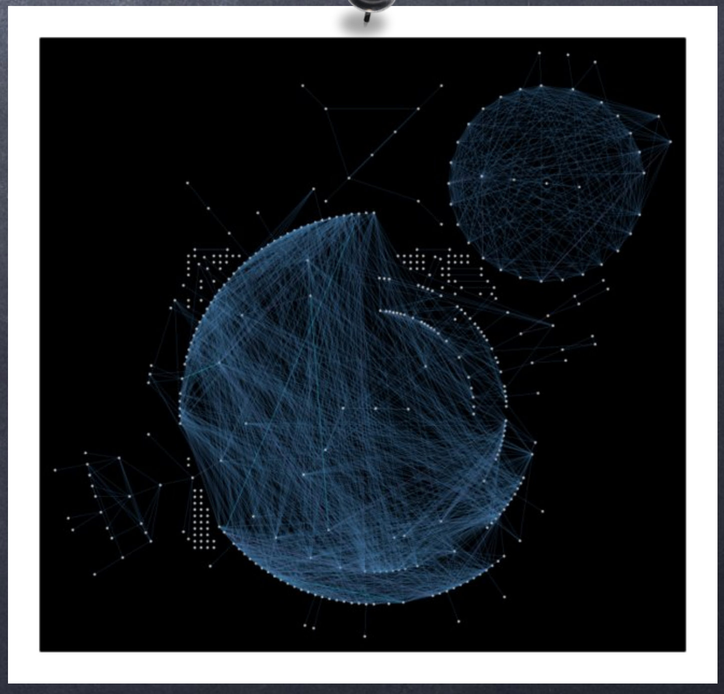
Random graph models for analysing real networks

blerina sinaimeri



A Simple Random Graph Model
Erdös-Renyi model

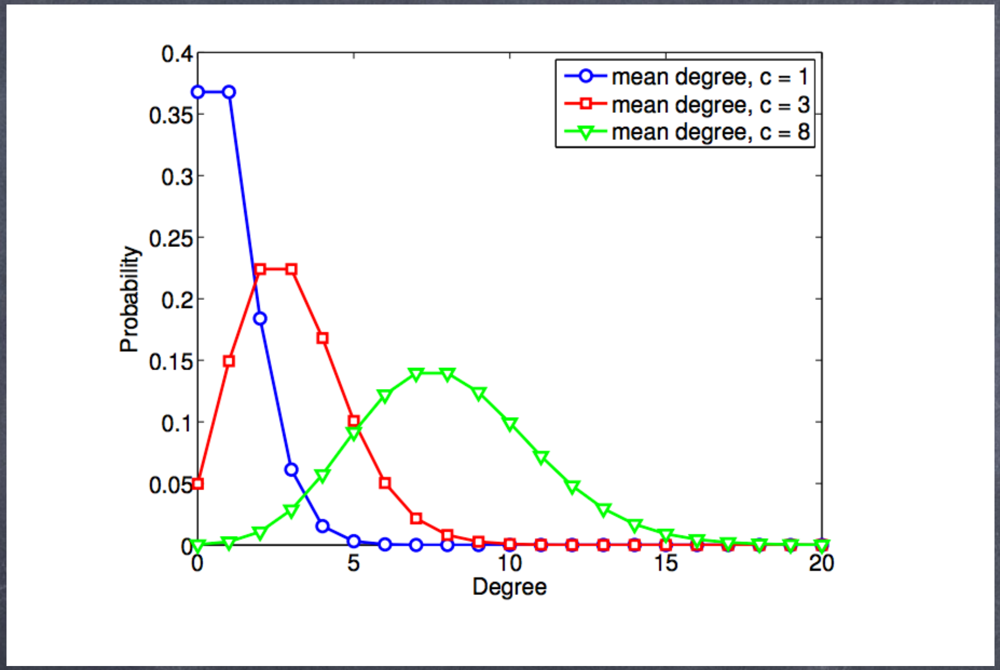
- The Erdös-Renyi model is denoted $G(n,p)$ and is one of the simplest random models.
- All graphs on n vertices.
- Every edge is formed with probability $p \in (0, 1)$ independently of every other edge.

A Simple Random Graph Model
Erdös-Renyi model

- Given a vertex v what is the probability v has degree d?
  - $P[deg(v)=d] = binom(n-1,d)p^{d}(1-p)^{n-1-d}$
- What is the expected mean degree of a vertex?
  - $E[d] = np$
- We can use a poisson approximation to compute an expected degree distribution of G(n, p) for sparse graphs.

Degree distribution in the Erdös-Renyi model

- The poison distribution has mean and variance c.



