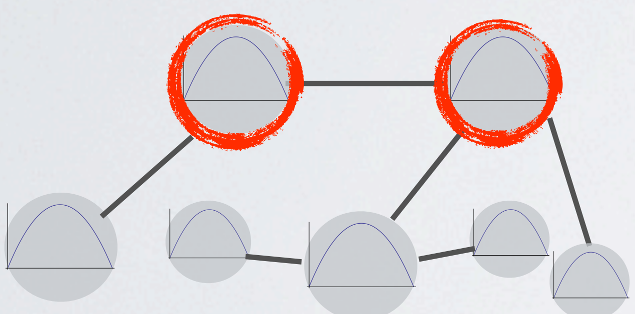
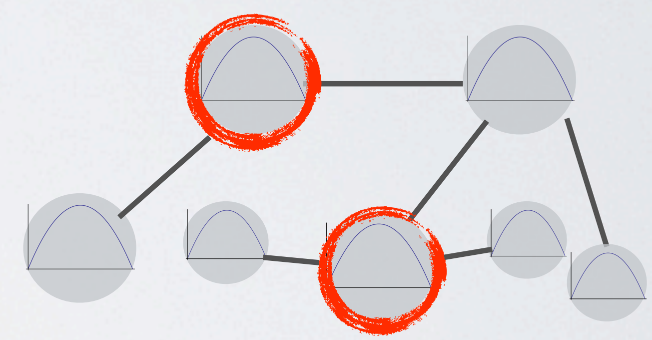
Networks Of Chaotic Maps:

A New Network Growth Model,

Inferring Topology From Symbolic Dynamics



Final Project
NLP, NCASO
June 4, 2010

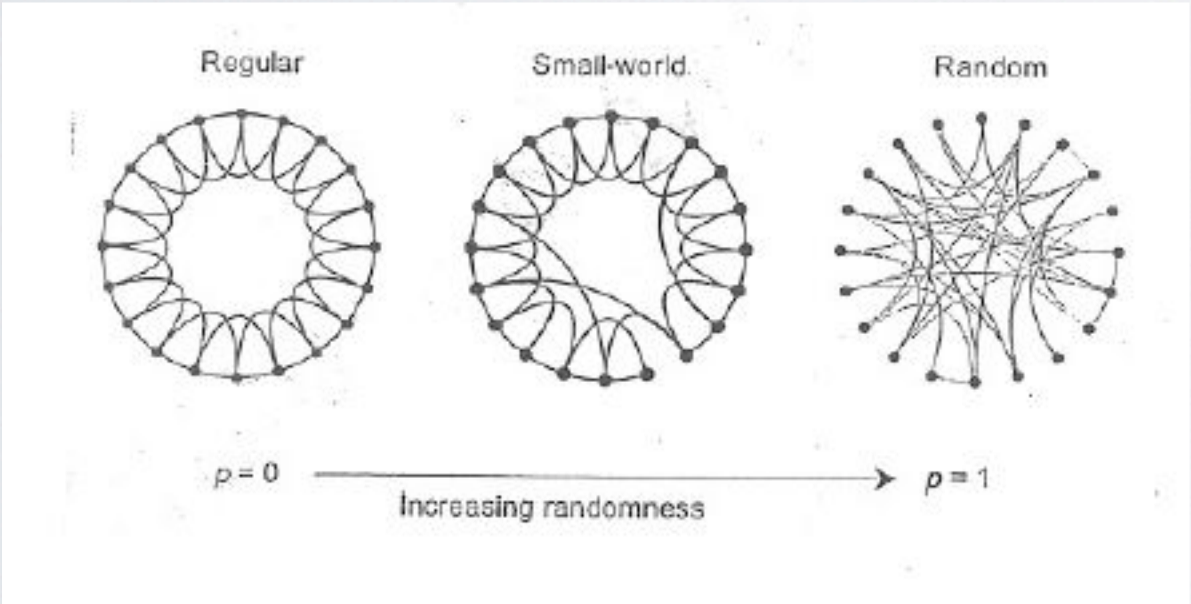


Charlie Brummitt
Graduate Group in Applied Math
cbrummitt@math.ucdavis.edu

CLASSIC NETWORK GROWTH MODELS

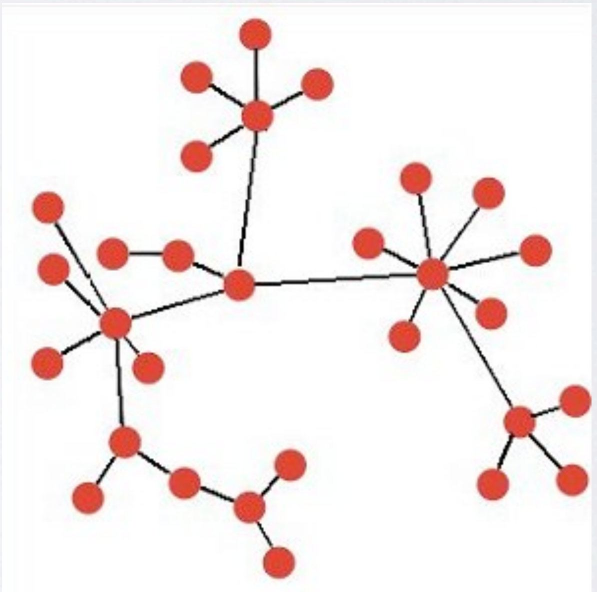
Watts-Strogatz

randomly, independently rewire links of a grid



Preferential Attachment

new nodes form links with probability proportional to target's degree

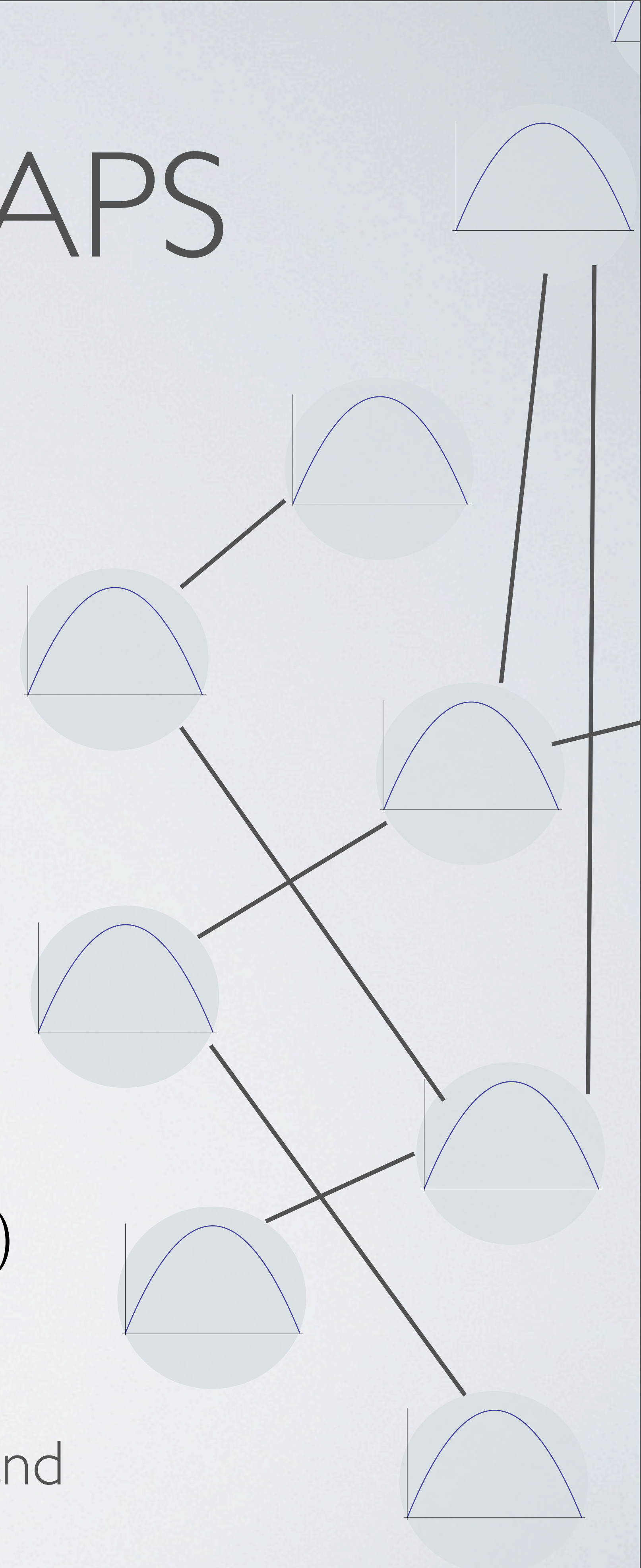


Generate "small-world" networks with properties like small diameter, high clustering, power-law degree distributions

DYNAMIC LINKS, DYNAMIC NODES

- Many network growth models have *dynamic* links but *static* nodes
- But in many real world systems nodes are dynamic
    - e.g., neurons, Internet routers, airports, people
- Motivates network growth models with dynamic nodes

NETWORK OF MAPS



- Undirected graph
- Nodes: logistic maps $f(x) = rx(1-x)$
  (or other one-dimensional maps)
- 300 nodes, 5000 edges (randomly chosen)
- Evolution:

coupling parameter

$$x_i(t+1) = (1-\epsilon)f(x_i(t)) + \frac{\epsilon}{k_i} \sum_{j \in B(i)} f(x_j(t))$$

degree

Each node evolves according to its own logistic map and an evenly weighted sum over its neighbors

GLOBAL REWIRING ALGORITHM

P. Gong and C. Van Leeuwen, 2004. D. van den Berg and C. van Leeuwen, 2004.

- Iterate for a transient so that the system's on an attractor
- Iterate for more time steps, and after each update, **rewire**:

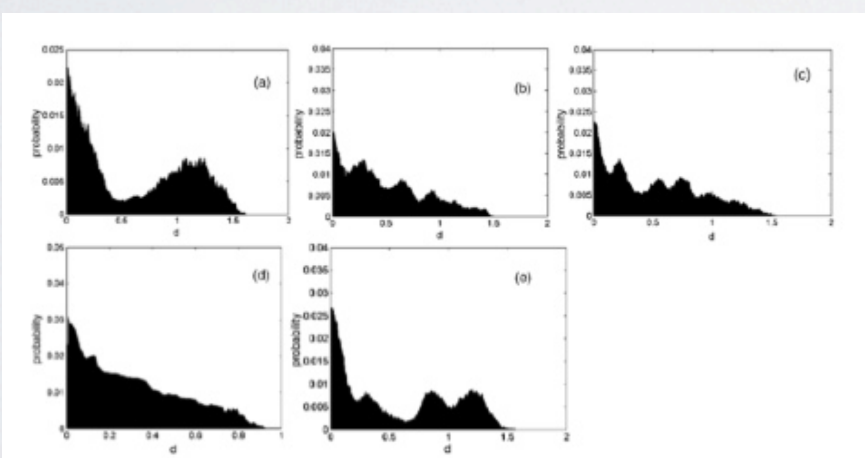
1. Choose a node at random (the "pivot")
2. Find which other node is most coherent with the pivot, i.e., which minimizes $d_{ij}(t) = |x_i(t) - x_j(t)|$ (the "candidate")
3. If the candidate is already connected to the pivot, do nothing
4. Else form a link between the two, and sever the link between the pivot and its least coherent neighbor (i.e., the neighbor k that maximizes $d_{ik}(t) = |x_i(t) - x_k(t)|$).

