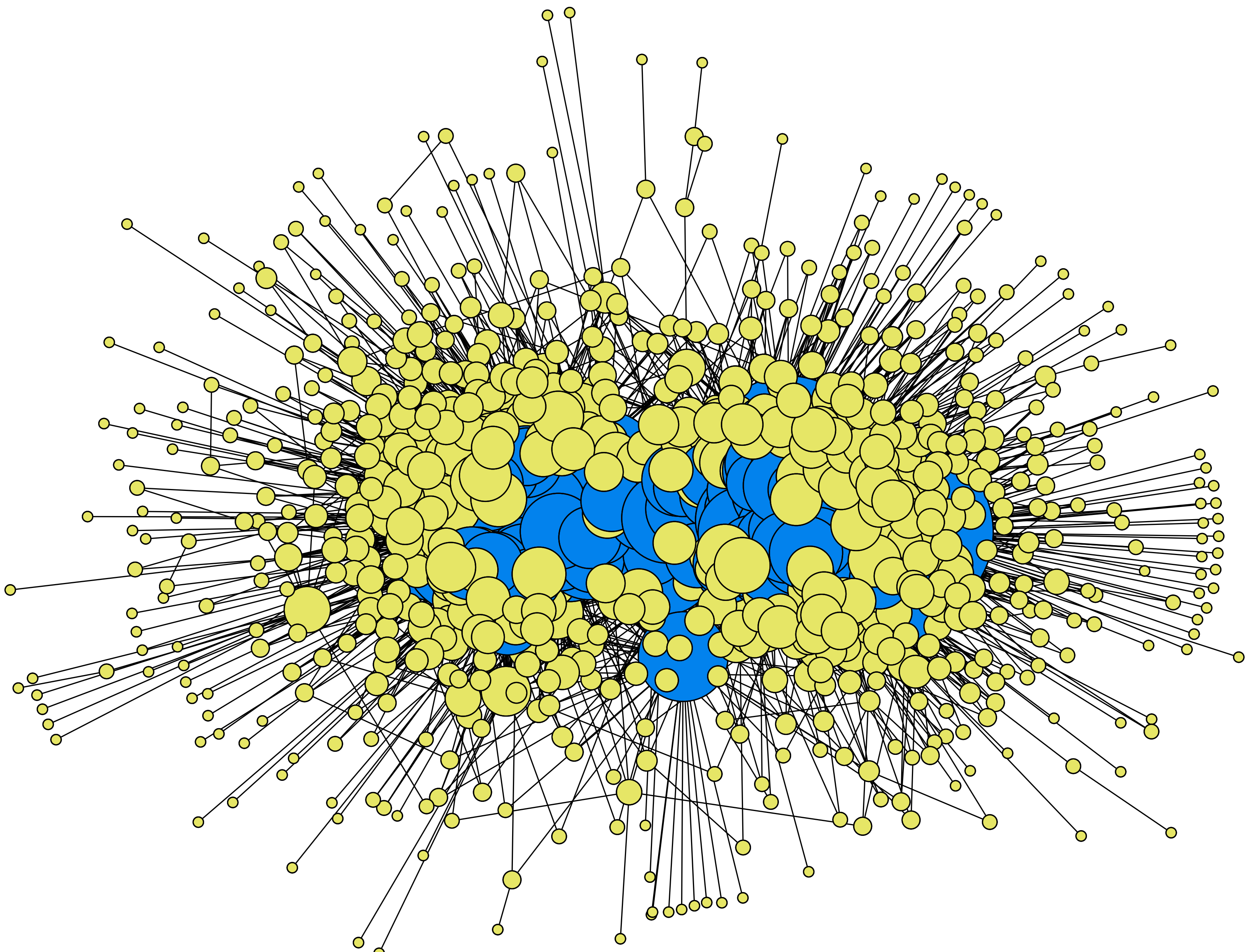
for each pair i, j with $t_i = r$ and $t_j = s$, the number of edges between them is

