TWO MODELS OF NETWORKING SOCIAL AGENTS

Petter Holme[1] Gourab Ghoshal[2] Mark Newman[2]

[1]University of New Mexico

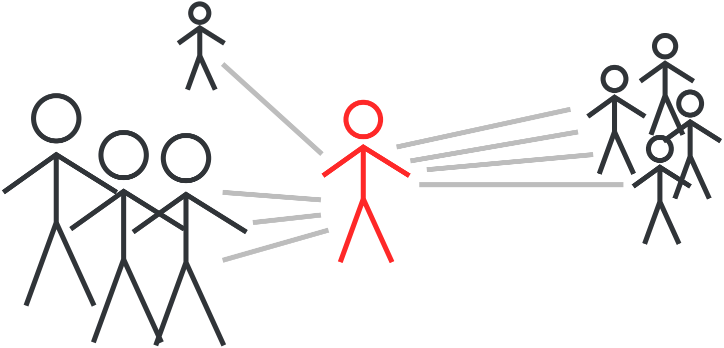
[2]University of Michigan

September 2006, Erice

http://www.cs.unm.edu/~holme/

HIGH CENTRALITY / LOW DEGREE: motivation

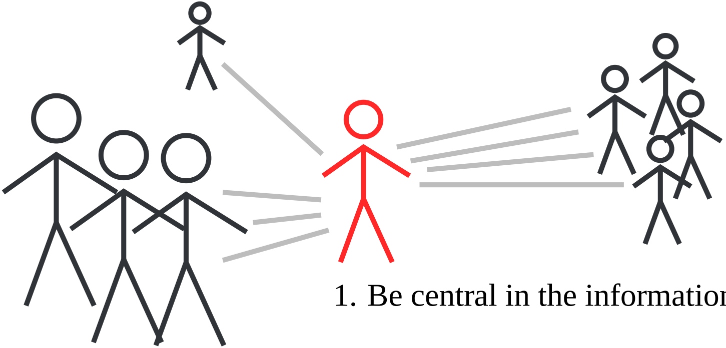
In diplomacy, lobbying or other political or corporate networking, it is important to:



Holme & Ghoshal, Phys. Rev. Lett. **96**, 098701 (2006)

motivation

In diplomacy, lobbying or other political or corporate networking, it is important to:

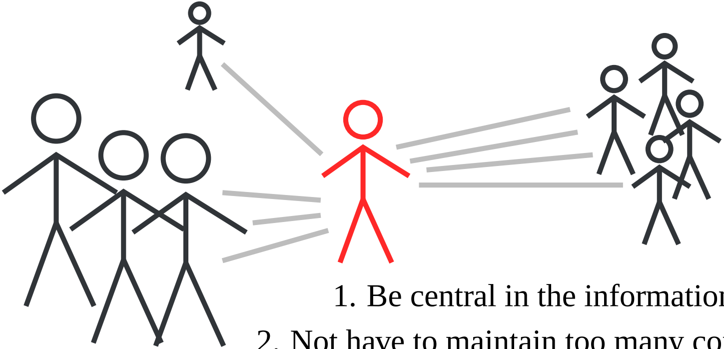


1. Be central in the information flow.

Holme & Ghoshal, Phys. Rev. Lett. **96**, 098701 (2006)

motivation

In diplomacy, lobbying or other political or corporate networking, it is important to:



1. Be central in the information flow.
2. Not have to maintain too many contacts.

Holme & Ghoshal, Phys. Rev. Lett. **96**, 098701 (2006)

score function

- Central is good—*closeness centrality*

$$C(i) = (N-1)/\sum_{j\neq i} d(i,j)$$

- If the network is disconnected, being a part of a large component is good.
- Large degree is bad.

score function

- Central is good—*closeness centrality*

$$C(i) = (N-1)/\sum_{j\neq i} d(i,j)$$

- If the network is disconnected, being a part of a large component is good.
- Large degree is bad.

score function

- Central is good—*closeness centrality*

$$C(i) = (N-1) \Big/ \sum_{j \neq i} d(i,j)$$

- If the network is disconnected, being a part of a large component is good.
- Large degree is bad.

score function

- Central is good—*closeness centrality*

$$C(i) = (N-1) / \sum_{j \neq i} d(i,j)$$

- If the network is disconnected, being a part of a large component is good.
- Large degree is bad.

score function

Component size can be incorporated by modifying the definition of closeness: If we sum the reciprocals (instead of inverting the sum), we get the score function:

Definition

$$s(i) = \begin{cases} (1/k_i) \sum_{H_i} 1/d(i,j) & \text{if } k_i > 0 \\ 0 & \text{if } k_i = 0 \end{cases} \quad (1)$$

H_i is the component i belongs to, except i

score function

Component size can be incorporated by modifying the definition of closeness: If we sum the reciprocals (instead of inverting the sum), we get the score function:

Definition

$$s(i) = \begin{cases} (1/k_i) \sum_{H_i} 1/d(i,j) & \text{if } k_i > 0 \\ 0 & \text{if } k_i = 0 \end{cases} \quad (1)$$

H_i is the component i belongs to, except i

score function

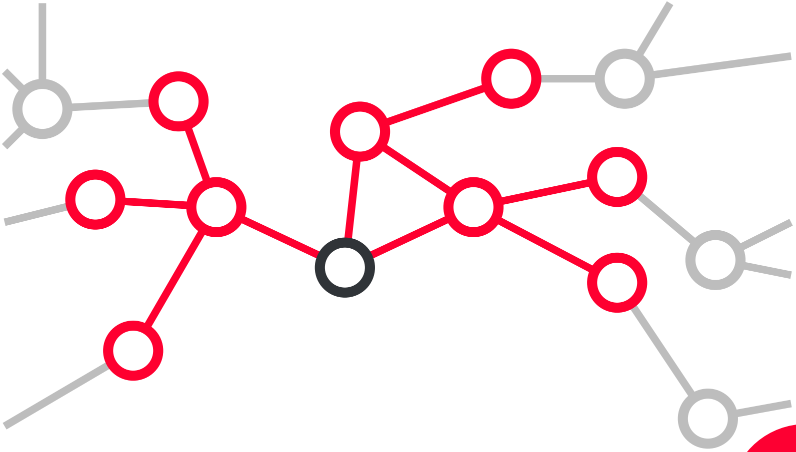
Component size can be incorporated by modifying the definition of closeness: If we sum the reciprocals (instead of inverting the sum), we get the score function:

Definition

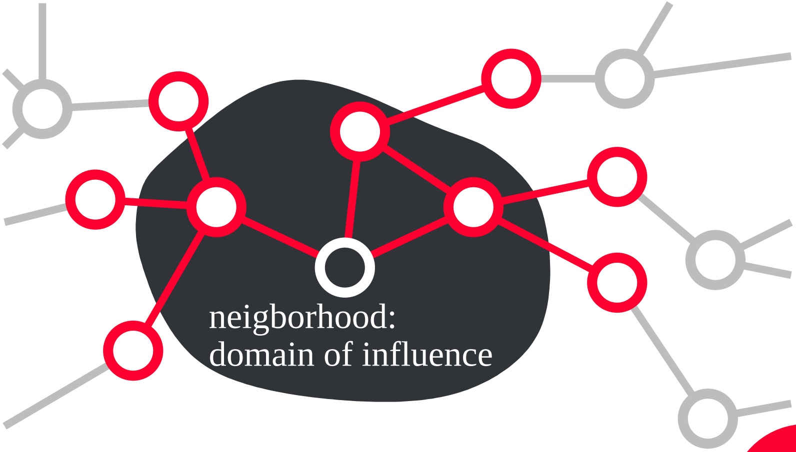
$$s(i) = \begin{cases} (1/k_i) \sum_{H_i} 1/d(i,j) & \text{if } k_i > 0 \\ 0 & \text{if } k_i = 0 \end{cases} \qquad (1)$$