GLOBAL REWIRING ALGORITHM

Advantages

- toy model for neurogenesis
- interesting dynamics:
  below threshold population, dynamic in time:

coherences



  above threshold, settles onto small-world topology

Disadvantages

- artificial choice of who rewires (requires random number generator)
- unrealistic choice for targets of new links: link to the most coherent node, no matter where it is in the network

MODIFICATION (1 OF 2): WHO REWIRES?

- Nodes rewire whenever their total decoherence with their neighbors $\sum_{j \in B(i)} |x_j(t) - x_i(t)|$ exceeds a threshold **τ**.
- Interpretation: nodes out of sync with their neighbors get "fed up" and form a new link to achieve greater coherence
- **τ** controls the rate at which nodes rewire their links
  set **τ** high: nodes rarely rewire
  set **τ** low: nodes frequently rewire
- Benefit: the nodes truly are autonomous
  (no random number generator)

MODIFICATION (2 OF 2): WHO WILL BE MY NEW FRIEND?

- Before: form a new link with most coherent node,
  no matter where it is in the network
- Change: form a link with most coherent node
  at most σ *hops away*
- Interpretation: nodes form connections with friends' friends;
  other nodes are too far away to know about them
- Usually take σ = 2 or 3 (diameters of our networks ~3 or 4)

PYTHON IMPLEMENTATION

OneDimMaps.py

- LogisticMap
- TentMap
- CubicMap
- CuspMap

class MapNode

- 'Logistic'
- parameter
- state
- iterate

NetworkX
http://networkx.lanl.gov/

```
>>> import networkx as nx

>>> G=nx.Graph()
>>> G.add_node("spam")
>>> G.add_edge(1,2)
>>> print G.nodes()
[1, 2, 'spam']
>>> print G.edges()
[(1, 2)]
```

class NetworkOfMaps.py

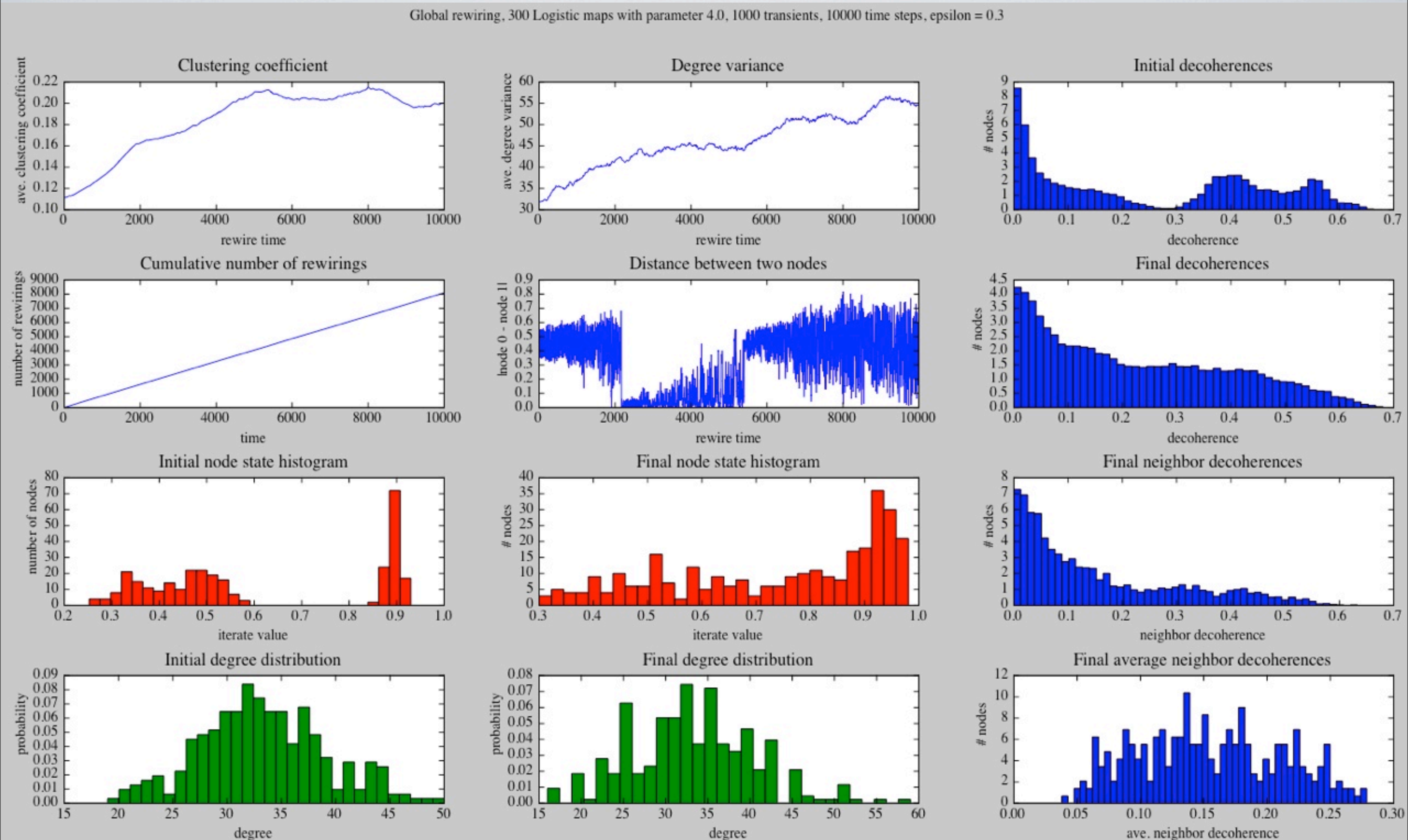
- import NetworkX
- nodes [0, 1, ..., N-1]
- graph G
- update(numTimes)
- globalRewire()
- localRewire(τ ,σ)

AnalyzeNetMapsGlobal.py

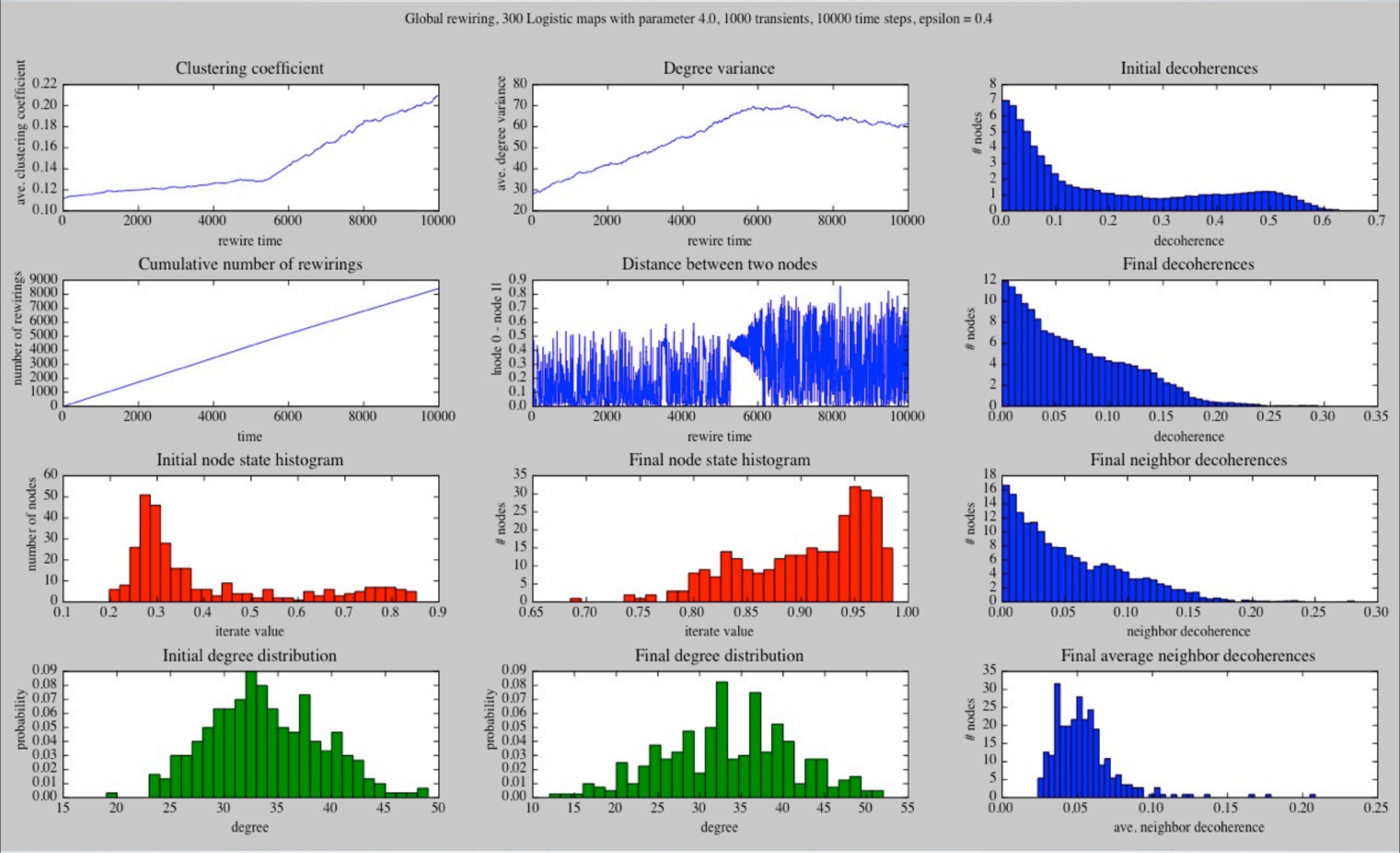
- net = NetworkOfMaps(...)
- net.updateRewire(10000)
- net.diameter()
- net.degreeVariance()
- net.degreeHistogram()
- net.decoherenceHistogram()