$$m_{ij} \sim \text{Poi}(d_i d_j w_{rs})$$

now the degrees are parameters, not data to be explained

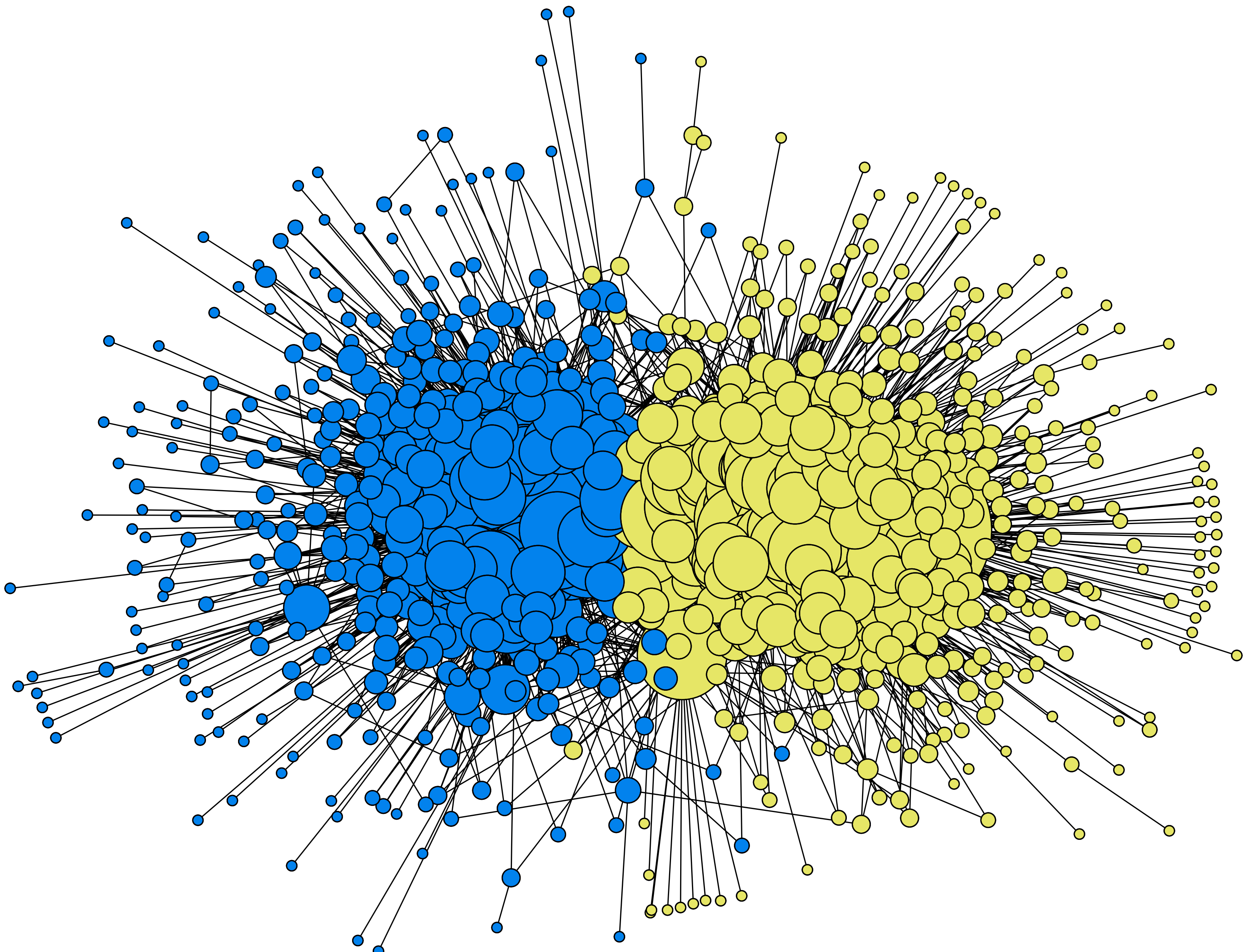
can again write down the BP/EM algorithm

Blogs: vanilla block model



[Karrer & Newman, 2010]

Blogs: degree-corrected block model



[Karrer & Newman, 2010]

Strengths and weaknesses

Strengths and weaknesses

degree-corrected models don't mind inhomogeneous degree distributions...

Strengths and weaknesses

degree-corrected models don’t mind inhomogeneous degree distributions...

but they also can’t use the degrees to help them label the nodes

Strengths and weaknesses

degree-corrected models don’t mind inhomogeneous degree distributions...

but they also can’t use the degrees to help them label the nodes

on some networks, they perform worse than the vanilla model

Strengths and weaknesses

degree-corrected models don't mind inhomogeneous degree distributions...

but they also can't use the degrees to help them label the nodes

on some networks, they perform worse than the vanilla model

yet another model: first generate vertex degrees d_i according to some distribution whose parameters depend on t_i (e.g. power law)

Strengths and weaknesses

degree-corrected models don't mind inhomogeneous degree distributions...

but they also can't use the degrees to help them label the nodes

on some networks, they perform worse than the vanilla model

yet another model: first generate vertex degrees d_i according to some distribution whose parameters depend on t_i (e.g. power law)

then generate edges according to the degree-corrected model

Strengths and weaknesses

degree-corrected models don't mind inhomogeneous degree distributions...

but they also can't use the degrees to help them label the nodes

on some networks, they perform worse than the vanilla model

yet another model: first generate vertex degrees d_i according to some distribution whose parameters depend on t_i (e.g. power law)

then generate edges according to the degree-corrected model

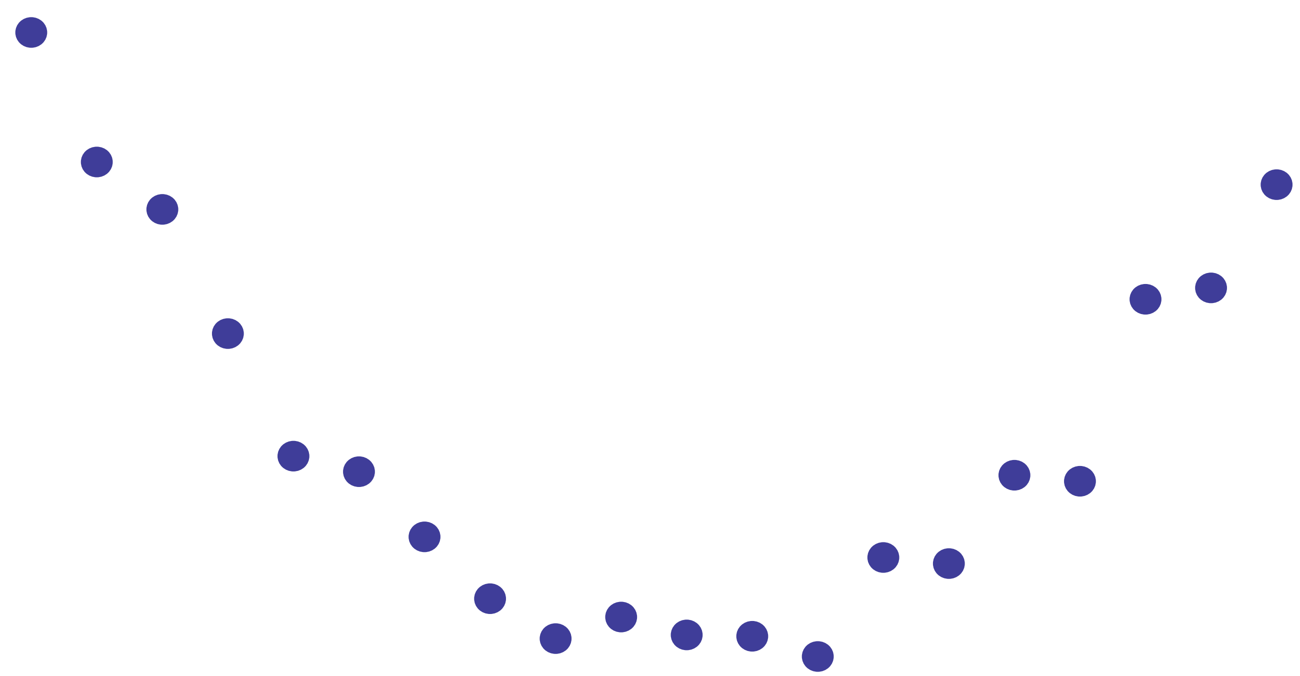
for some networks (e.g. word adjacency networks) works better than either vanilla or degree-corrected model