H_i is the component i belongs to, except i

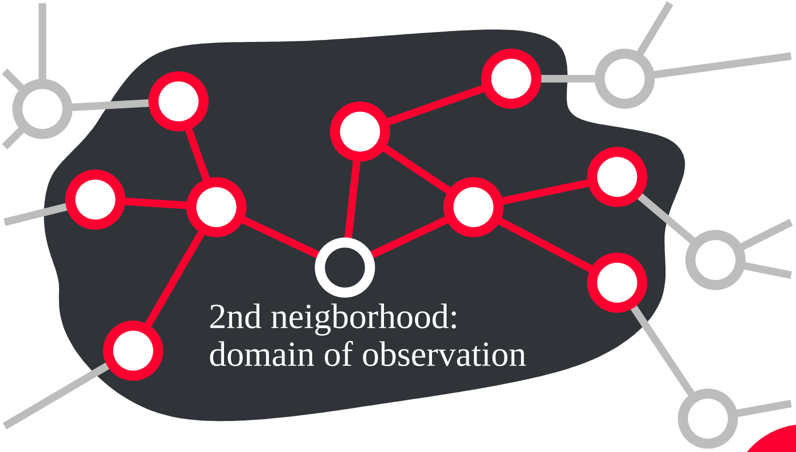
moves



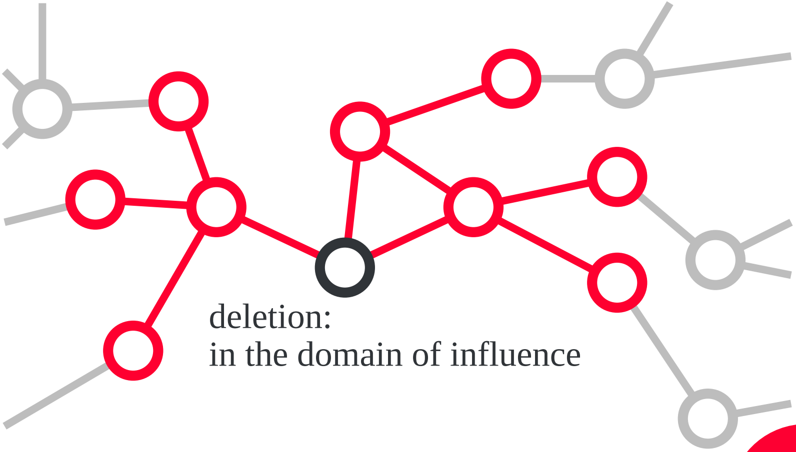
moves



moves

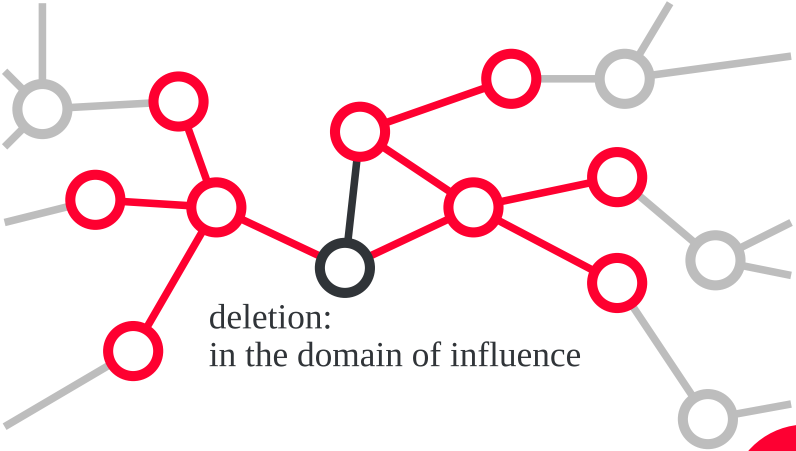


moves



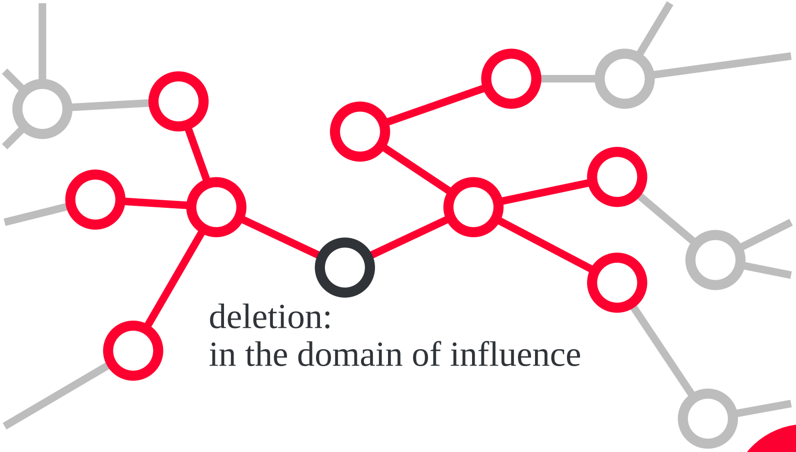
deletion:
in the domain of influence

moves

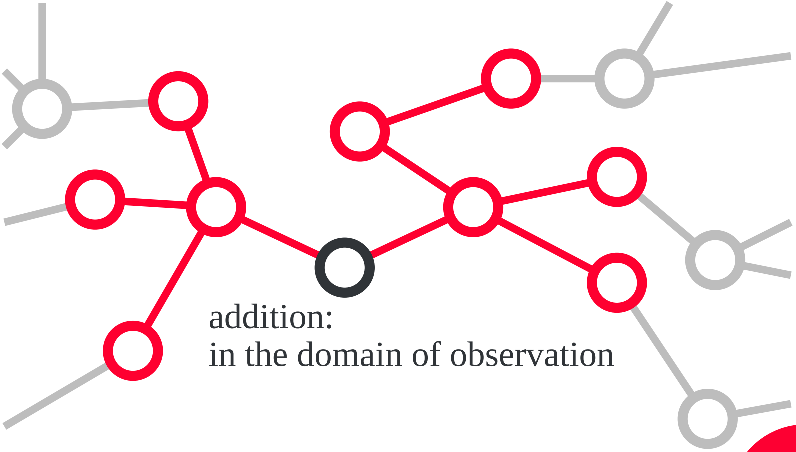


deletion:
in the domain of influence

moves

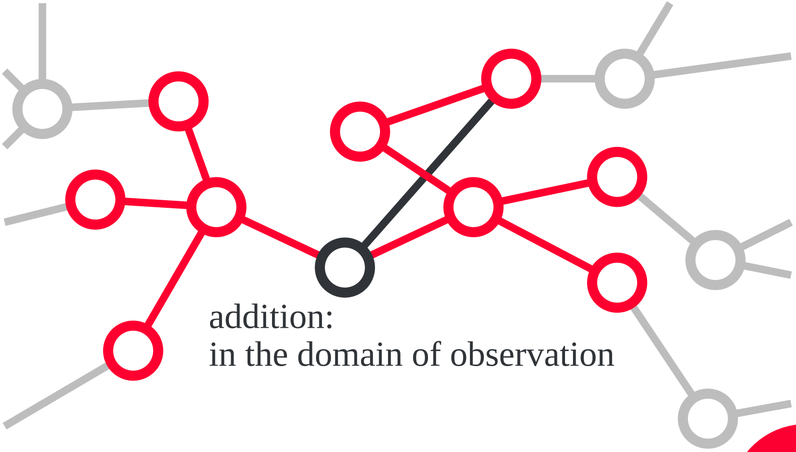


moves



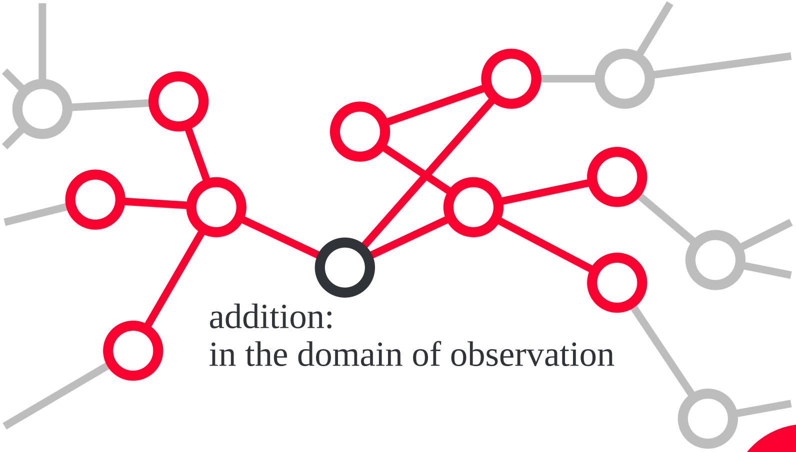
addition:
in the domain of observation

moves



addition:
in the domain of observation

moves



addition:
in the domain of observation

actions: deciding what move to make

MAXD Choose the vertex two steps away with largest degree.

MIND Choose the vertex two steps away with smallest degree.

MAXC Choose the vertex two steps away with highest centrality.

MINC Choose the vertex two steps away with lowest centrality.

RND Choose a random vertex two steps away.

NO Do not make the move.

actions: deciding what move to make

- **MAXD** Choose the vertex two steps away with largest degree.
- **MIND** Choose the vertex two steps away with smallest degree.
- **MAXC** Choose the vertex two steps away with highest centrality.
- **MINC** Choose the vertex two steps away with lowest centrality.
- **RND** Choose a random vertex two steps away.
- **NO** Do not make the move.

actions: deciding what move to make

MAXD Choose the vertex two steps away with largest degree.