- Many realistic networks show a heavy-tail distribution.(A distribution with a "tail" that is "heavier" than an exponential)

Individual clustering coefficient

- The clustering coefficient measures the degree of clustering of a typical node's neighborhood. It is defined as the likelihood that any two nodes with a common neighbor are themselves connected. The individual clustering coefficient for a vertex v is given by:

$$CC(G,v)=\frac{\#\text{triangles that contain } v}{(1/2)d(v)[d(v)-1]}$$

Claim: The expectation of the individual clustering clustering coefficient in $G(n, p)$ is p.

Individual clustering coefficient

	n	average degree	cc	cc in G(n,p)
Actors network	225226	61	0.79	0.00027
power grid	4941	2.67	0.08	0,005
C. elegans	282	14	0.28	0.05

Example from paper Watts and Strogatz Nature, 1998

Configuration models

- A particularly unrealistic aspect of the Erdös-Renyi model $G(n, p)$ is its degree distribution, which we showed follows a Poisson distribution when the graph is sparse. In contrast, most real-world graphs exhibit heavy-tailed degree distributions.
- We can improve this aspect of our random graph model by using a generalization called the configuration model. We can define the random graph models based on the distribution of their degrees.
  - $G(n,\underline{k})$ where $\underline{k}=(k_1,\ldots k_n)$ is a degree sequence.
  - $\underline{k}$ can be any sequence. Can be fixed or a sequence of values drawn i.i.d. from some degree distribution $Pr(k)$. If $\underline{k}$ ~ Poisson(c/n) then the model produces something very near to the Erdös-Renyi model.

Configuration models

- Fixed degrees: Given a degree sequence $\underline{k}=(k_1,\ldots k_n)$ generate a graph uniformly at random from the set of graphs on n vertices having exactly $\underline{k}$ as a degree sequence.

- Is it always possible?

Configuration models

- Fixed degrees: Given a degree sequence $\underline{k}=(k_1,\ldots k_n)$ generate a graph uniformly at random from the set of graphs on n vertices having exactly $\underline{k}$ as a degree sequence.
- Is it always possible?
    - (5,3,1,1,1) ?
    - (5,3,1,1,1,2)

Degree sequences

Theorem [Erdös-Gallai]

A non negative sequence of integers $d_1 \geqq d_2 \geqq \ldots \geqq d_n$ is graphical i.e. can be represented as the degree sequence of a finite simple graph on n vertices if and only if $d_1 + d_2 + \ldots + d_n$ is even and for every $1 \leqq k \leqq n$ it holds:

$$\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min(d_i, k)$$

Havel-Hakimi algorithm

A non negative sequence of integers $d_1 \geqq d_2 \geqq \ldots \geqq d_n$ is

graphical on n vertices if and only if the sequence $d_2 - 1 \geqq d_3$ $-1 \geqq \ldots \geqq d_{d1+1} - 1 \geqq d_{d1+2} \geqq \ldots \geqq d_n$ is graphical.

Configuration models

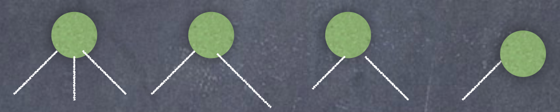
- Fixed degrees: Given a degree sequence $\underline{k}=(k_1,\ldots k_n)$ generate a graph uniformly at random from the set of graphs on n vertices having exactly $\underline{k}$ as a degree sequence.
  - Matching algorithm
  - Switching algorithm

Matching algorithm

```
//INPUT: d=(d_1, ... , d_n)
//OUTPUT: list of edges
//Initialization
Edge.List<-();
Node.List<-();
//Create fake Node.List:
For i in {1, ... , n} do
      While d_i >= 1 do
        Node.list <- concatenate(Node.list,i)
        d_i <- d_i -1
      Endwhile
EndFor
//Create Edge.List
while Node.List is not empty do
  Choose randomly i, j in Node.List without replacement
  Edge.List <- concatenate(Edge.List, {i,j})
End while
```

If Edge.List contains loops or multipledges repeat.

Idea



Matching algorithm

```
//INPUT: d=(d_1, … , d_n)
//OUTPUT: list of edges
//Initialization
Edge.List<-();
Node.List<-();
//Create fake Node.List:
For i in {1, … , n} do
      While d_i >= 1 do
        Node.list <- concatenate(Node.list,i)
        d_i <- d_i -1
      Endwhile
EndFor
//Create Edge.List
while Node.List is not empty do
  Choose randomly i, j in Node.List without replacement
  Edge.List <- concatenate(Edge.List, {i,j})
End while
```