How can we tell if we're on the right track?

it's easy to fit data with a fancy model... the danger is overfitting

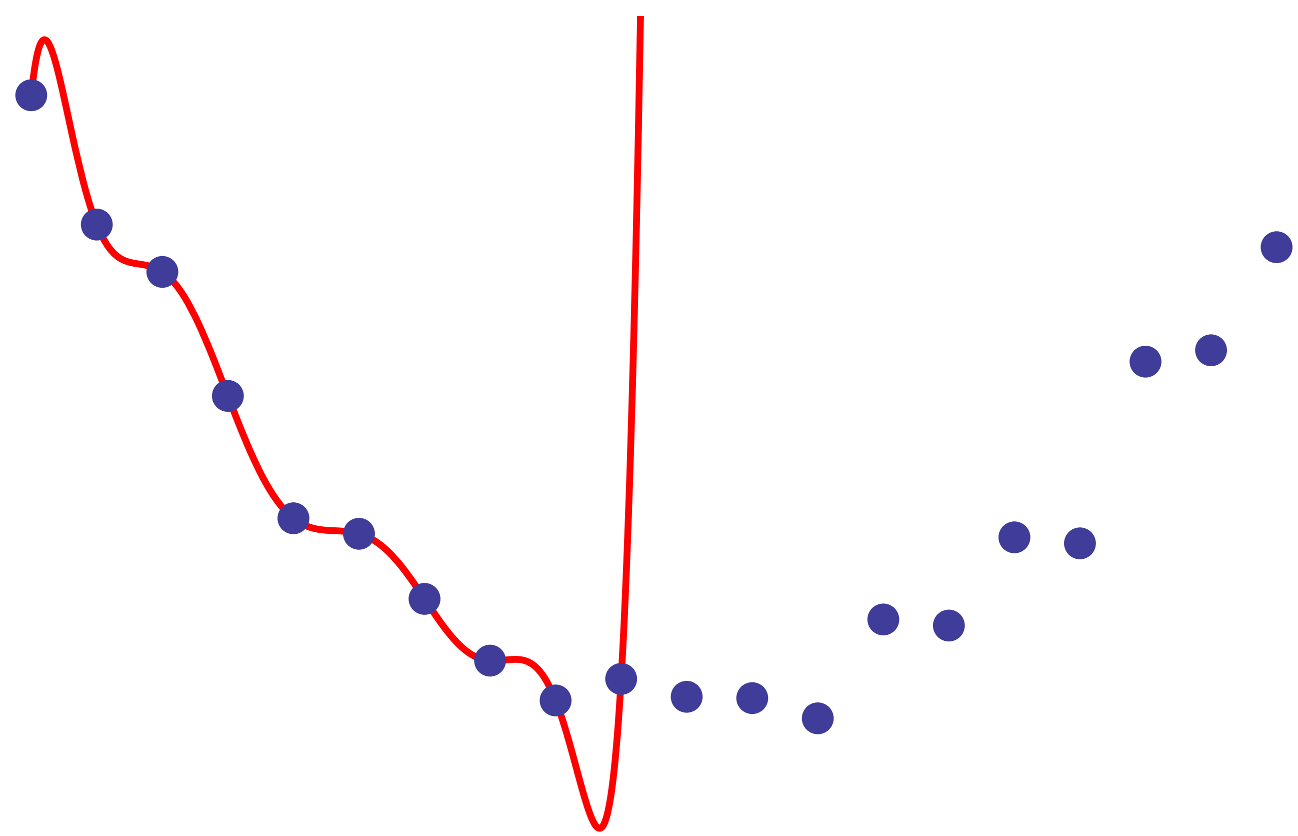
How can we tell if we're on the right track?