GLOBAL REWIRING RESULTS

- 300 nodes
- logistic map with $r = 4.0$
- 5000 edges initially chosen uniformly at random
- transient 1000, then update and rewire 10,000 time steps
- coupling constant $\epsilon = 0.3, 0.4$


Global rewiring, 300 Logistic maps with parameter 4.0, 1000 transients, 10000 time steps, epsilon = 0.3
Clustering coefficient
ave. clustering coefficient
rewire time
Degree variance
ave. degree variance
rewire time
Initial decoherences
nodes
decoherence
Cumulative number of rewirings
number of rewirings
time
Distance between two nodes
|node 0 - node 1|
rewire time
Final decoherences
nodes
decoherence
Initial node state histogram
number of nodes
iterate value
Final node state histogram
nodes
iterate value
Final neighbor decoherences
nodes
neighbor decoherence
Initial degree distribution
probability
degree
Final degree distribution
probability
degree
Final average neighbor decoherences
nodes
ave. neighbor decoherence

$$\epsilon = 0.3$$



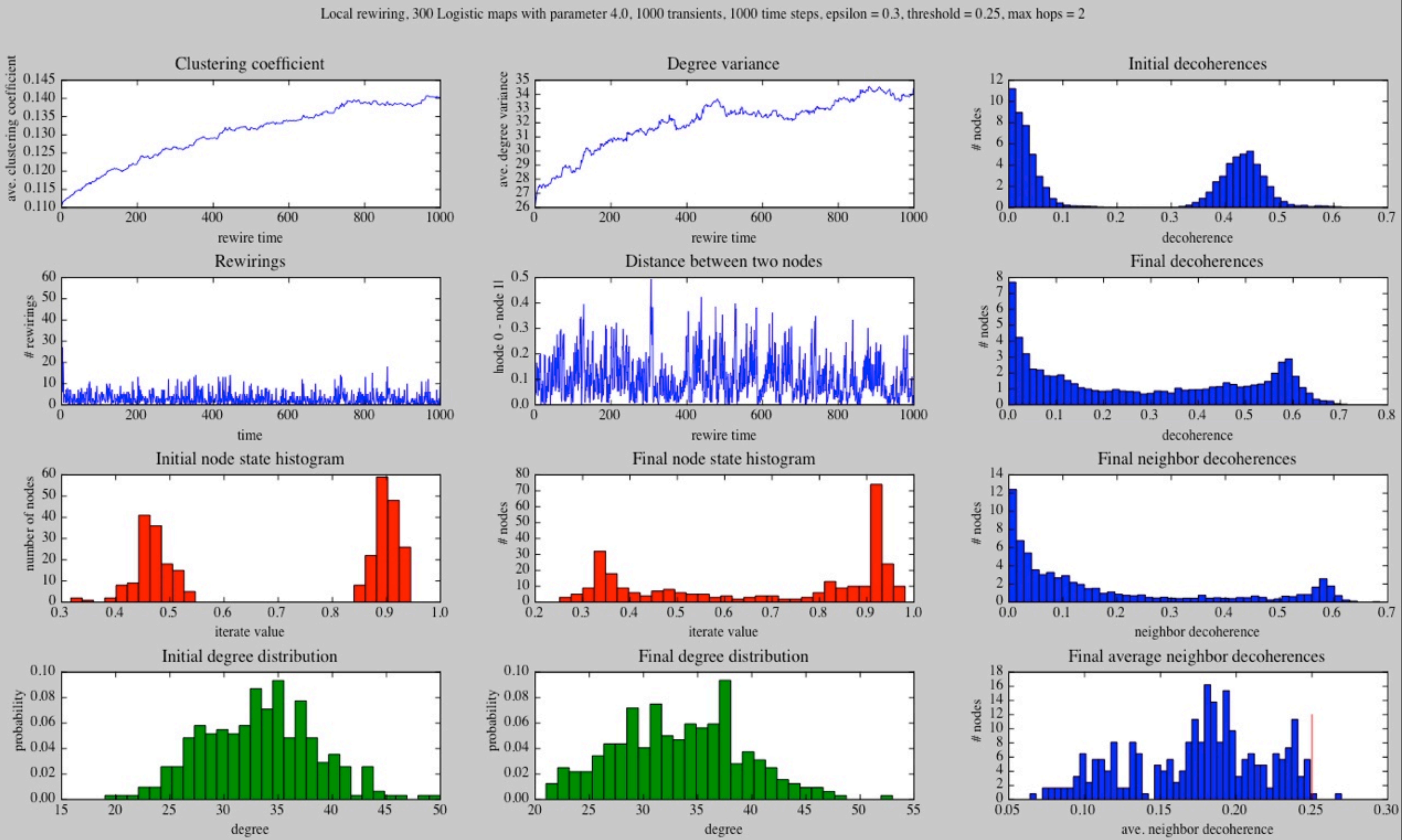
$$\epsilon = 0.4$$

(i.e., nodes depend more on neighbors)

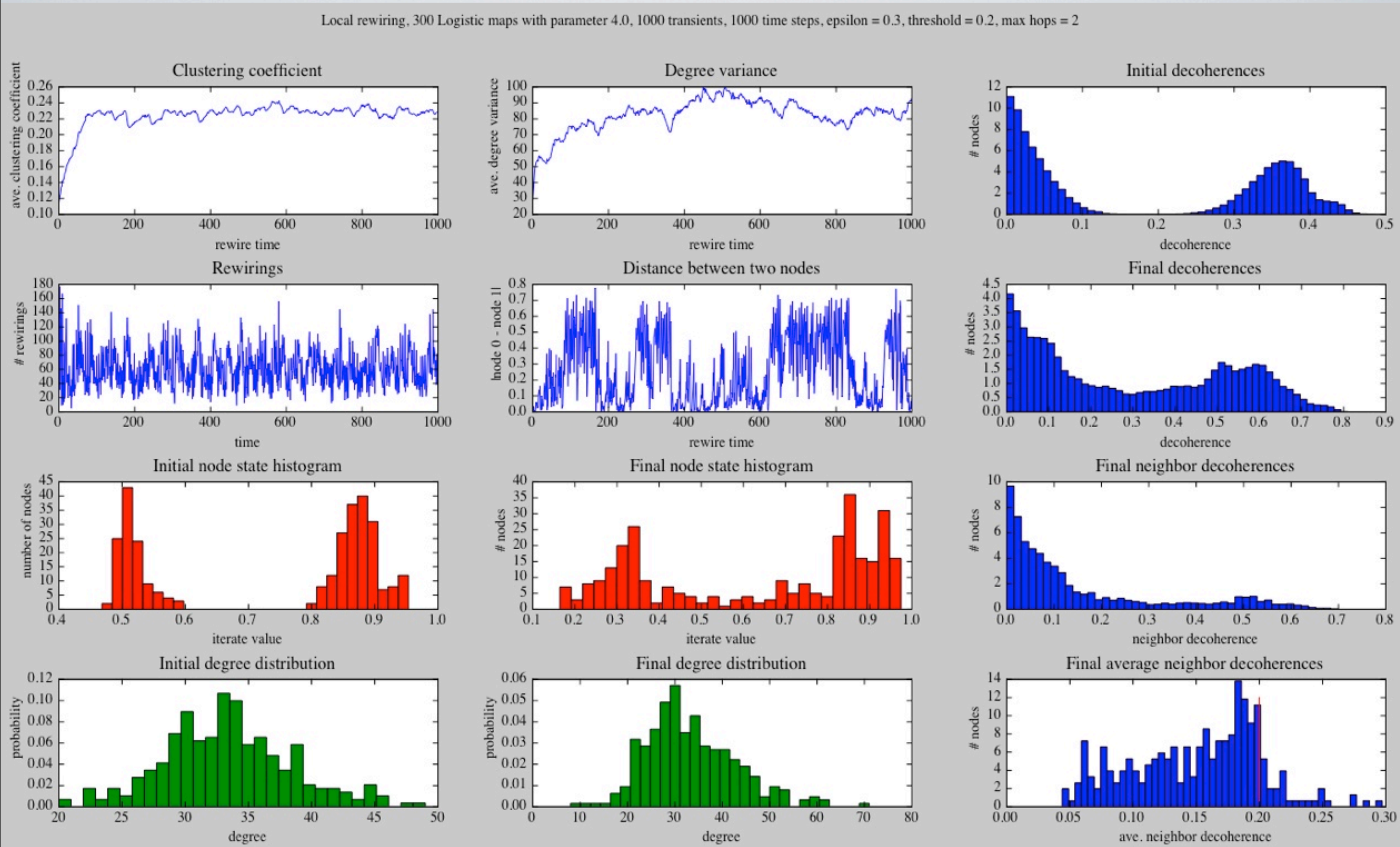
LOCAL REWIRING RESULTS

same setup as global rewiring

- 300 nodes
- logistic map with $r = 4.0$
- 5000 edges initially chosen uniformly at random
- transient 1000, then update and rewire 10,000 time steps
- coupling constant $\epsilon = 0.3$
- threshold $\tau = 0.25, 0.2$ max hops $\sigma = 2$ **NEW**