MIND Choose the vertex two steps away with smallest degree.

MAXC Choose the vertex two steps away with highest centrality.

MINC Choose the vertex two steps away with lowest centrality.

RND Choose a random vertex two steps away.

NO Do not make the move.

actions: deciding what move to make

MAXD	Choose the vertex two steps away with largest degree.
MIND	Choose the vertex two steps away with smallest degree.
MAXC	Choose the vertex two steps away with highest centrality.
MINC	Choose the vertex two steps away with lowest centrality.
RND	Choose a random vertex two steps away.
NO	Do not make the move.

actions: deciding what move to make

- **MAXD** Choose the vertex two steps away with largest degree.
- **MIND** Choose the vertex two steps away with smallest degree.
- **MAXC** Choose the vertex two steps away with highest centrality.
- **MINC** Choose the vertex two steps away with lowest centrality.
- **RND** Choose a random vertex two steps away.
- **NO** Do not make the move.

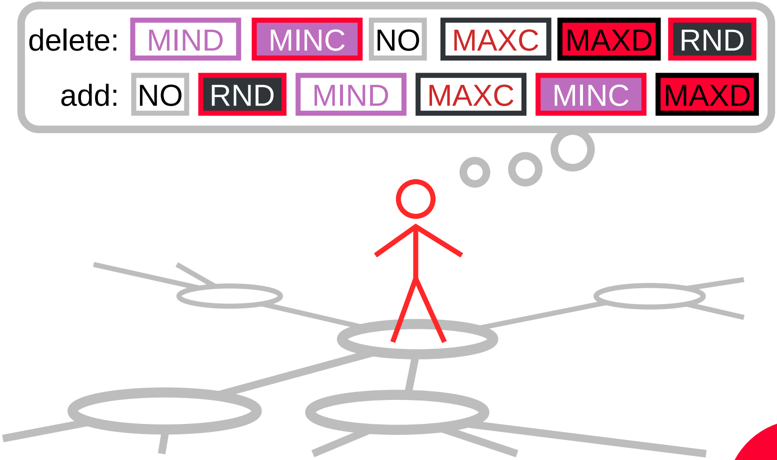
actions: deciding what move to make

- **MAXD** Choose the vertex two steps away with largest degree.
- **MIND** Choose the vertex two steps away with smallest degree.
- **MAXC** Choose the vertex two steps away with highest centrality.
- **MINC** Choose the vertex two steps away with lowest centrality.
- **RND** Choose a random vertex two steps away.
- **NO** Do not make the move.

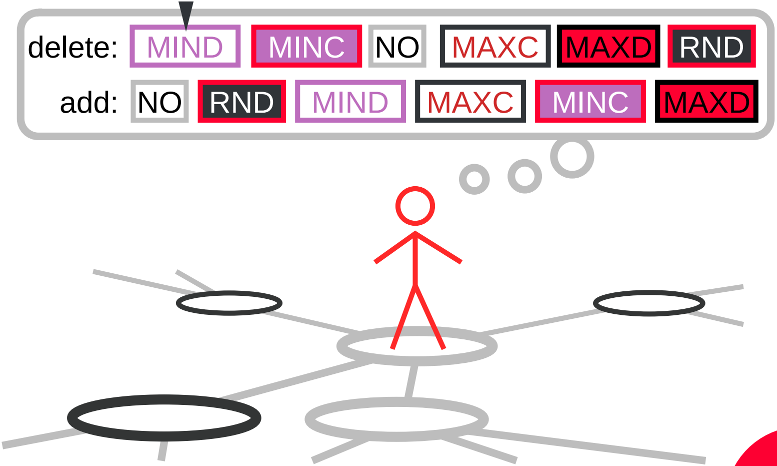
actions: deciding what move to make

- **MAXD** Choose the vertex two steps away with largest degree.
- **MIND** Choose the vertex two steps away with smallest degree.
- **MAXC** Choose the vertex two steps away with highest centrality.
- **MINC** Choose the vertex two steps away with lowest centrality.
- **RND** Choose a random vertex two steps away.
- **NO** Do not make the move.

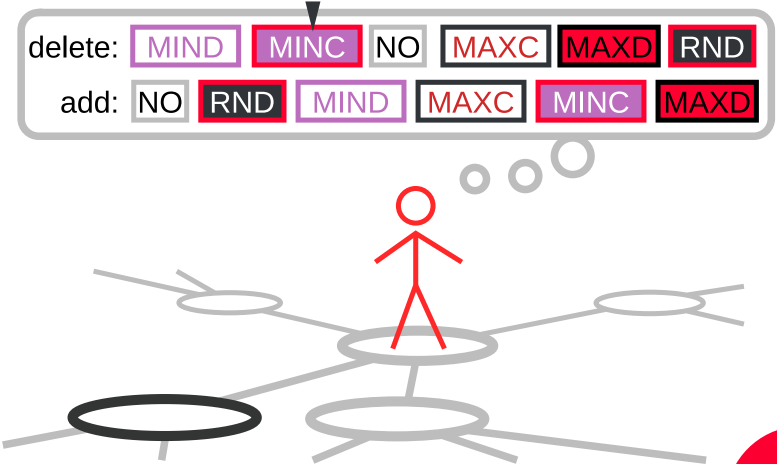
strategies: sequences of actions



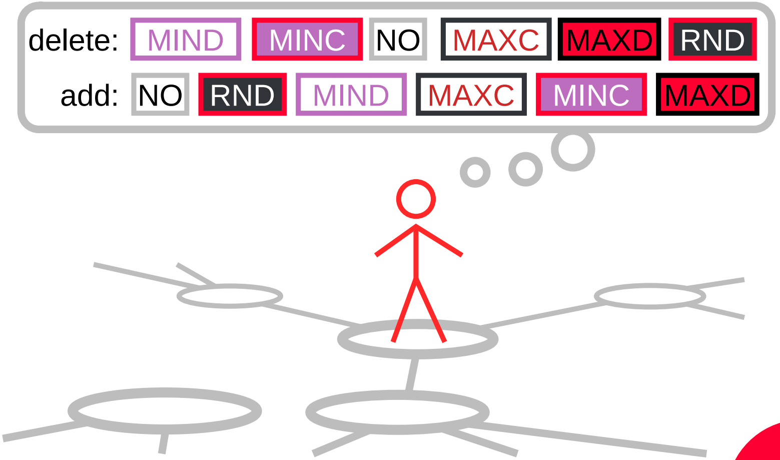
strategies: sequences of actions



strategies: sequences of actions



strategies: sequences of actions



global flow of the simulation

1. Initialize the network to an Erdős-Rényi network with N vertices and M_0 edges.
2. Use random permutations of the six actions as $\mathbf{s}_{\mathrm{add}}$ and $\mathbf{s}_{\mathrm{del}}$ for all vertices.
3. Calculate the score for all vertices.
4. Update the vertices synchronously by adding and deleting edges as selected by the strategy vectors. With probability p_r an edge is added to a random vertex instead of a neighbor's neighbor.
5. Every t_{strat}'th timestep, update the strategy vectors. For each vertex, with probability p_s, swap two elements in a vertex' strategy vector.
6. Increment the simulation time t and, if $t < t_{\mathrm{tot}}$, go to step 3.

global flow of the simulation

1. Initialize the network to an Erdős-Rényi network with N vertices and M_0 edges.
2. Use random permutations of the six actions as $\mathbf{s}_{\mathrm{add}}$ and $\mathbf{s}_{\mathrm{del}}$ for all vertices.
3. Calculate the score for all vertices.
4. Update the vertices synchronously by adding and deleting edges as selected by the strategy vectors. With probability p_r an edge is added to a random vertex instead of a neighbor's neighbor.
5. Every t_{strat}'th timestep, update the strategy vectors. For each vertex, with probability p_s, swap two elements in a vertex' strategy vector.
6. Increment the simulation time t and, if $t < t_{\mathrm{tot}}$, go to step 3.

global flow of the simulation

1. Initialize the network to an Erdős-Rényi network with N vertices and M_0 edges.
2. Use random permutations of the six actions as $\mathbf{s}_{\mathrm{add}}$ and $\mathbf{s}_{\mathrm{del}}$ for all vertices.
3. Calculate the score for all vertices.
4. Update the vertices synchronously by adding and deleting edges as selected by the strategy vectors. With probability p_r an edge is added to a random vertex instead of a neighbor's neighbor.
5. Every t_{strat}'th timestep, update the strategy vectors. For each vertex, with probability p_s, swap two elements in a vertex' strategy vector.
6. Increment the simulation time t and, if $t < t_{\mathrm{tot}}$, go to step 3.