it's easy to fit data with a fancy model... the danger is overfitting



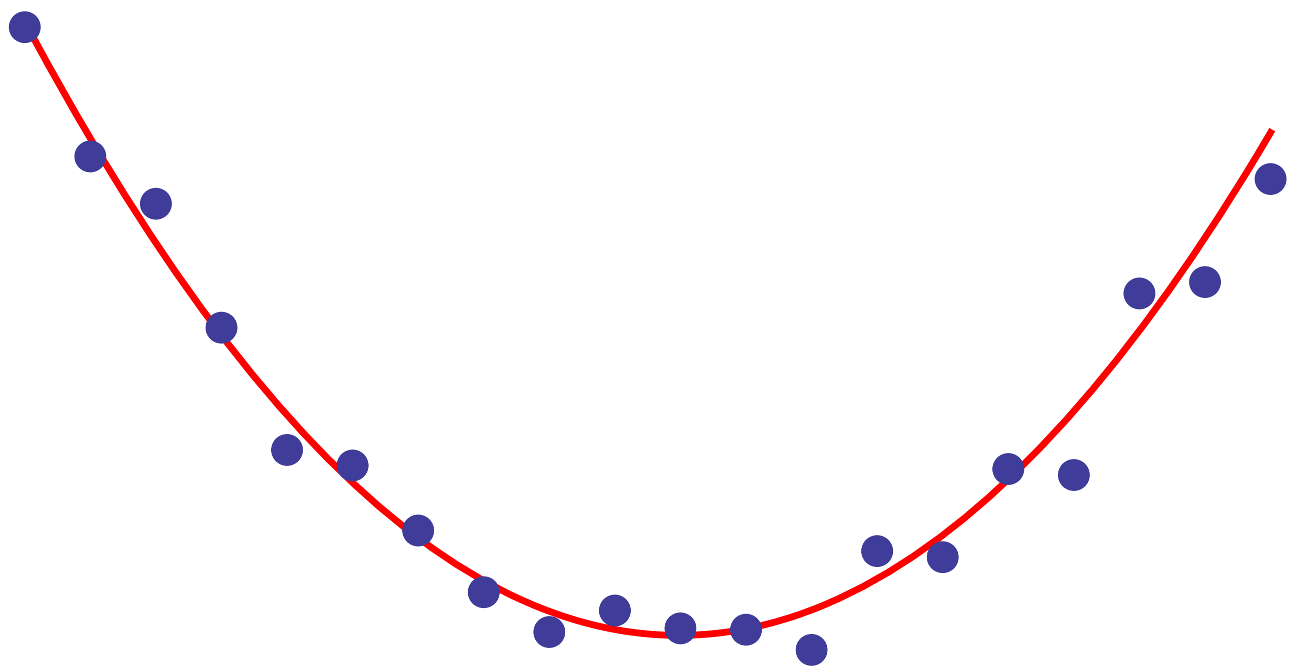
How can we tell if we're on the right track?

it's easy to fit data with a fancy model... the danger is overfitting



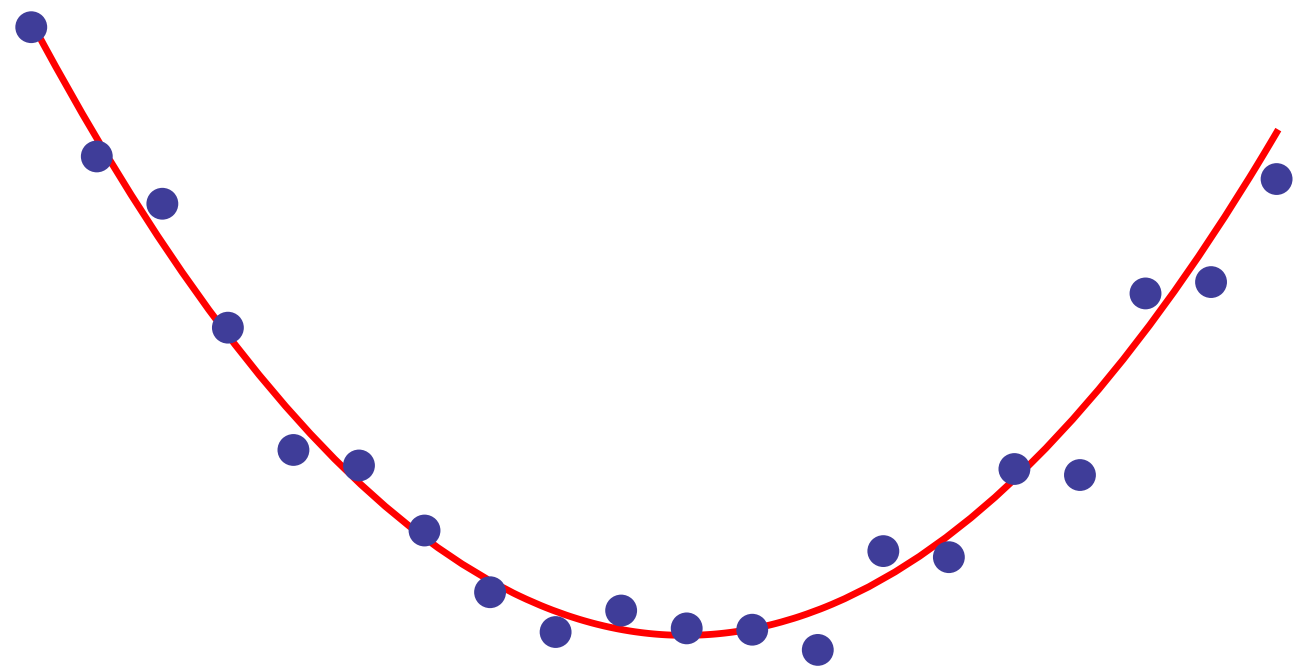
How can we tell if we're on the right track?

it's easy to fit data with a fancy model... the danger is overfitting



How can we tell if we're on the right track?

it's easy to fit data with a fancy model... the danger is overfitting



can we generalize from part of the data to the rest of it?

Active learning

Active learning

suppose we can learn a node's attributes, but at a cost

Active learning

suppose we can learn a node's attributes, but at a cost

we want to make good guesses about most of the nodes, after querying just a few of them—which ones?