Local rewiring, 300 Logistic maps with parameter 4.0, 1000 transients, 1000 time steps, epsilon = 0.3, threshold = 0.25, max hops = 2
Clustering coefficient
ave. clustering coefficient
rewire time
Degree variance
ave. degree variance
rewire time
Initial decoherences
nodes
decoherence
Rewirings
rewirings
time
Distance between two nodes
|node 0 - node 1|
rewire time
Final decoherences
nodes
decoherence
Initial node state histogram
number of nodes
iterate value
Final node state histogram
nodes
iterate value
Final neighbor decoherences
nodes
neighbor decoherence
Initial degree distribution
probability
degree
Final degree distribution
probability
degree
Final average neighbor decoherences
nodes
ave. neighbor decoherence

$$\tau = 0.25$$



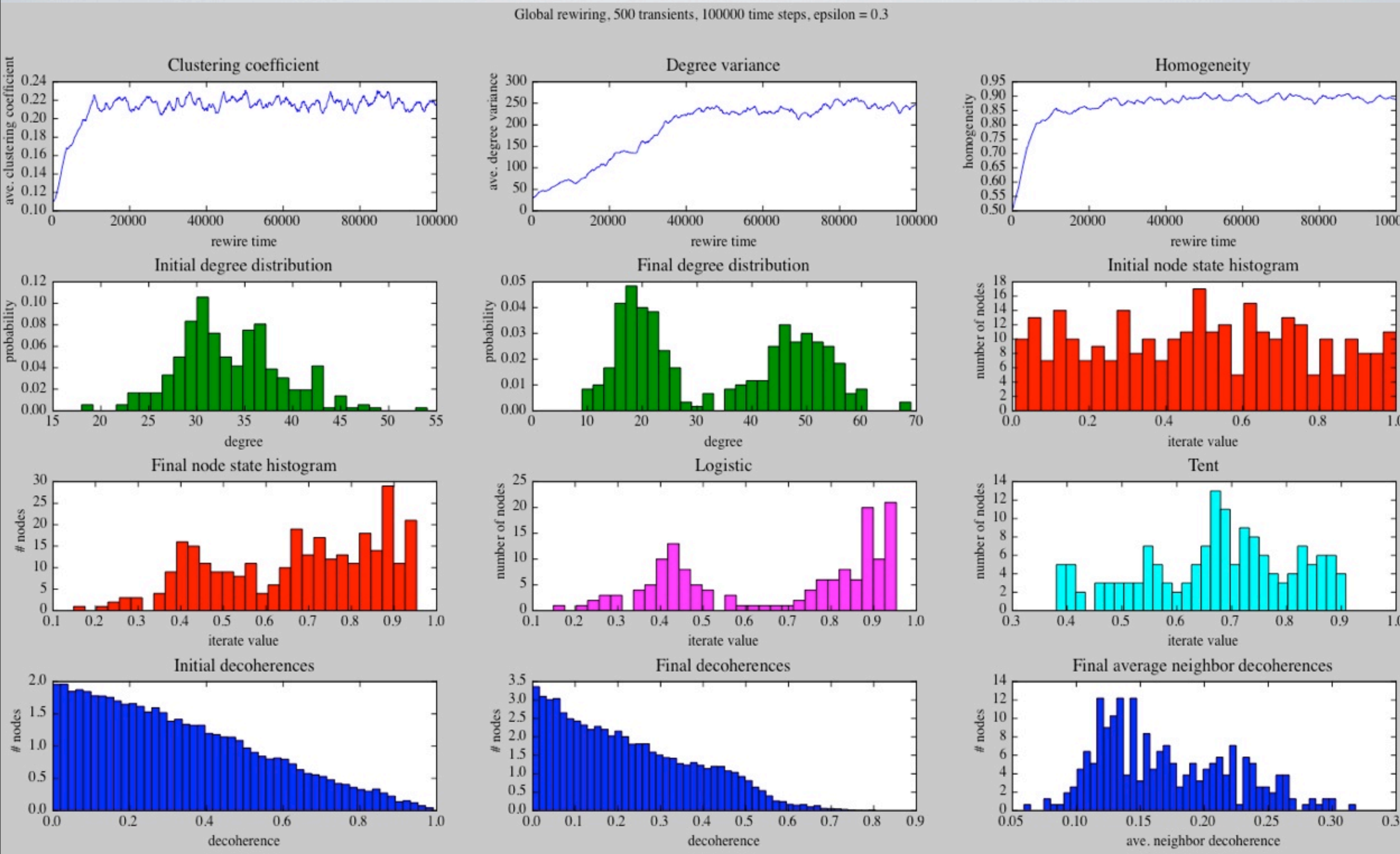
$$\tau = 0.2$$

(i.e., nodes rewire more)

- clustering coefficient: ~0.2
- more driving energy
- topology settles down faster

TWO MAPS ON A NETWORK

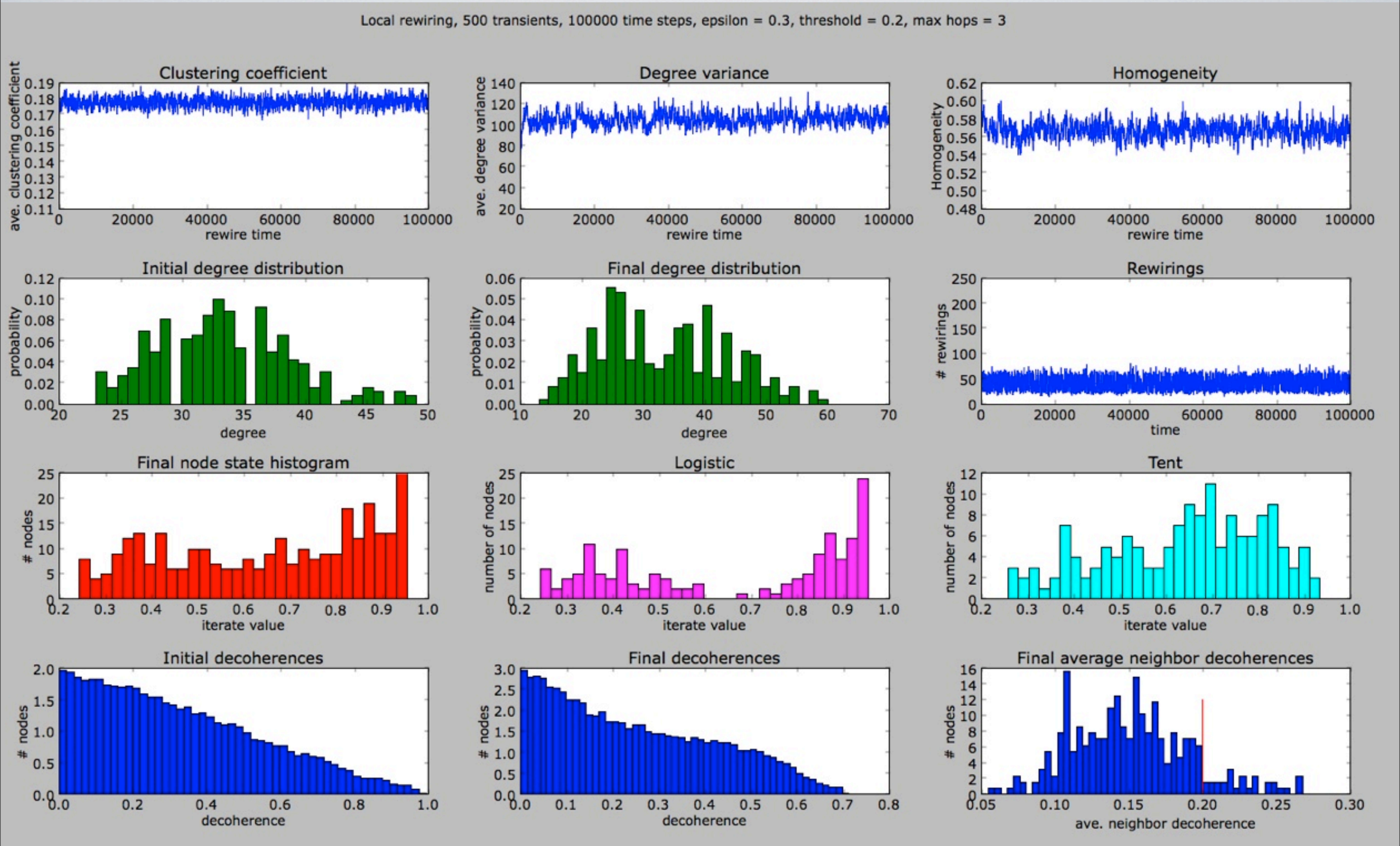
- Heterogeneous population
- Some nodes are **logistic** maps, others are **tent** maps
- Do they coalesce into homogeneous communities?
  - If so, how homogenous?
  - Could we distinguish the nodes from their iterates?
- <u>Heterogeneity</u> $:=$ fraction of edges that connect same nodes



Global

homogeneity ≈ 0.9

$\epsilon = 0.3$



$\epsilon = 0.3$
$\tau = 0.2, \sigma = 3$

Local

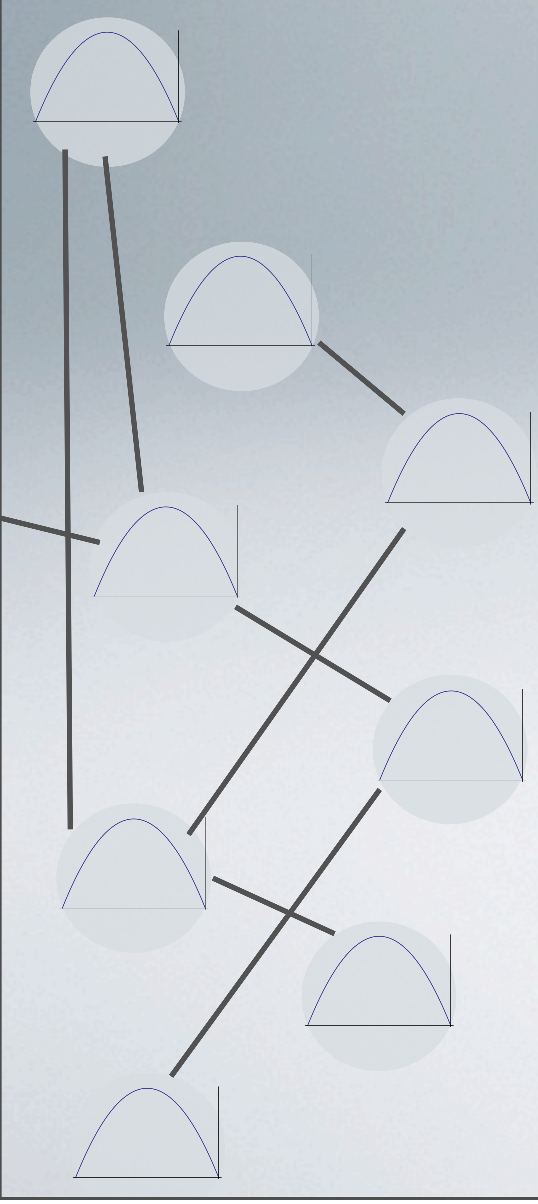
homogeneity ≈ 0.6

REWIRING CONCLUSIONS

- Each node is inherently chaotic
- Nodes try to "reign in their chaos" by associating with others who are similar
- New *local rewiring* algorithm:
  - Avoids unrealistic way of choosing who rewires and to whom
  - Achieves similar results: e.g., skewed degree distribution, high clustering ~0.2, interesting temporal dynamics
  - Depends heavily on threshold τ, not so much on max hops σ
- Two maps on the network coalesce into homogeneous communities (more so for global rewiring because nodes can search further)
- Local Rewiring => more rewirings => knock out of "ground state" (homogeneous connections)

PART II

COMPUTATIONAL MECHANICS OF A NETWORK OF MAPS



Can we infer the topology?