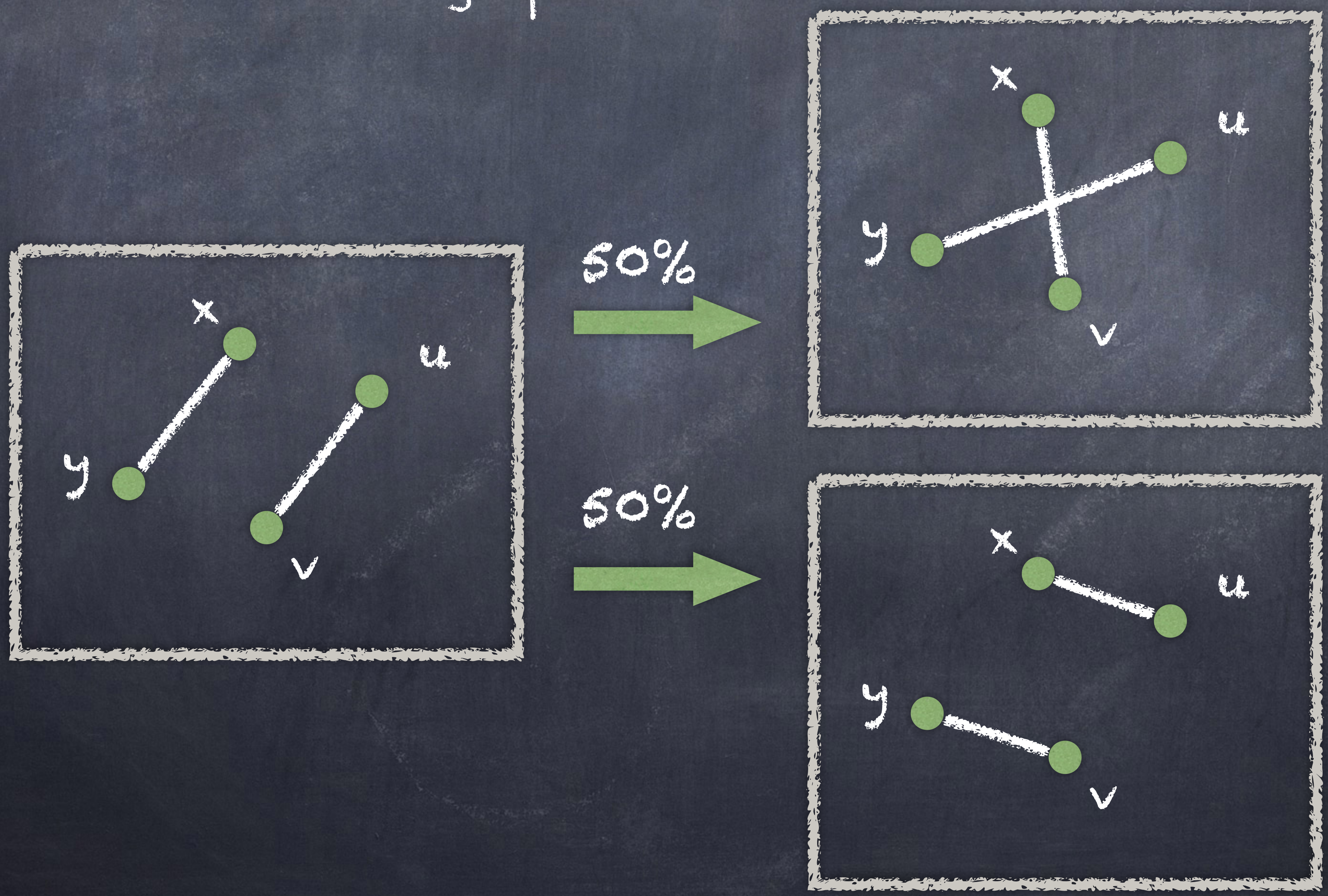
If Edge.List contains loops or multipledges repeat.

Problem:

- May introduce graphs with loops and double edges, and if we have a graph with higher vertex degrees, we may fail to come up with a simple generalized random graph within a reasonable amount of trials.

Switching algorithm

Idea: This method starts by considering any graph which satisfies the required degree distribution (i.e. node i has degree d_i and change it by performing a long series of random edge crosses, until it becomes a generalized random graph.



Switching algorithm

Idea: This method starts by considering any graph which satisfies the required degree distribution (i.e. node i has degree d_i and change it by performing a long series of random edge crosses, until it becomes a generalized random graph.

```
//INPUT: Edge.List satisfying d=(d1, ... , dn) and Nr_iterations
//OUTPUT: Edge.List of the random graph
While Nr_iterations >= 1 do
      Choose e1= {x,y} and e2={u,v} uniformly at random inside Edge.List
      //Cross the edges randomly for example {x,u} and {y,v}
      Switch(e1, e2)
      If no loop or multiple edge is created then replace e1, e2 with the new edges.
      Nr_iterations <- Nr_iterations - 1
EndWhile
```

Empirically Nr_Iterations=100