Active learning

suppose we can learn a node's attributes, but at a cost

we want to make good guesses about most of the nodes, after querying just a few of them—which ones?

query the node with the largest *mutual information* between it and the others:

$$\begin{aligned} I(v, G-v) &= H(v) - H(v \mid G-v) \\ &= H(G-v) - H(G-v \mid v) \end{aligned}$$

Active learning

suppose we can learn a node's attributes, but at a cost

we want to make good guesses about most of the nodes, after querying just a few of them—which ones?

query the node with the largest *mutual information* between it and the others:

$$
\begin{aligned}
I(v, G - v) &= H(v) - H(v \mid G - v) \\
&= H(G - v) - H(G - v \mid v)
\end{aligned}
$$

average amount of information we learn about *G-v* we learn by querying *v*

Active learning

suppose we can learn a node's attributes, but at a cost

we want to make good guesses about most of the nodes, after querying just a few of them—which ones?

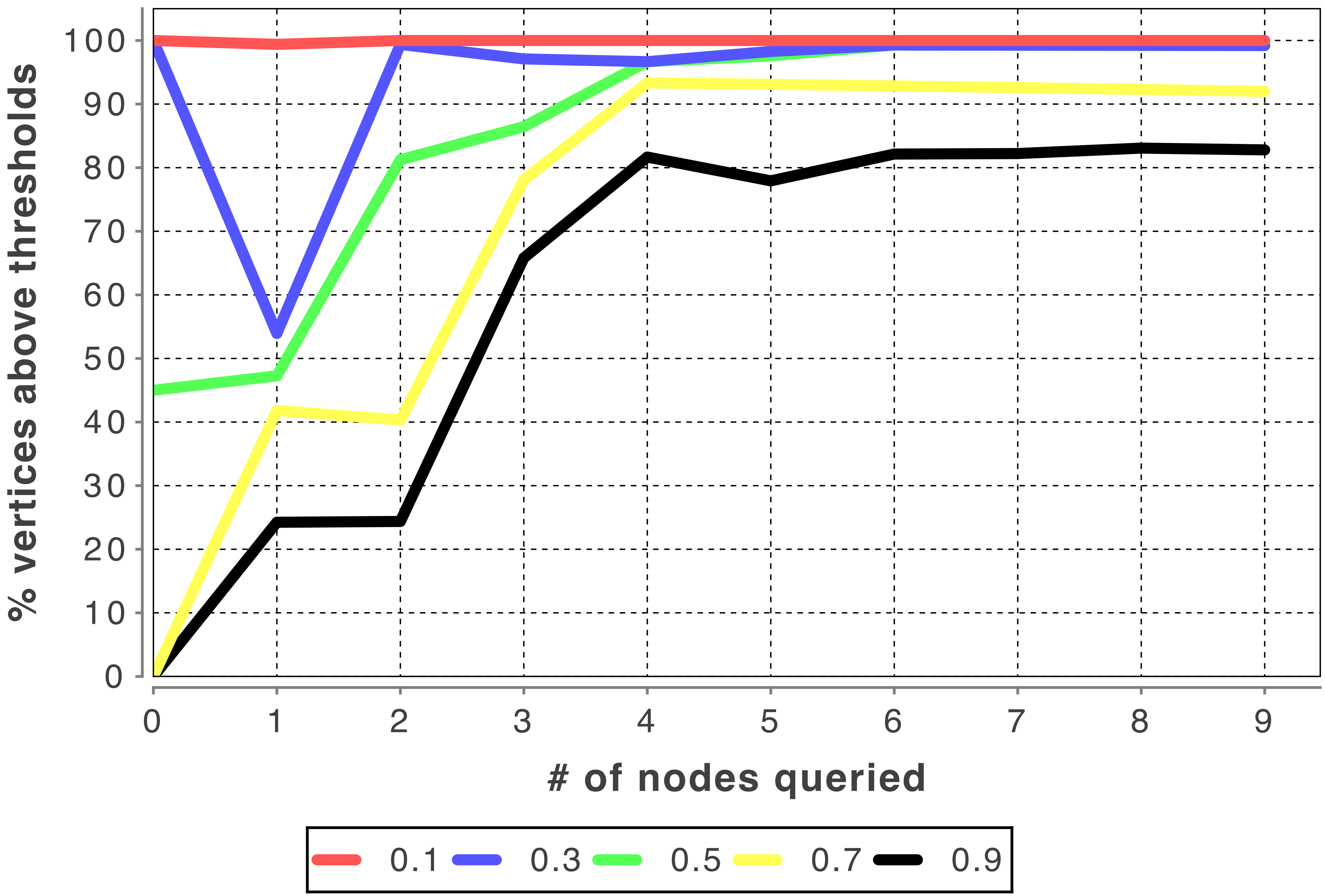
query the node with the largest *mutual information* between it and the others:

$$
\begin{aligned}
I(v, G-v) &= H(v) - H(v \mid G-v) \\
&= H(G-v) - H(G-v \mid v)
\end{aligned}
$$