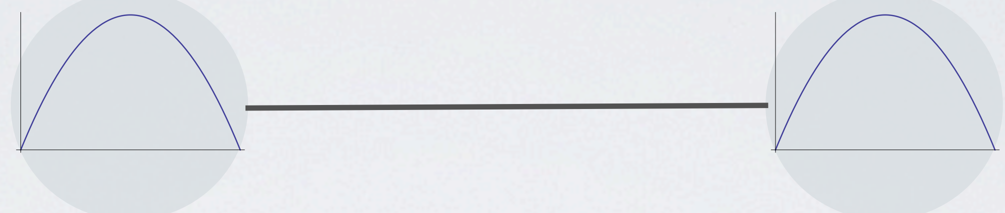
0.792 0.026 0.075 0.045...
0.004 0.38 0.183 0.755...
0.568 0.975 0.623 0.413...
0.022 0.893 0.103 0.646...
0.684 0.344 0.321 0.774...
0.902 0.793 0.782 0.854...
0.11 0.767 0.707 0.81...
0.107 0.387 0.524 0.055...

time series of iterates

B A A A...
A A A B...
B B B A...
A B A B...
B A A B...
B B B B...
A B B B...
A A B A...

symbols

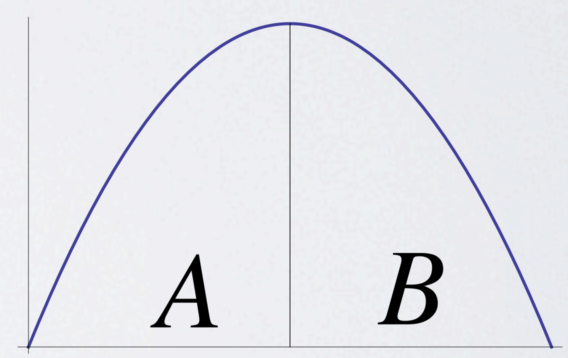
MUTUAL INFORMATION OF TWO COUPLED LOGISTIC MAPS



$$x_1(t+1) = (1-\epsilon)f(x_1(t)) + \epsilon f(x_2(t))$$

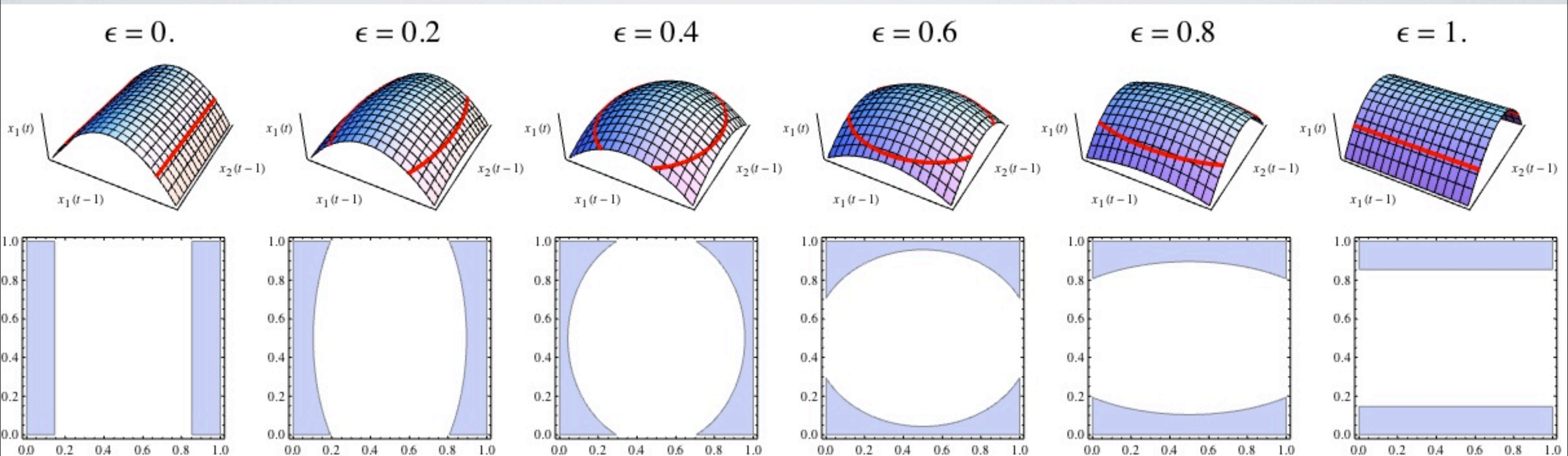
- Generating partition: decision point 1/2

$$x_1(t) = 0.38297512 \rightsquigarrow \sigma_1(t) = A$$



- Analytical expression for $I(\sigma_1(t); \sigma_2(t))$?

MUTUAL INFORMATION OF TWO COUPLED LOGISTIC MAPS



- Need to determine the area of the unit square that gets mapped above 1/2

  * weighted by the asymptotic distribution over the unit square *

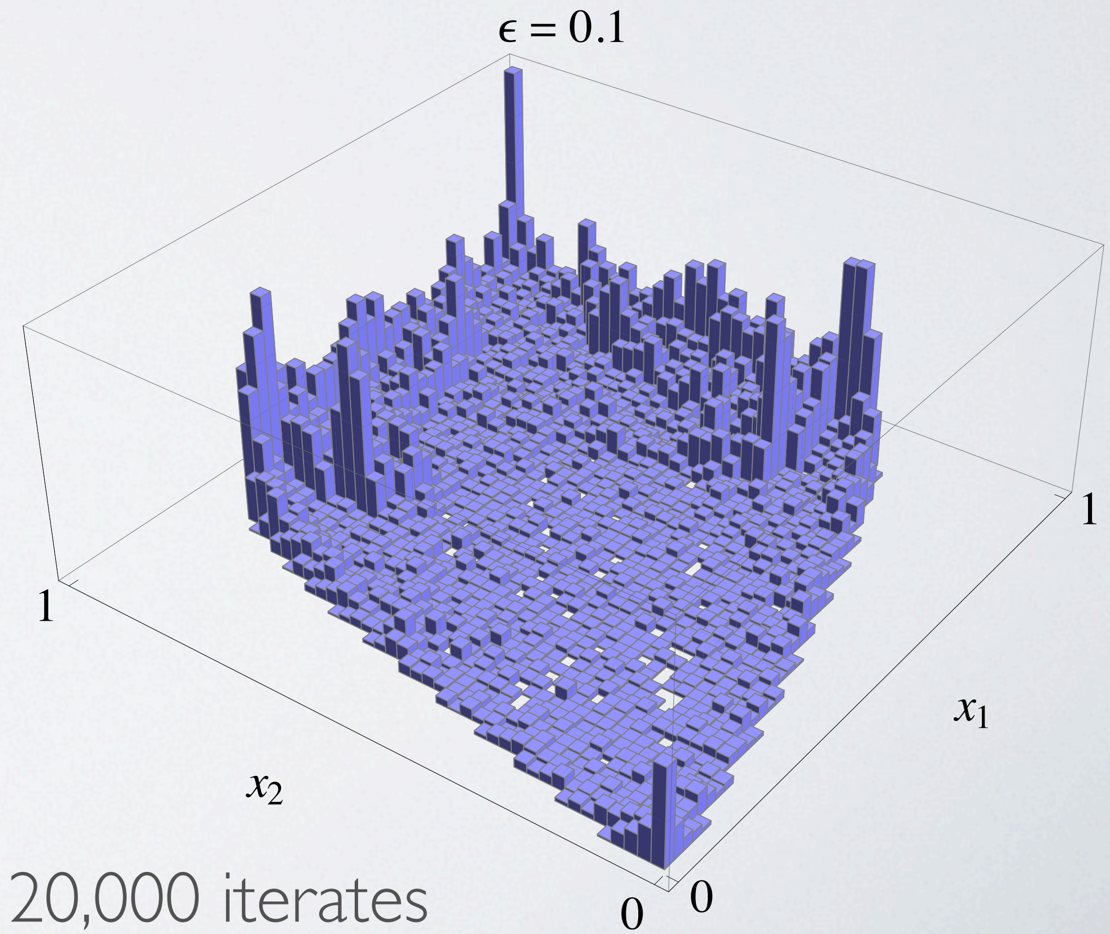
MUTUAL INFORMATION OF TWO COUPLED LOGISTIC MAPS

- *If* the two logistic maps sampled the unit interval uniformly, this would be feasible:

$$\mathrm{Area}(\epsilon) = \frac{\left(4\sqrt{1-\epsilon} - \sqrt{2-4\epsilon}\right)\sqrt{\epsilon} - \sin^{-1}\left(\sqrt{2}\sqrt{\epsilon}\right)}{4\sqrt{(1-\epsilon)\epsilon}}$$