global flow of the simulation

1. Initialize the network to an Erdős-Rényi network with N vertices and M_0 edges.
2. Use random permutations of the six actions as $\mathbf{s}_{\mathrm{add}}$ and $\mathbf{s}_{\mathrm{del}}$ for all vertices.
3. Calculate the score for all vertices.
4. Update the vertices synchronously by adding and deleting edges as selected by the strategy vectors. With probability p_r an edge is added to a random vertex instead of a neighbor's neighbor.
5. Every t_{strat}'th timestep, update the strategy vectors. For each vertex, with probability p_s, swap two elements in a vertex' strategy vector.
6. Increment the simulation time t and, if $t < t_{\mathrm{tot}}$, go to step 3.

global flow of the simulation

1. Initialize the network to an Erdős-Rényi network with N vertices and M_0 edges.
2. Use random permutations of the six actions as $\mathbf{s}_{\mathrm{add}}$ and $\mathbf{s}_{\mathrm{del}}$ for all vertices.
3. Calculate the score for all vertices.
4. Update the vertices synchronously by adding and deleting edges as selected by the strategy vectors. With probability p_r an edge is added to a random vertex instead of a neighbor's neighbor.
5. Every t_{strat}'th timestep, update the strategy vectors. For each vertex, with probability p_s, swap two elements in a vertex' strategy vector.
6. Increment the simulation time t and, if $t < t_{\mathrm{tot}}$, go to step 3.

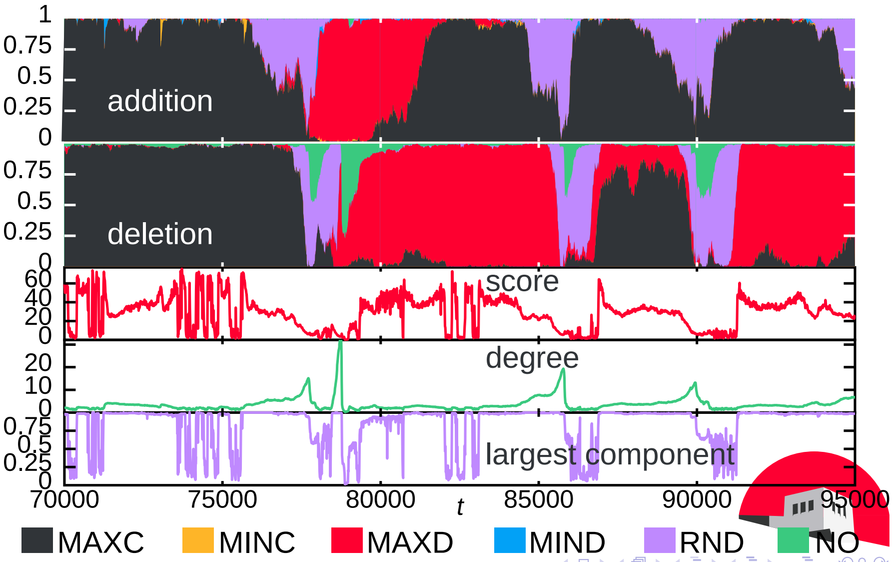
global flow of the simulation

1. Initialize the network to an Erdős-Rényi network with N vertices and M_0 edges.
2. Use random permutations of the six actions as $\mathbf{s}_{\mathrm{add}}$ and $\mathbf{s}_{\mathrm{del}}$ for all vertices.
3. Calculate the score for all vertices.
4. Update the vertices synchronously by adding and deleting edges as selected by the strategy vectors. With probability p_r an edge is added to a random vertex instead of a neighbor's neighbor.
5. Every t_{strat}'th timestep, update the strategy vectors. For each vertex, with probability p_s, swap two elements in a vertex' strategy vector.
6. Increment the simulation time t and, if $t < t_{\mathrm{tot}}$, go to step 3.

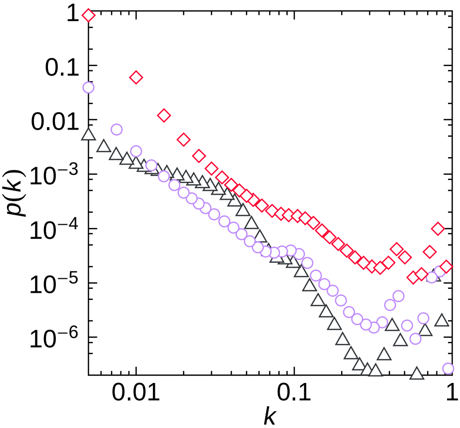
global flow of the simulation

1. Initialize the network to an Erdős-Rényi network with N vertices and M_0 edges.
2. Use random permutations of the six actions as $\mathbf{s}_{\mathrm{add}}$ and $\mathbf{s}_{\mathrm{del}}$ for all vertices.
3. Calculate the score for all vertices.
4. Update the vertices synchronously by adding and deleting edges as selected by the strategy vectors. With probability p_r an edge is added to a random vertex instead of a neighbor's neighbor.
5. Every t_{strat}'th timestep, update the strategy vectors. For each vertex, with probability p_s, swap two elements in a vertex' strategy vector.
6. Increment the simulation time t and, if $t < t_{\mathrm{tot}}$, go to step 3.

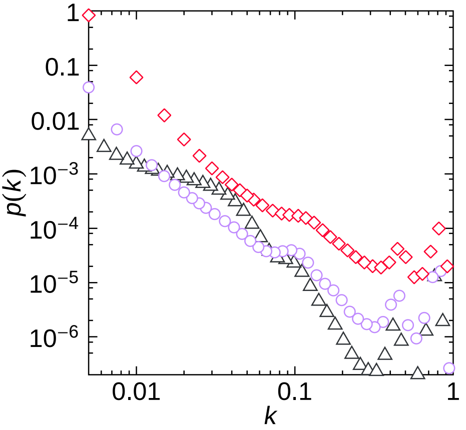
time evolution

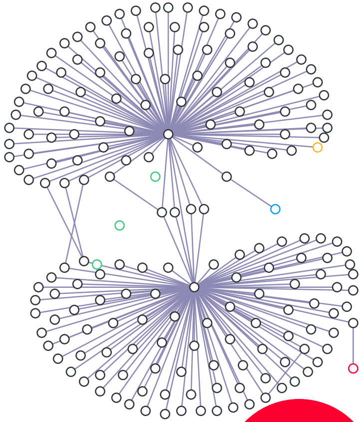


MAXC dominated network structure: degree distribution

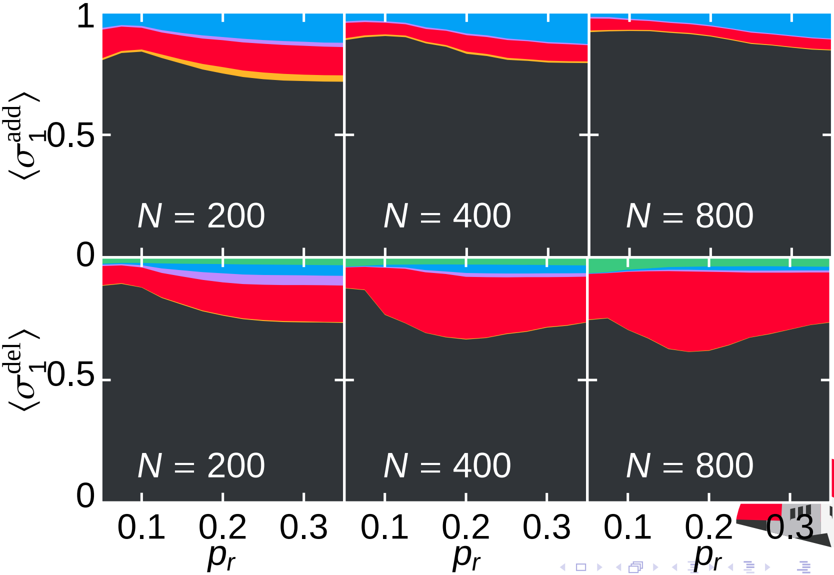


MAXC dominated network structure: example

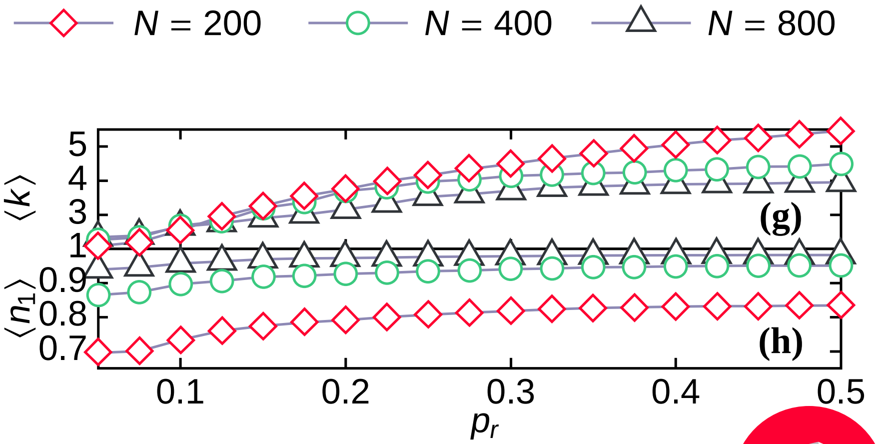




effect of random moves: histograms



effect of random moves: degree & cluster size



conclusions

- A simple problem that gets quite convoluted when one wants to be general.
- Complex time evolution with spikes, quasi-equilibria and trends.
- Network structure and strategy densities are correlated.
- The most common strategy, over a large range of parameter space, is MAXC.
- MAXC gives a bimodal degree distribution
- The NO/NO strategy is not stable—Red Queen.
- The network gets sparser and more connected with size.