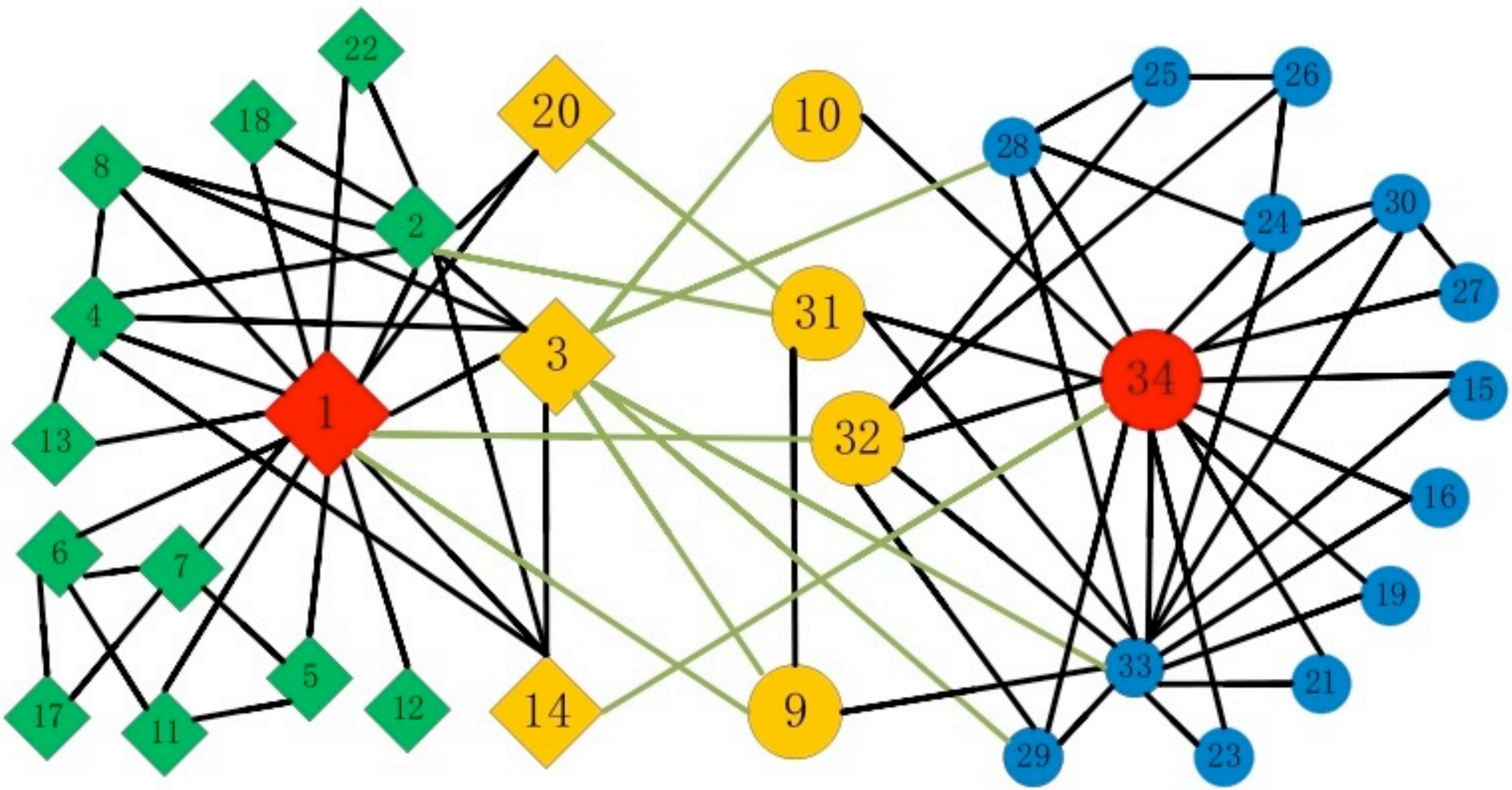
average amount of information we learn about *G-v* we learn by querying *v*

high when we're uncertain about *v*, and when *v* is highly correlated with others