- But the two maps asymptotically sample the unit square in a complicated way

- Analytical expression: hard



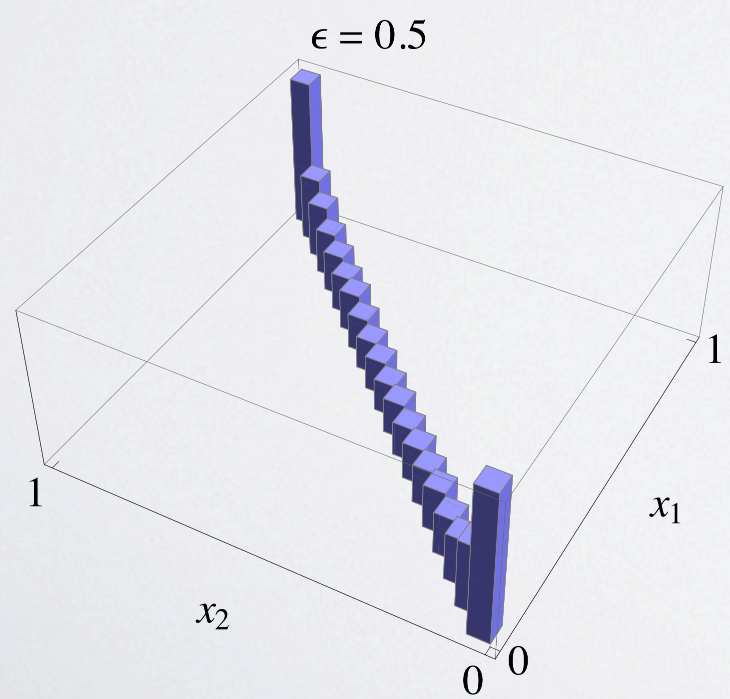
histogram of 20,000 iterates

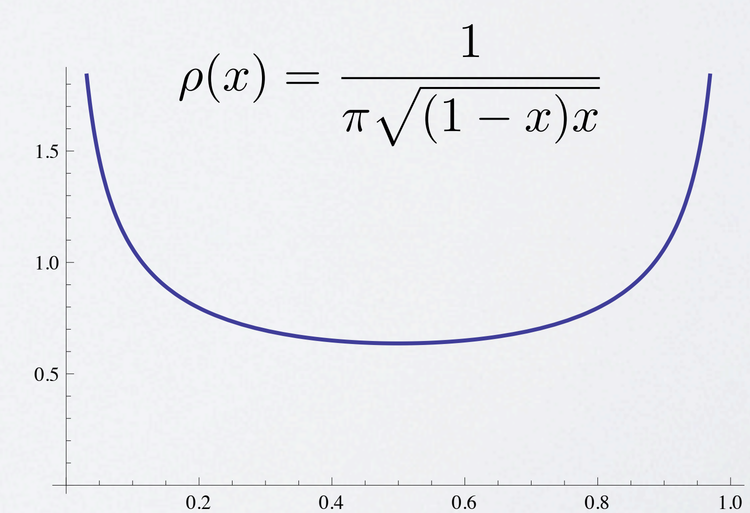
MUTUAL INFORMATION OF TWO COUPLED LOGISTIC MAPS

- Trivial case: $\epsilon = \frac{1}{2}$ $\quad x_1(t+1) = \frac{f(x_1(t)) + f(x_2(t))}{2} = x_2(t+1)$
- After one time step, they synchronize: $x_1(1) = x_2(1)$

$$\boxed{I(\sigma_1(t), \sigma_2(t)) = H(\sigma_1(t)) = 1}$$

(in expectation: $\langle\cdot\rangle_t$)





TRANSFER ENTROPY

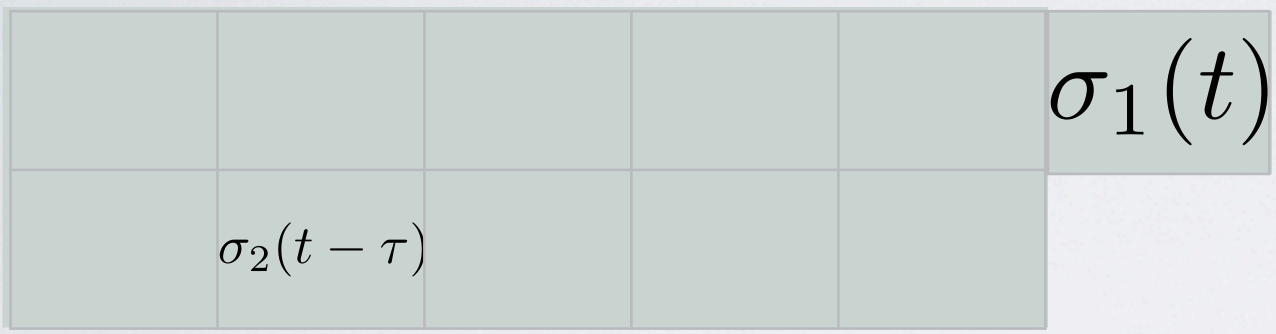
Node 1	$\sigma_1(t-5)$	$\sigma_1(t-4)$	$\sigma_1(t-3)$	$\sigma_1(t-2)$	$\sigma_1(t-1)$	$\sigma_1(t)$
Node 2	$\sigma_2(t-5)$	$\sigma_2(t-4)$	$\sigma_2(t-3)$	$\sigma_2(t-2)$	$\sigma_2(t-1)$	

X	X'
Y	

$$\boxed{T_{2\to 1} := I[X';Y|X]}$$

Info. theoretic measure of coherence of systems evolving in time

TIME-DELAYED MUTUAL INFO.



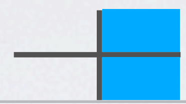
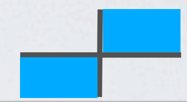
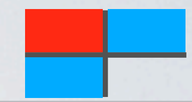
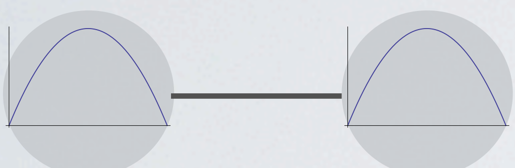
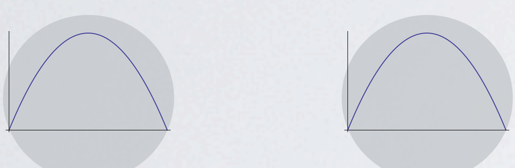
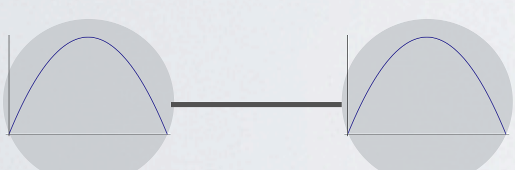
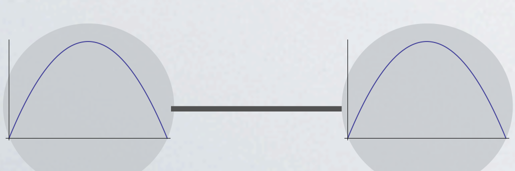
$$M_{1,2}(\tau) := I[\sigma_1(t); \sigma_2(t-\tau)]$$

- Transfer entropy is supposedly better because it distinguishes *exchanged information* from *shared info. due to common history* (Schreiber, Phys. Rev. Letters, 2000)

- I find mixed results...

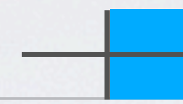
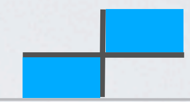
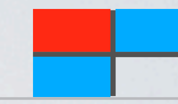
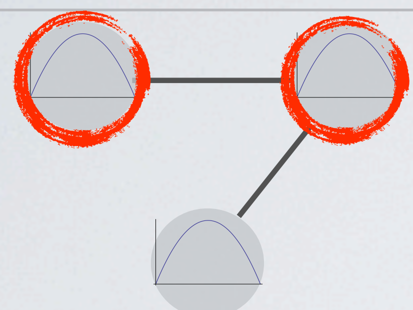
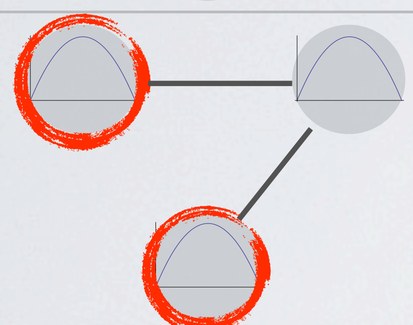
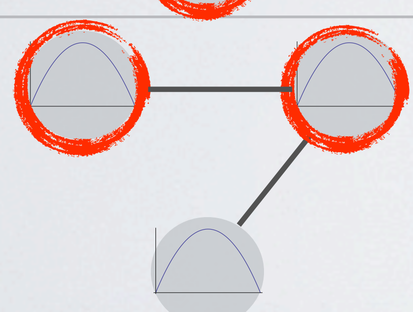
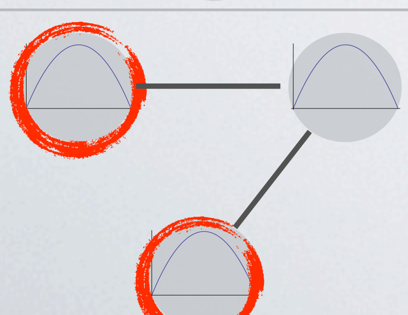
TWO NODES

- 1,000 transients
- 1,000,000 iterates
- r=4.0

		time-delayed mutual information		transfer entropy
Network	Coupling ϵ	$I[\sigma_1(t);\sigma_2(t)]$ 	$I[\sigma_1(t);\sigma_2(t-1)]$ 	$I[\sigma_1(t);\sigma_2(t-1)\vert\sigma_1(t-1)]$ 
	0.5	1.0	0.0000004	**0**
	0	0.000000001	0.0000001	0.0000001
	0.1	**0.005**	0.005	**0.05**
	0.2	**0.38**	0.04	**0.10**

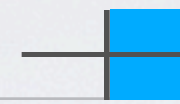
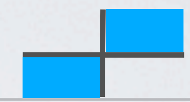
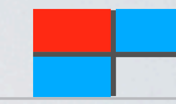
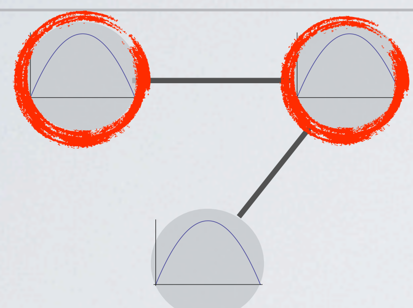
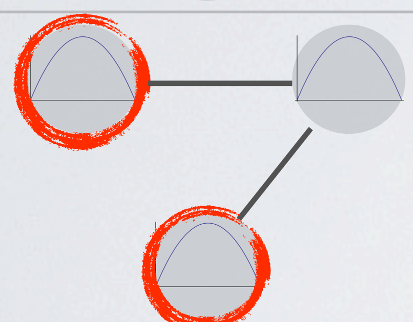
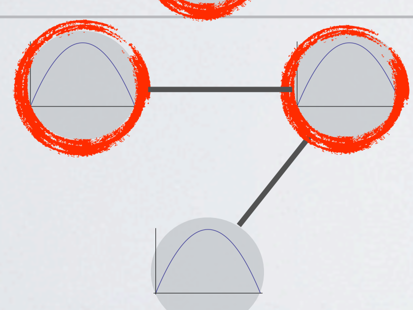
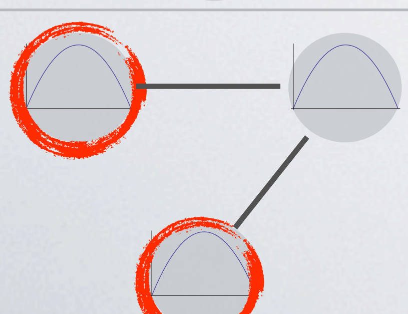
THREE NODES