conclusions

- A simple problem that gets quite convoluted when one wants to be general.
- Complex time evolution with spikes, quasi-equilibria and trends.
- Network structure and strategy densities are correlated.
- The most common strategy, over a large range of parameter space, is MAXC.
- MAXC gives a bimodal degree distribution
- The NO/NO strategy is not stable—Red Queen.
- The network gets sparser and more connected with size.

conclusions

- A simple problem that gets quite convoluted when one wants to be general.
- Complex time evolution with spikes, quasi-equilibria and trends.
- Network structure and strategy densities are correlated.
- The most common strategy, over a large range of parameter space, is MAXC.
- MAXC gives a bimodal degree distribution
- The NO/NO strategy is not stable—Red Queen.
- The network gets sparser and more connected with size.

conclusions

- A simple problem that gets quite convoluted when one wants to be general.
- Complex time evolution with spikes, quasi-equilibria and trends.
- Network structure and strategy densities are correlated.
- The most common strategy, over a large range of parameter space, is MAXC.
- MAXC gives a bimodal degree distribution
- The NO/NO strategy is not stable—Red Queen.
- The network gets sparser and more connected with size.

conclusions

- A simple problem that gets quite convoluted when one wants to be general.
- Complex time evolution with spikes, quasi-equilibria and trends.
- Network structure and strategy densities are correlated.
- The most common strategy, over a large range of parameter space, is MAXC.
- MAXC gives a bimodal degree distribution
- The NO/NO strategy is not stable—Red Queen.
- The network gets sparser and more connected with size.

conclusions

- A simple problem that gets quite convoluted when one wants to be general.
- Complex time evolution with spikes, quasi-equilibria and trends.
- Network structure and strategy densities are correlated.
- The most common strategy, over a large range of parameter space, is MAXC.
- MAXC gives a bimodal degree distribution
- The NO/NO strategy is not stable—Red Queen.
- The network gets sparser and more connected with size.

conclusions

- A simple problem that gets quite convoluted when one wants to be general.
- Complex time evolution with spikes, quasi-equilibria and trends.
- Network structure and strategy densities are correlated.
- The most common strategy, over a large range of parameter space, is MAXC.
- MAXC gives a bimodal degree distribution
- The NO/NO strategy is not stable—Red Queen.
- The network gets sparser and more connected with size.

conclusions

- A simple problem that gets quite convoluted when one wants to be general.
- Complex time evolution with spikes, quasi-equilibria and trends.
- Network structure and strategy densities are correlated.
- The most common strategy, over a large range of parameter space, is MAXC.
- MAXC gives a bimodal degree distribution
- The NO/NO strategy is not stable—Red Queen.
- The network gets sparser and more connected with size.

COEVOLUTION OF NETWORKS & OPINIONS: the idea

P. Holme & M. E. J. Newman, e-print physics/0603023

- Opinions spread over social networks.
- People with the same opinion are likely to become acquainted.
- We try to combine these points into a simple model of simultaneous opinion spreading and network evolution.

COEVOLUTION OF NETWORKS & OPINIONS: the idea

P. Holme & M. E. J. Newman, e-print physics/0603023

- Opinions spread over social networks.
- People with the same opinion are likely to become acquainted.
- We try to combine these points into a simple model of simultaneous opinion spreading and network evolution.

COEVOLUTION OF NETWORKS & OPINIONS: the idea

P. Holme & M. E. J. Newman, e-print physics/0603023

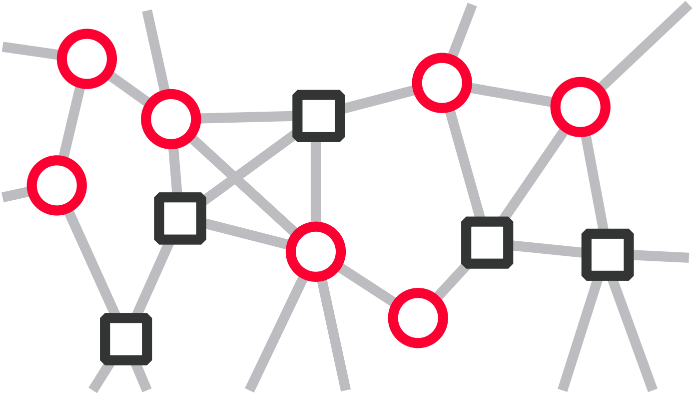
- Opinions spread over social networks.
- People with the same opinion are likely to become acquainted.
- We try to combine these points into a simple model of simultaneous opinion spreading and network evolution.

COEVOLUTION OF NETWORKS & OPINIONS: the idea

P. Holme & M. E. J. Newman, e-print physics/0603023

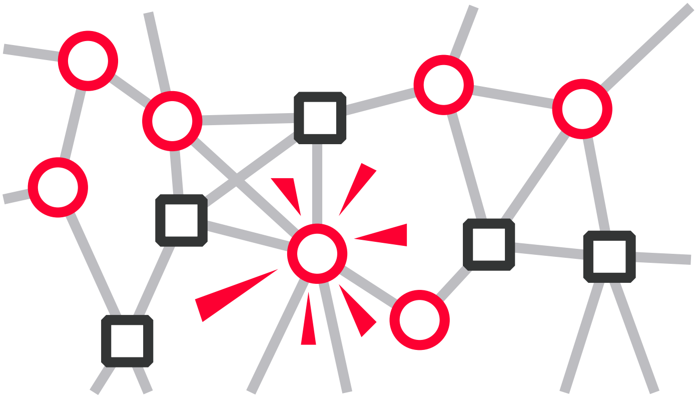
- Opinions spread over social networks.
- People with the same opinion are likely to become acquainted.
- We try to combine these points into a simple model of simultaneous opinion spreading and network evolution.

the voter model



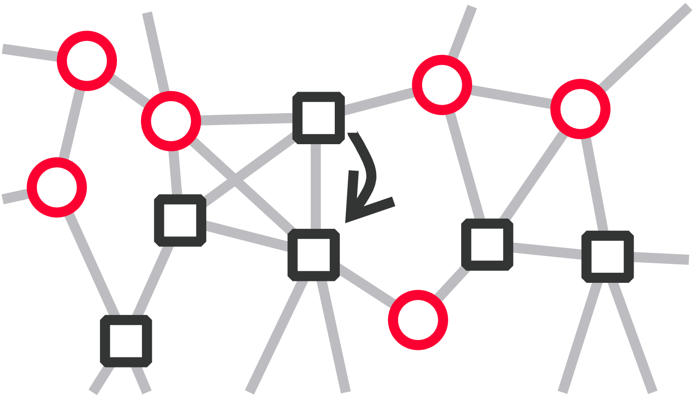
Clifford & Sudbury, Biometrika **60**, 581 (1973).
Holley & Liggett, Ann. Probab. **3**, 643 (1975).