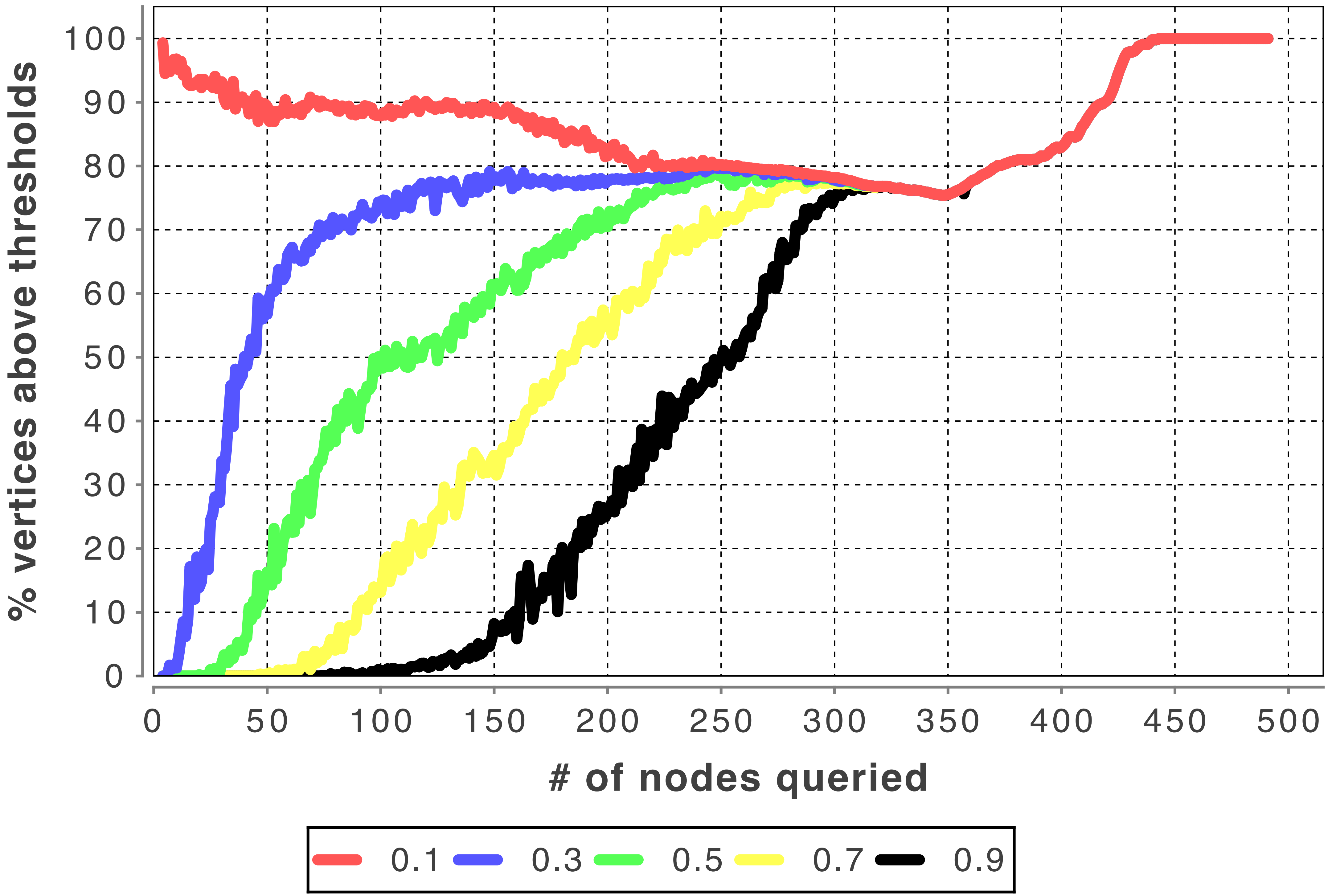
The Karate Club again



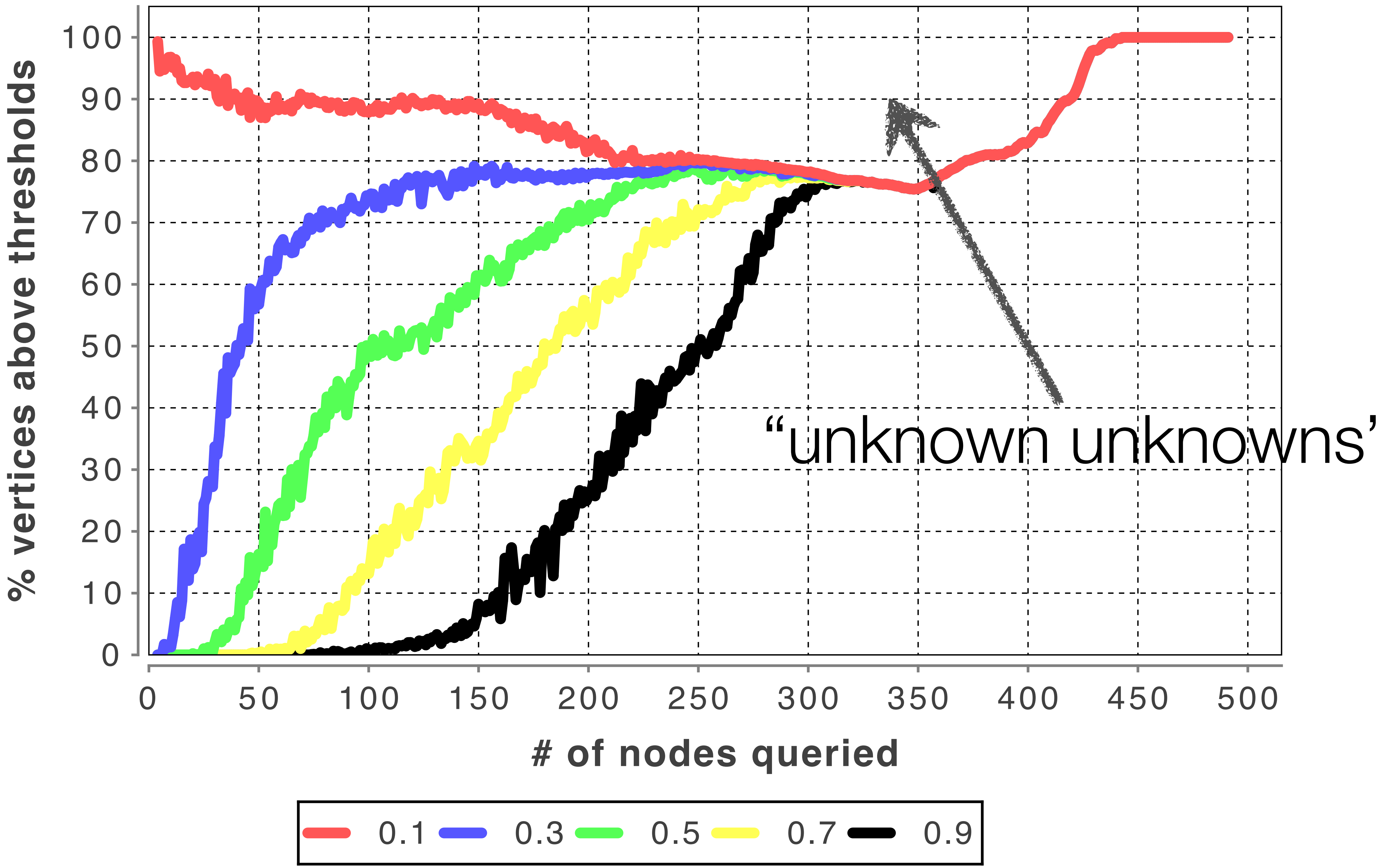
Which vertices do we query first?



An antarctic food web



An antarctic food web



Which rubber, which road?

Which rubber, which road?

these topological methods are all very nice, but...

Which rubber, which road?

these topological methods are all very nice, but...

what do they actually tell us about the *function* of nodes, and their dynamics?

Which rubber, which road?

these topological methods are all very nice, but...

what do they actually tell us about the *function* of nodes, and their dynamics?

do they give us good coarse-grainings of the dynamics?