- 1,000 transients
- 100,000 iterates
- r=4.0

		time-delayed mutual information		transfer entropy
Network	Coupling ϵ	$I[\sigma_1(t);\sigma_2(t)]$ 	$I[\sigma_1(t);\sigma_2(t-1)]$ 	$I[\sigma_1(t);\sigma_2(t-1)\|\sigma_1(t-1)]$ 
	0.5	1.0 sync	0.000001	0.00001
	0.5	1.0	0.000001	0.00001
	0.1	**0.001** bad	0.007	**0.05** good
	0.1	**0.01**	0.006	**0.002**

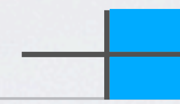
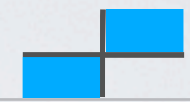
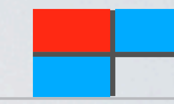
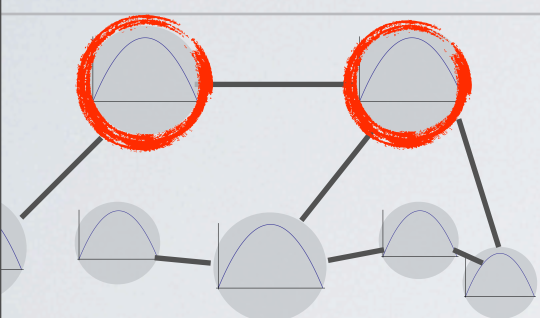
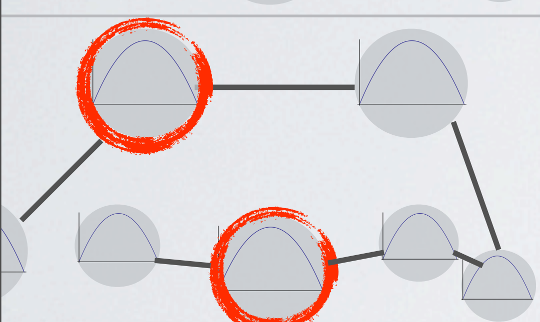
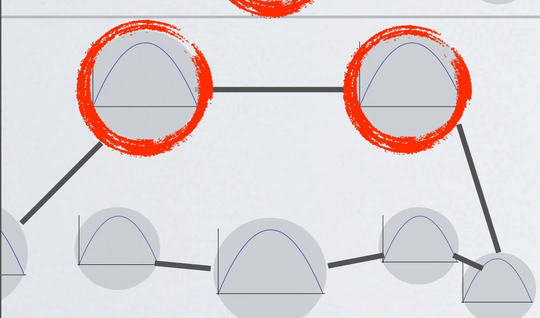
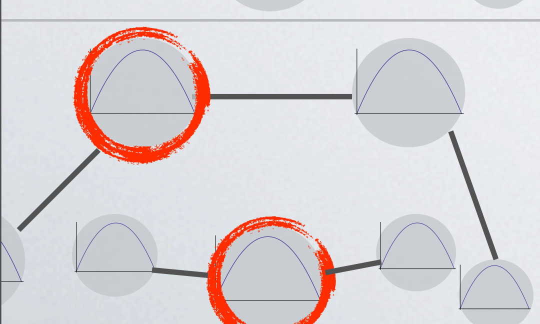
THREE NODES

- 1,000 transients
- 1,000,000 iterates
- r=4.0

		time-delayed mutual information		transfer entropy
Network	Coupling ϵ	$I[\sigma_1(t);\sigma_2(t)]$ 	$I[\sigma_1(t);\sigma_2(t-1)]$ 	$I[\sigma_1(t);\sigma_2(t-1)\vert\sigma_1(t-1)]$ 
	0.3	**0.30** ↕ good	0.04	**0.10** ↕ bad
	0.3	0.07	0.02	0.11
	0.6	1.0 ↕ sync	0.0000006	**0** ↕ bad
	0.6	1.0	0.0000006	**0**

300 NODES

- 1,000 transients
- 100,000 iterates
- r=4.0
- 5000 edges

		time-delayed mutual information		transfer entropy
Network	Coupling ϵ	$I[\sigma_1(t);\sigma_2(t)]$ 	$I[\sigma_1(t);\sigma_2(t-1)]$ 	$I[\sigma_1(t);\sigma_2(t-1)\vert\sigma_1(t-1)]$ 
	0.5	0.46	0.082	0.438
		↕ close		↕ close
	0.5	0.44	0.083	0.430
	0.2	0.19	0.25	0.07
		↕ bad		↕ good
	0.2	0.41	0.42	0.007

COMMENTS

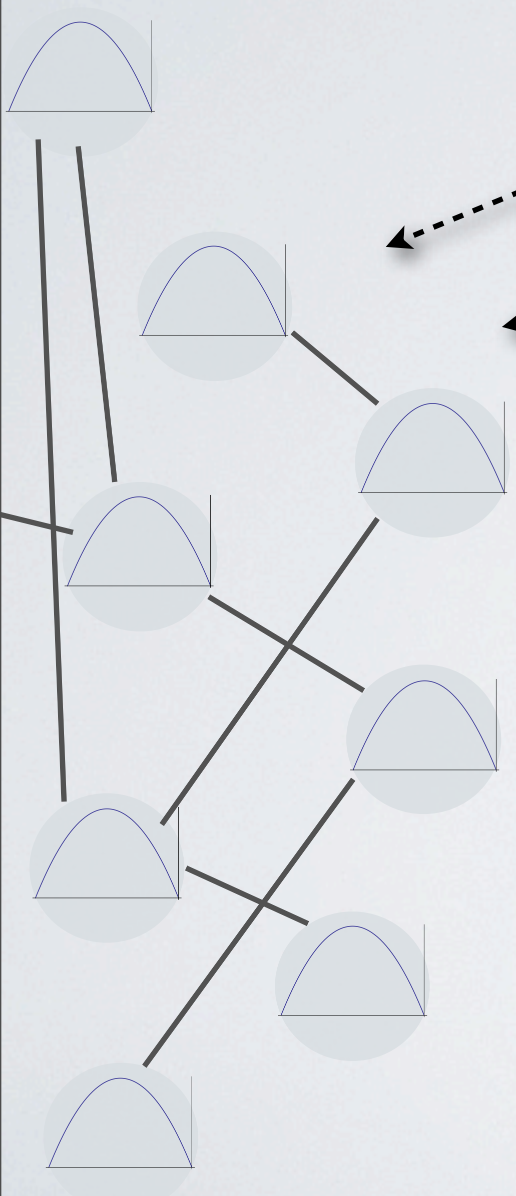
- Mutual information (no time delay) and transfer entropy can detect who's connected to whom, but not always
- Transfer entropy $\not>, \not<$ mutual information
- Future: try conditioning on more past symbols

$$I(\sigma_1(t); \sigma_2(t-1), ..., \sigma_2(t-k) | \sigma_1(t-1), ..., \sigma_1(t-k)$$

- Infer paths in the network? Utilize paths in the network?

CAUSAL STATE FILTERING

Easier to infer the topology?



0.792 0.026 0.075 0.045...
0.004 0.38 0.183 0.755...
0.568 0.975 0.623 0.413...
0.022 0.893 0.103 0.646...
0.684 0.344 0.321 0.774...
0.902 0.793 0.782 0.854...
0.11 0.767 0.707 0.81...
0.107 0.387 0.524 0.055...

iterates

B A A A...
A A A B...
B B B A...
A B A B...
B A A B...
B B B B...
A B B B...
A A B A...

symbols

inference

G F C D...
H F G C...
F C D C...
C B E E...
F G D E...
F C F G...
E E F G...
F C D G...

ε-machine states

CAUSAL STATE FILTERING

4 steps:

1. Infer ε-machines from nodes' symbolic output

2. Feed symbolic time series into ε-machine, synchronize to a state, then output the ensuing ε-machine states

$$\underbrace{ABBA}_{\text{synchronizing word}}AABAABA... \rightsquigarrow \underbrace{DFEGCDFDCE...}_{\epsilon\text{-machine states}}$$

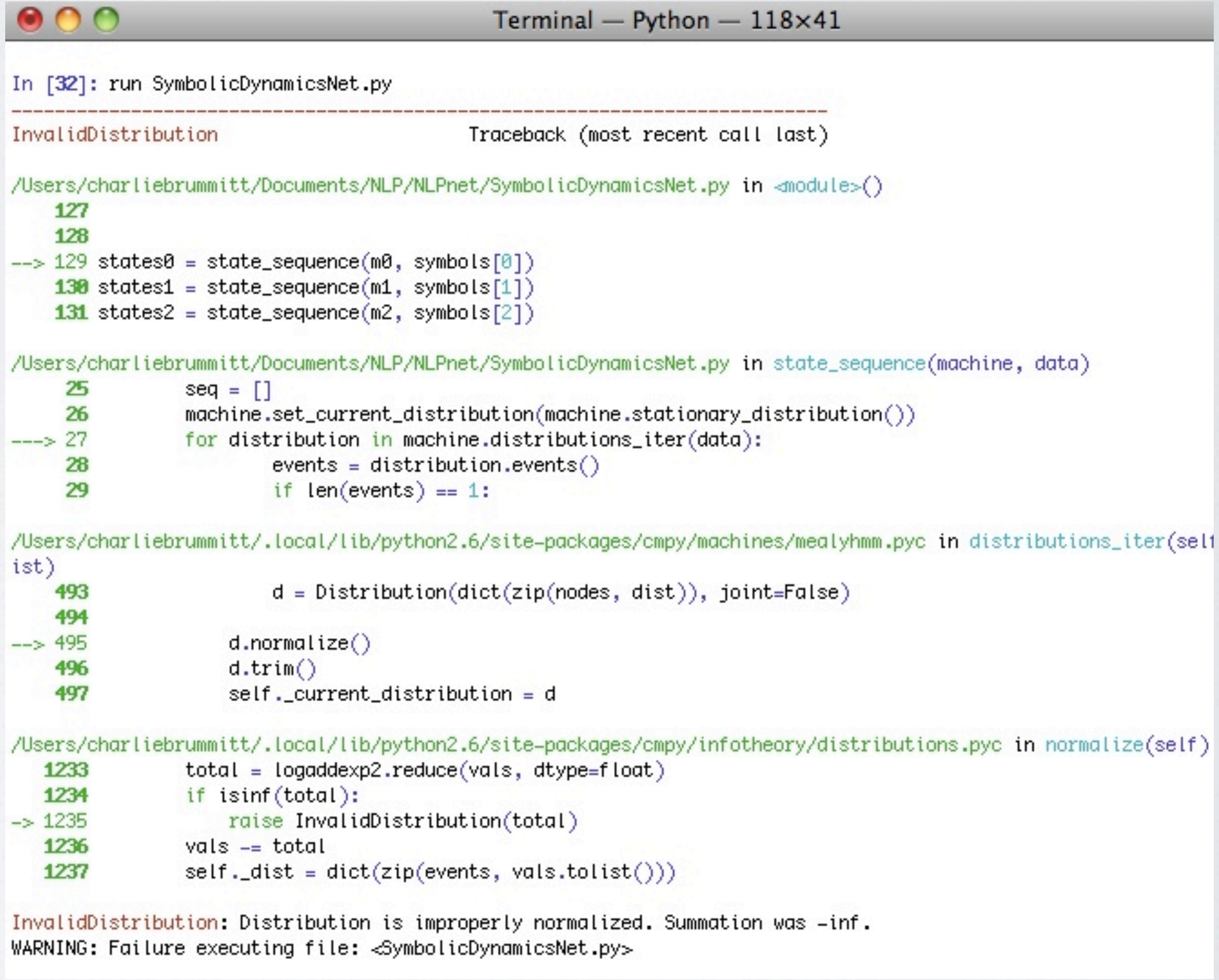
3. Compute mutual information & transfer entropy on ε-machine states time series

