Complex Networks

Shoichi Yip

M2 PCS

3 April 2024

Complex networks

Networks are often the backbone of many complex systems.

Biological systems, citations in papers, social networks can be all modeled as complex networks.

The modeling of complex networks has stem from the study of models of random graphs, starting with the pioneering work by Paul Erdős and Alfréd Rényi in 1959.

Random graphs

The problem with ER graphs

Graphs generated using the ER model have a Poisson degree distribution.

However, experimental study of real world graphs show us that most of the times they do not follow a Poissonian behaviour.

We can overcome this problem by using the configuration model.

The configuration model

The configuration model allows us to generate graphs exactly with a given degree sequence $\vec{k}$.

The degree sequence is a vector, for example,

$$\vec{k} = \begin{pmatrix} 3 & 2 & 3 & 1 & 1 \end{pmatrix}$$

where each i-th element is the degree of the i-th node.

Sampling a graph from a configuration model

The configuration model provides us with a constructive algorithm in order to find an instance of a random graph, provided a degree sequence $\vec{k}$.

For example, if we take the $\vec{k}$ degree sequence from the previous slide we can start with a set of unconnected nodes where each i-th node has k_i stubs, or half-edges.

Then we get a random graph instance by taking two random stubs at a time and connecting them. On general grounds, the configuration model allows for the presence of multiple edges and self-edges.

Sampling a graph from a configuration model

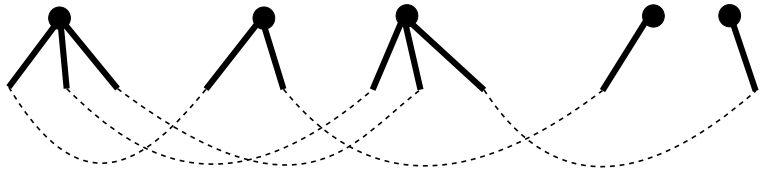

Figure: Connecting stubs in the configuration model

Configuration model given a degree distribution

We can adapt the configuration model to a specific task, that of sampling graphs that have a specific degree distribution.

We can in fact first of all define a degree sequence such that the abundance of nodes of degree d are given by a degree distribution p_d. Then we can proceed as we would do for the configuration model, by wiring the stubs.

We can hence extract random graph instances that nearly exactly match the degree distribution.

Our ensemble

In our particular case, we define a random graph ensemble $\mathcal{G}$ such that:

- the graph has N nodes;
- the graph is generated using the configuration model;
- the graph does not contain self-edges and multiple edges;
- we define a parameter π and the the graph is such that the fraction of the nodes $p_1 = 1 - \pi$ has degree 1, and the remaining fraction $p_4 = \pi$ has degree 4.

Algorithm to generate RG instance

Algorithm 1 Algorithm to generate stubs and connect them

for k_i in degree sequence $\vec{k}$ **do**
    Generate node with k_i stubs
end for
while Graph G not generated **do**
    while Stub list is not exhausted **do**
        Pick two random stubs
        Create a candidate edge between them
        Check whether they are not multiedge or self-edge
    end while
end while