Which rubber, which road?

these topological methods are all very nice, but...

what do they actually tell us about the *function* of nodes, and their dynamics?

do they give us good coarse-grainings of the dynamics?

can nodes in the same group “stand in” for each other under stress?

Which rubber, which road?

these topological methods are all very nice, but...

what do they actually tell us about the *function* of nodes, and their dynamics?

do they give us good coarse-grainings of the dynamics?

can nodes in the same group “stand in” for each other under stress?

are there good generative models for benchmark power grids?

Which rubber, which road?

these topological methods are all very nice, but...

what do they actually tell us about the *function* of nodes, and their dynamics?

do they give us good coarse-grainings of the dynamics?

can nodes in the same group “stand in” for each other under stress?

are there good generative models for benchmark power grids?

it’s easy to add attributes, e.g. geography, but we also need to generate loads, line capacities, generators...

Which rubber, which road?

these topological methods are all very nice, but...

what do they actually tell us about the *function* of nodes, and their dynamics?

do they give us good coarse-grainings of the dynamics?

can nodes in the same group "stand in" for each other under stress?

are there good generative models for benchmark power grids?

it's easy to add attributes, e.g. geography, but we also need to generate loads, line capacities, generators...

off-the-cuff idea: points distributed like the fraction distribution of population, e.g. according to a Levy flight, and then connected geometrically

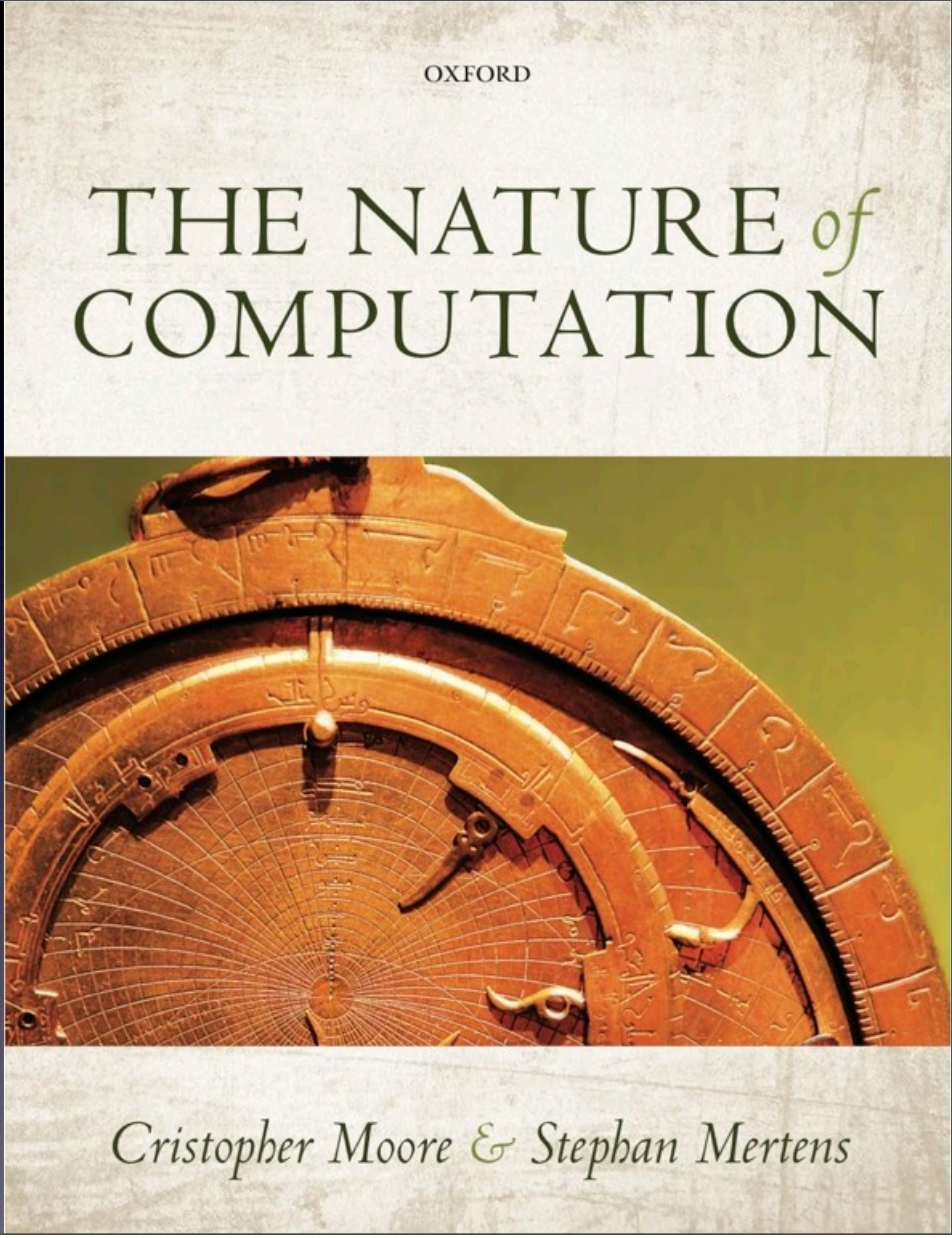
Shameless Plug

This book rocks! You somehow manage to combine the fun of a popular book with the intellectual heft of a textbook.
— Scott Aaronson

A treasure trove of information on algorithms and complexity, presented in the most delightful way.
— Vijay Vazirani

A creative, insightful, and accessible introduction to the theory of computing, written with a keen eye toward the frontiers of the field and a vivid enthusiasm for the subject matter.
— Jon Kleinberg

Oxford University Press, 2011



Acknowledgments

and the McDonnell Foundation