4. Infer the network topology

CAUSAL STATE FILTERING

- Inferring the **ε**-machine is hard
- When converting symbols to **ε**-machine states, I feed *forbidden words* into the **ε**-machine

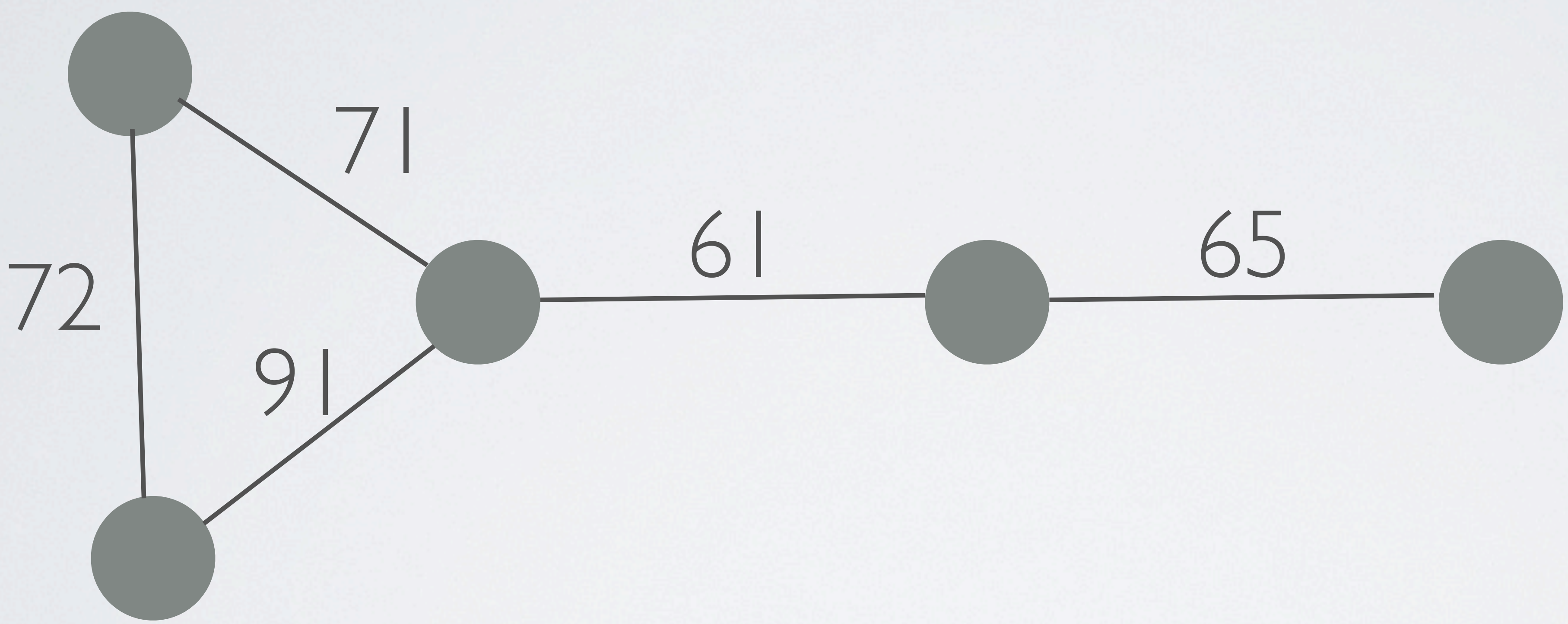
→ errors in CMPy



```
Terminal — Python — 118×41

In [32]: run SymbolicDynamicsNet.py
---------------------------------------------------------------------------
InvalidDistribution                       Traceback (most recent call last)

/Users/charliebrummitt/Documents/NLP/NLPnet/SymbolicDynamicsNet.py in <module>()
    127
    128
--> 129 states0 = state_sequence(m0, symbols[0])
    130 states1 = state_sequence(m1, symbols[1])
    131 states2 = state_sequence(m2, symbols[2])

/Users/charliebrummitt/Documents/NLP/NLPnet/SymbolicDynamicsNet.py in state_sequence(machine, data)
     25         seq = []
     26         machine.set_current_distribution(machine.stationary_distribution())
---> 27         for distribution in machine.distributions_iter(data):
     28                 events = distribution.events()
     29                 if len(events) == 1:

/Users/charliebrummitt/.local/lib/python2.6/site-packages/cmpy/machines/mealyhmm.pyc in distributions_iter(self, l
ist)
    493                 d = Distribution(dict(zip(nodes, dist)), joint=False)
    494
--> 495             d.normalize()
    496             d.trim()
    497             self._current_distribution = d

/Users/charliebrummitt/.local/lib/python2.6/site-packages/cmpy/infotheory/distributions.pyc in normalize(self)
   1233         total = logaddexp2.reduce(vals, dtype=float)
   1234         if isinf(total):
-> 1235             raise InvalidDistribution(total)
   1236         vals -= total
   1237         self._dist = dict(zip(events, vals.tolist()))

InvalidDistribution: Distribution is improperly normalized. Summation was -inf.
WARNING: Failure executing file: <SymbolicDynamicsNet.py>
```

- Reset to asymptotic distribution. But: comparing two time series difficult

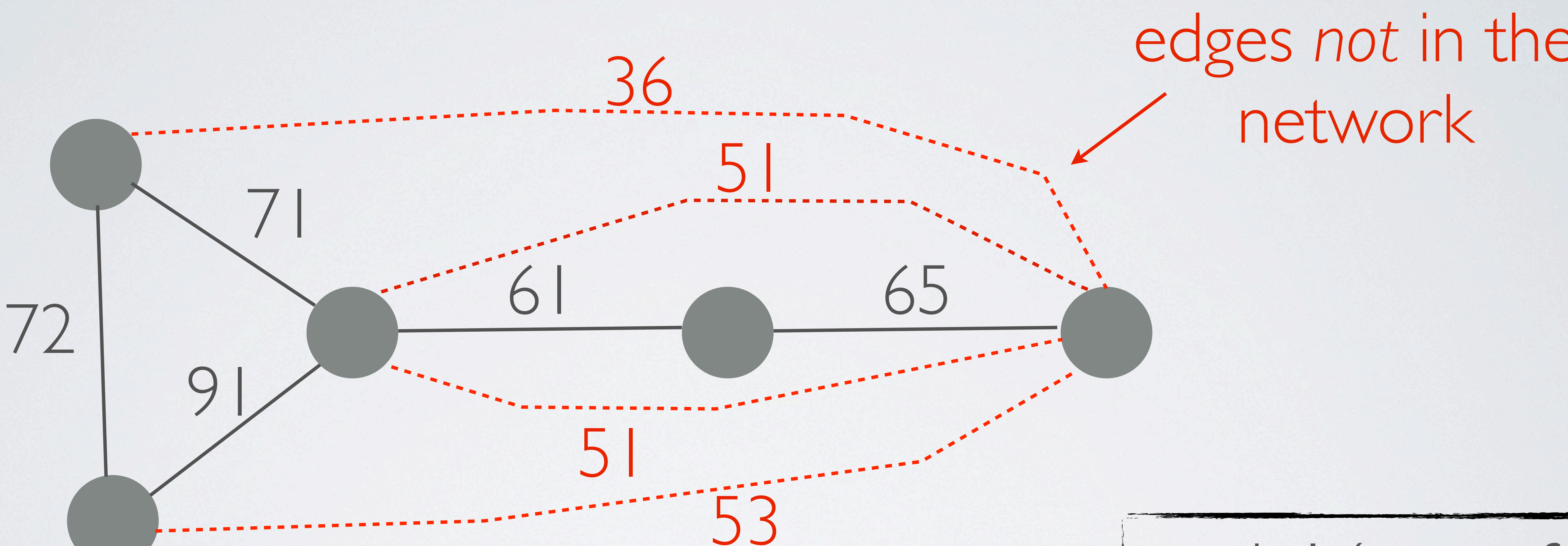
INFER NETWORK TOPOLOGY



edge labels: mutual info ×10^-3

100,000 iterates of data, **ε**=0.2

INFER NETWORK TOPOLOGY



works! (sort of)

edge labels: mutual info ×10^-3

100,000 iterates of data, **ε**=0.2

PART II: CONCLUSIONS

- Inferring topology from information measures is hard but has hope
- Mutual information (no time delay) seems to work better than transfer entropy
- Future:
  - Large networks of maps, other maps (tent, cusp, ...), other topologies (grids, grids with rewires, ...)
  - Information measures tailored toward graphs? e.g., use degree? paths?
  - Use time delay and more history to determine paths?
  - Network of chaotic ODEs: more tractable & coherent than logistic maps?