the voter model



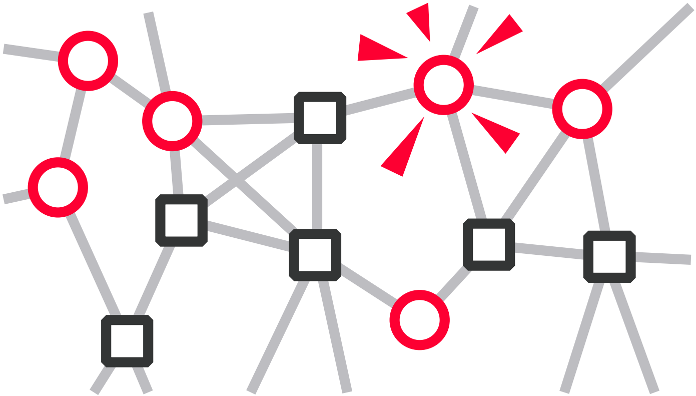
choose one vertex randomly

the voter model



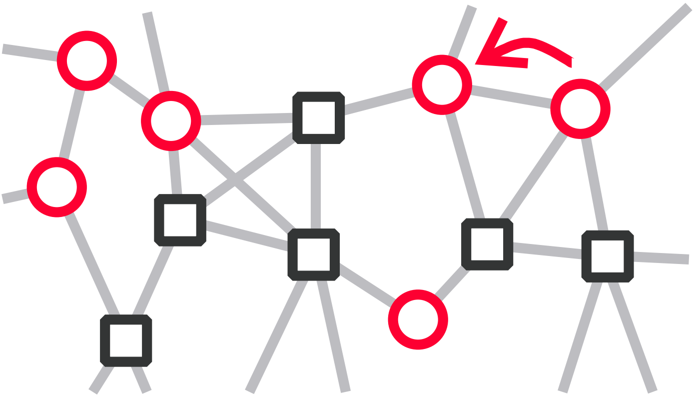
copy the opinion of a random neighbor

the voter model



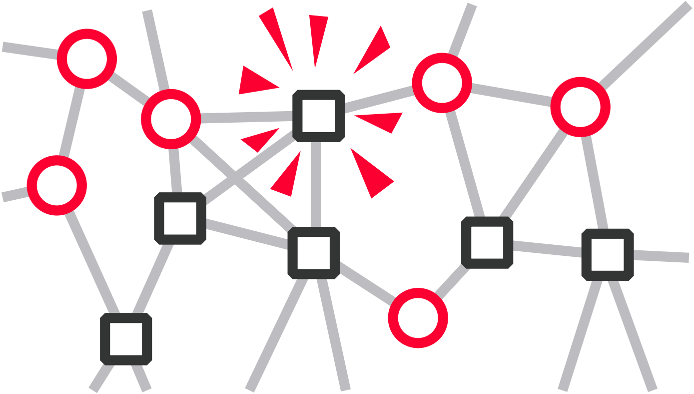
and so on . . .

the voter model



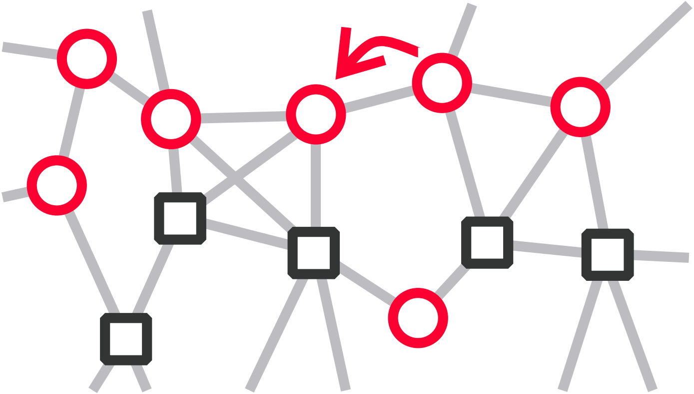
and so on . . .

the voter model



and so on . . .

the voter model



and so on . . .

the voter model

acquaintance dynamics

- People of similar interests are likely to get acquainted. e.g.: McPherson *et al.*, Ann. Rev. Sociol. **27**, 415 (2001).
- The number of edges is constant.

the voter model

acquaintance dynamics

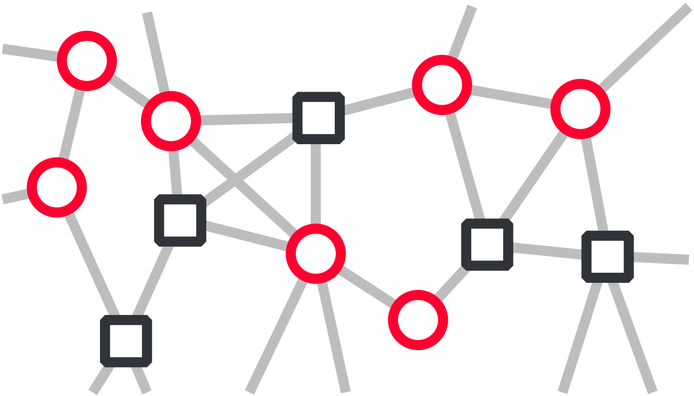
- People of similar interests are likely to get acquainted. e.g.: McPherson *et al.*, Ann. Rev. Sociol. **27**, 415 (2001).
- The number of edges is constant.

the voter model

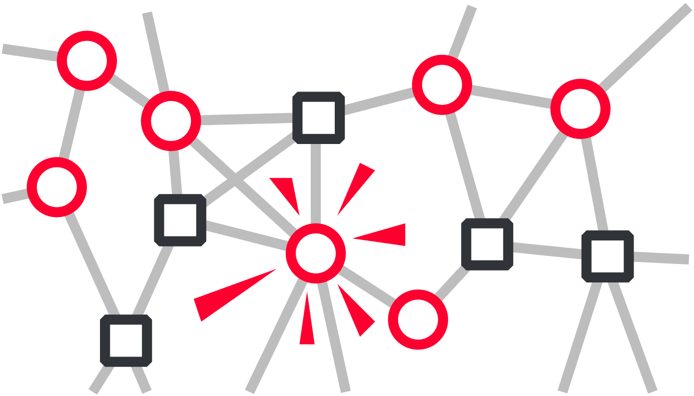
acquaintance dynamics

- People of similar interests are likely to get acquainted. e.g.: McPherson *et al.*, Ann. Rev. Sociol. **27**, 415 (2001).
- The number of edges is constant.

acquaintance dynamics

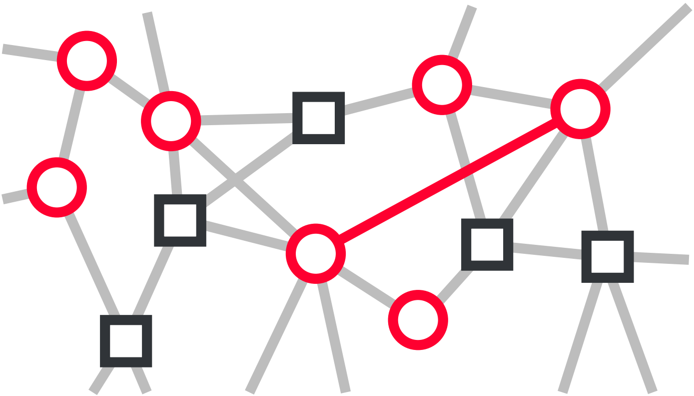


acquaintance dynamics



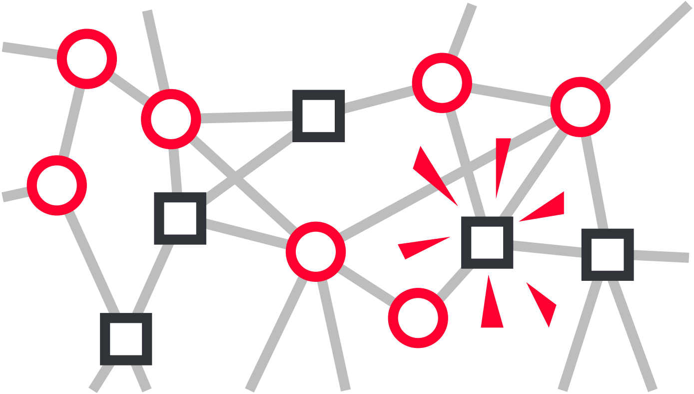
choose one vertex randomly

acquaintance dynamics



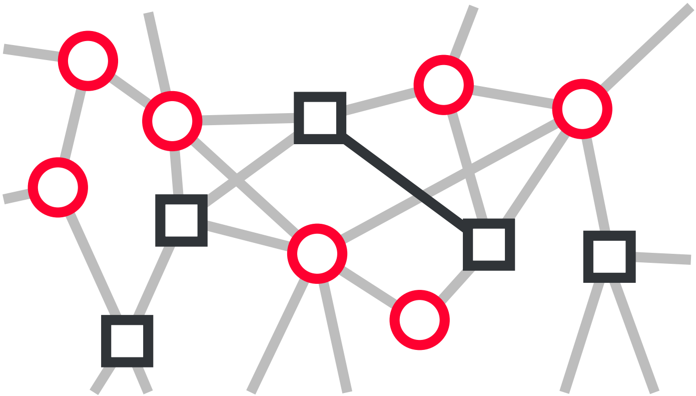
rewire an edge to a vertex with the same opinion

acquaintance dynamics



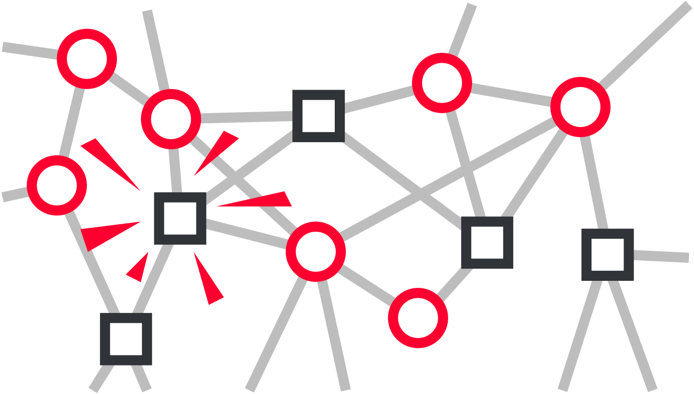
and so on . . .

acquaintance dynamics



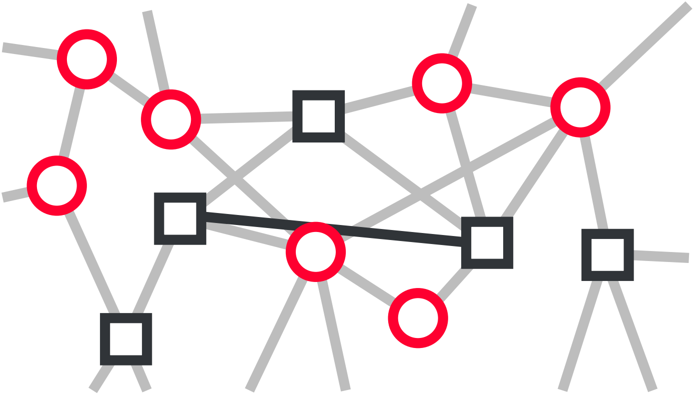
and so on . . .

acquaintance dynamics



and so on . . .

acquaintance dynamics



and so on . . .

our model

1. Start with a random network of N vertices $M = \bar{k}N/2$ edges and $G = N/\gamma$ randomly assigned opinions.
2. Pick a vertex i at random.
3. With a probability ϕ make a voter model step from i . . .
4. . . . otherwise make an acquaintance formation step from i.
5. If there are edges leading between vertices of different opinions—iterate from step 2.

our model

1. Start with a random network of N vertices $M = \bar{k}N/2$ edges and $G = N/\gamma$ randomly assigned opinions.
2. Pick a vertex i at random.
3. With a probability ϕ make a voter model step from i . . .
4. . . . otherwise make an acquaintance formation step from i.
5. If there are edges leading between vertices of different opinions—iterate from step 2.

our model

1. Start with a random network of N vertices $M = \bar{k}N/2$ edges and $G = N/\gamma$ randomly assigned opinions.
2. Pick a vertex i at random.
3. With a probability ϕ make a voter model step from i . . .
4. . . . otherwise make an acquaintance formation step from i.
5. If there are edges leading between vertices of different opinions—iterate from step 2.

our model

1. Start with a random network of N vertices $M = \bar{k}N/2$ edges and $G = N/\gamma$ randomly assigned opinions.
2. Pick a vertex i at random.
3. With a probability ϕ make a voter model step from i . . .
4. . . . otherwise make an acquaintance formation step from i.
5. If there are edges leading between vertices of different opinions—iterate from step 2.

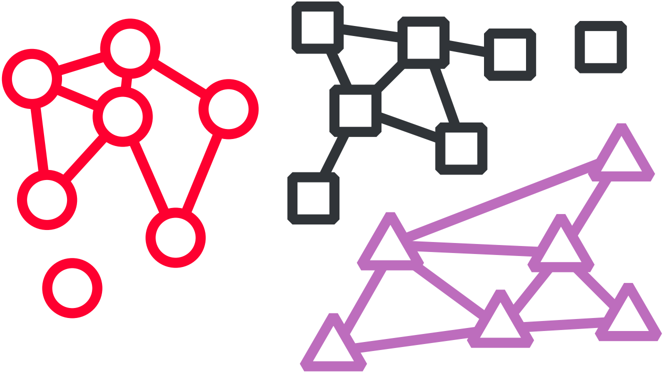
our model

1. Start with a random network of N vertices $M = \bar{k}N/2$ edges and $G = N/\gamma$ randomly assigned opinions.
2. Pick a vertex i at random.
3. With a probability ϕ make a voter model step from i . . .
4. . . . otherwise make an acquaintance formation step from i.
5. If there are edges leading between vertices of different opinions—iterate from step 2.

our model

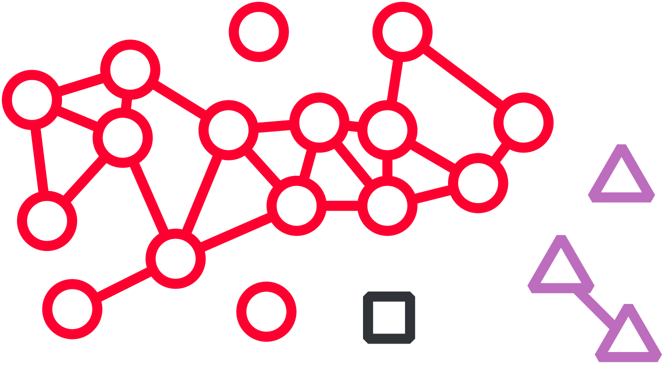
1. Start with a random network of N vertices $M = \bar{k}N/2$ edges and $G = N/\gamma$ randomly assigned opinions.
2. Pick a vertex i at random.
3. With a probability ϕ make a voter model step from i . . .
4. . . . otherwise make an acquaintance formation step from i.
5. If there are edges leading between vertices of different opinions—iterate from step 2.

phases



low ϕ—clusters of similar sizes

phases



high ϕ—one dominant cluster

quantities we measure

- The relative largest size S of a cluster (of vertices with the same opinion).
- The average time τ to reach consensus.

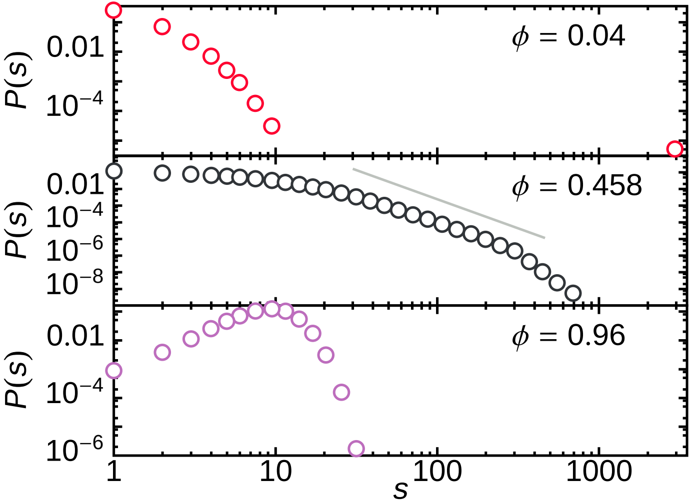
quantities we measure

- The relative largest size S of a cluster (of vertices with the same opinion).
- The average time τ to reach consensus.

quantities we measure

- The relative largest size S of a cluster (of vertices with the same opinion).
- The average time τ to reach consensus.

cluster size distribution



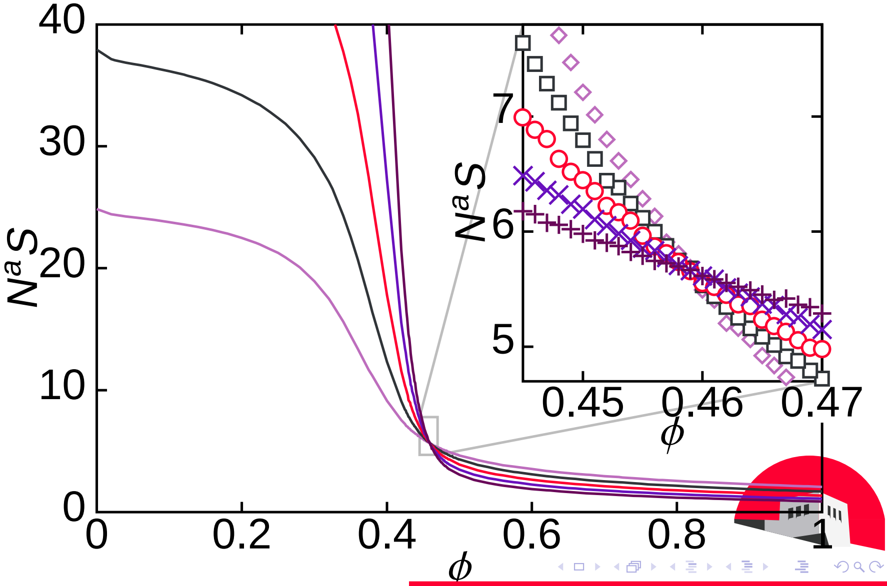
finding the phase transition

Assume a critical scaling form:

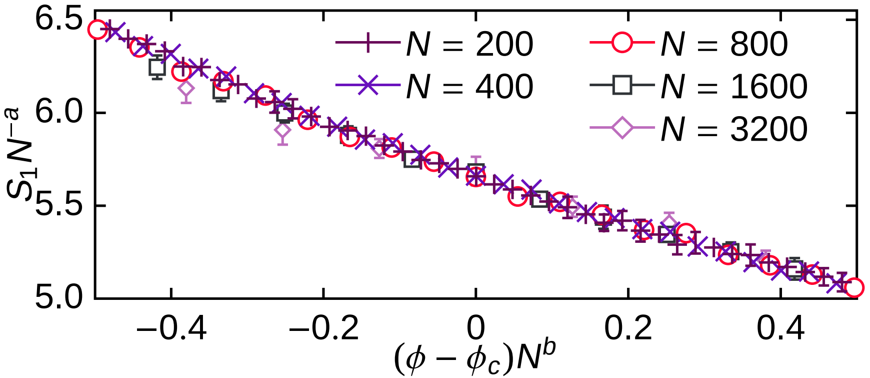
scaling form

$$S = N^{-a}\, F\Big(N^{b}(\phi - \phi_c)\Big)$$

finding the phase transition



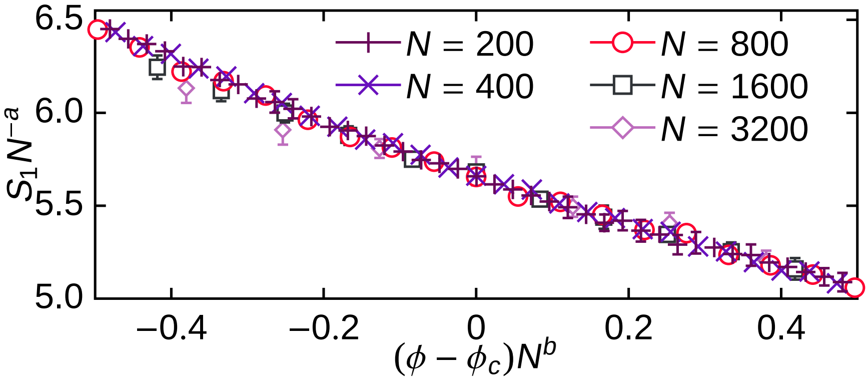
finding the phase transition



$a = 0.61 \pm 0.05$, $\phi_c = 0.458 \pm 0.008$, $b = 0.7 \pm 0.1$

random graph percolation: $a = b = 1/3$

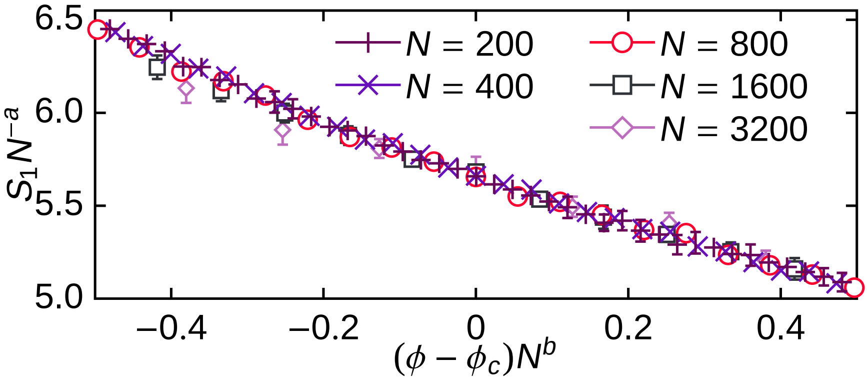
finding the phase transition



$a = 0.61 \pm 0.05$, $\phi_c = 0.458 \pm 0.008$, $b = 0.7 \pm 0.1$

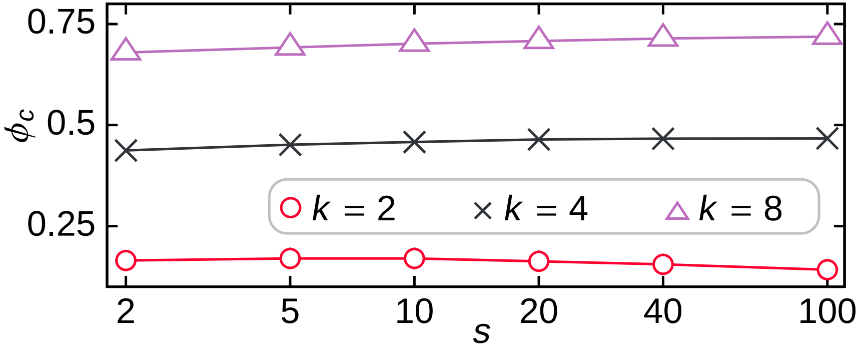
random graph percolation: $a = b = 1/3$

finding the phase transition

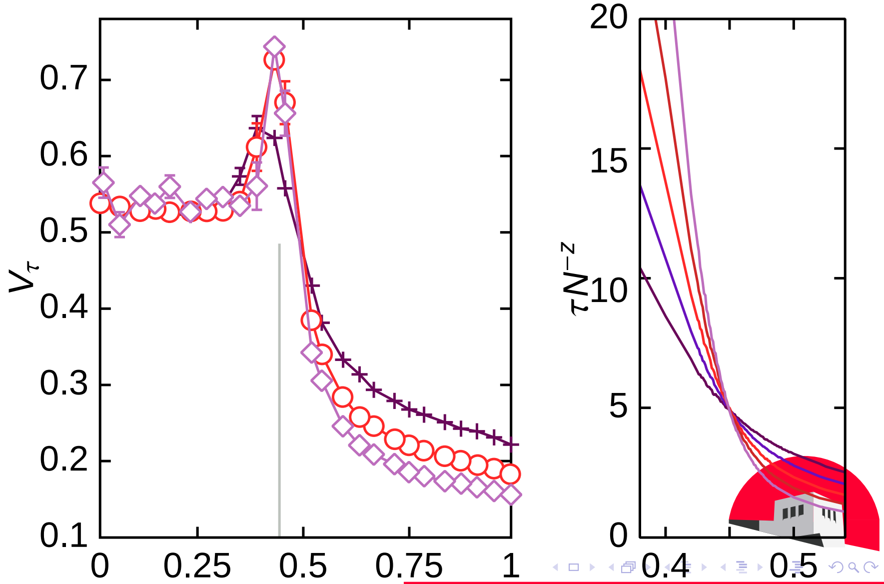


$a = 0.61 \pm 0.05$, $\phi_c = 0.458 \pm 0.008$, $b = 0.7 \pm 0.1$
random graph percolation: $a = b = 1/3$

phase diagram



dynamic critical behavior



conclusions

- We have proposed a simple, non-equilibrium model for the coevolution of networks and opinions.
- The model undergoes a second order phase transition between: One state of clusters of similar sizes. One state with one dominant cluster.
- The universality class is not the same as random graph percolation.
- In society, a tiny change in the social dynamics may cause a large change in the diversity of opinions.

conclusions

- We have proposed a simple, non-equilibrium model for the coevolution of networks and opinions.
- The model undergoes a second order phase transition between: One state of clusters of similar sizes. One state with one dominant cluster.
- The universality class is not the same as random graph percolation.
- In society, a tiny change in the social dynamics may cause a large change in the diversity of opinions.

conclusions

- We have proposed a simple, non-equilibrium model for the coevolution of networks and opinions.
- The model undergoes a second order phase transition between: One state of clusters of similar sizes. One state with one dominant cluster.
- The universality class is not the same as random graph percolation.
- In society, a tiny change in the social dynamics may cause a large change in the diversity of opinions.

conclusions

- We have proposed a simple, non-equilibrium model for the coevolution of networks and opinions.
- The model undergoes a second order phase transition between: One state of clusters of similar sizes. One state with one dominant cluster.
- The universality class is not the same as random graph percolation.
- In society, a tiny change in the social dynamics may cause a large change in the diversity of opinions.

conclusions

- We have proposed a simple, non-equilibrium model for the coevolution of networks and opinions.
- The model undergoes a second order phase transition between: One state of clusters of similar sizes. One state with one dominant cluster.
- The universality class is not the same as random graph percolation.
- In society, a tiny change in the social dynamics may cause a large change in the diversity of opinions.