Examples of random graphs

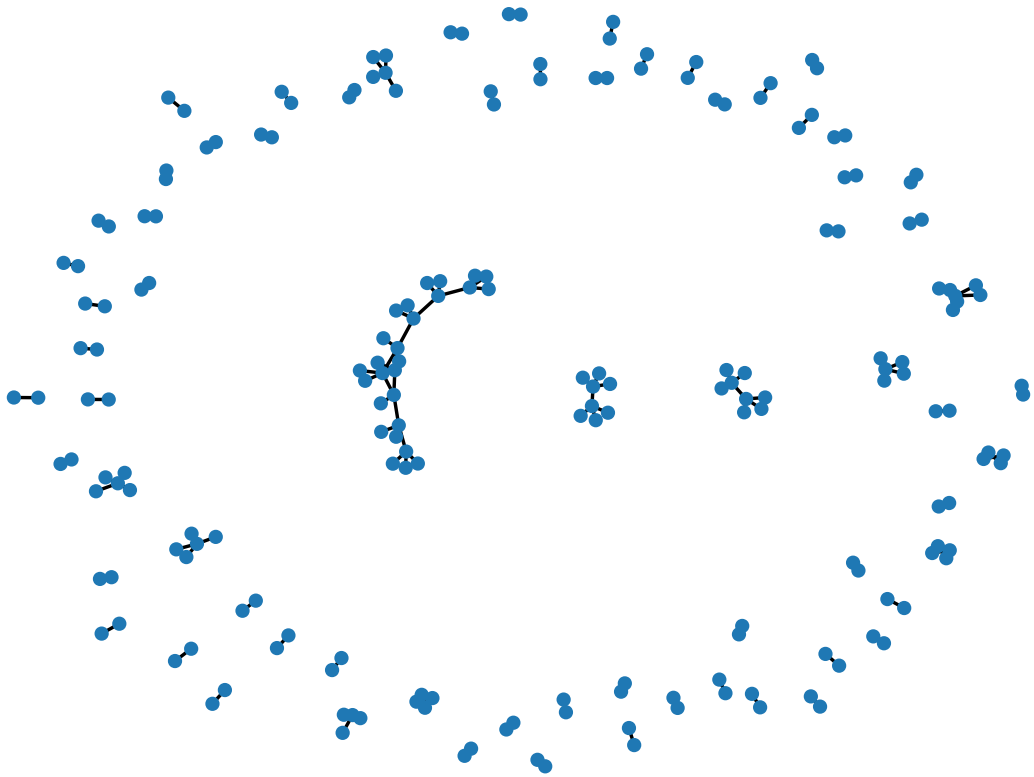

Figure: Instance of a random graph for $\pi = 0.1$

Examples of random graphs

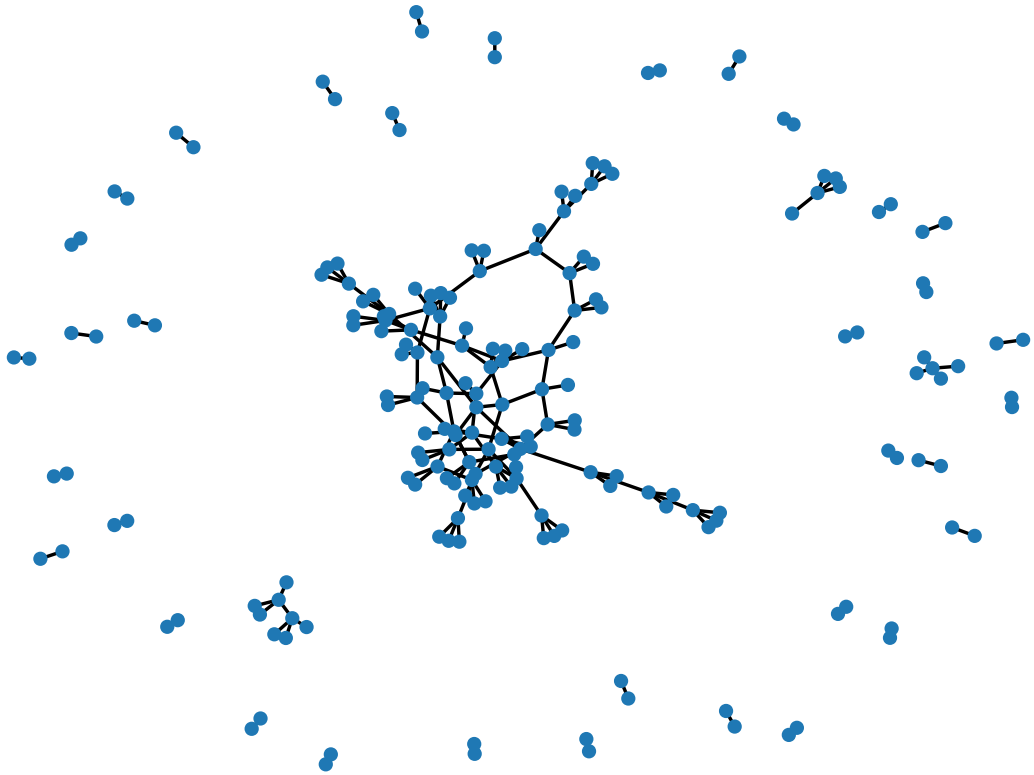

Figure: Instance of a random graph for $\pi = 0.3$

Examples of random graphs

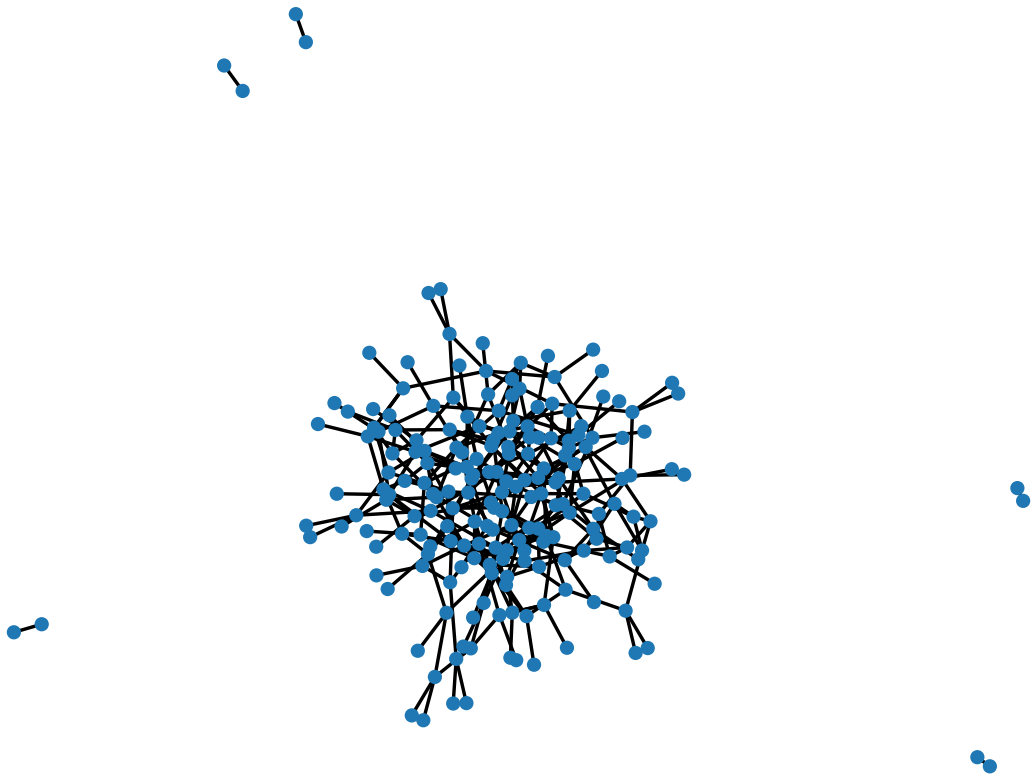

Figure: Instance of a random graph for $\pi = 0.7$

The giant component

Detecting the giant component in an instance

The onset of criticality

Size of the giant component

In order to find the size of the giant component we resort to the study of the probability that an excess node or a node belongs to the giant component [1, p. 56].

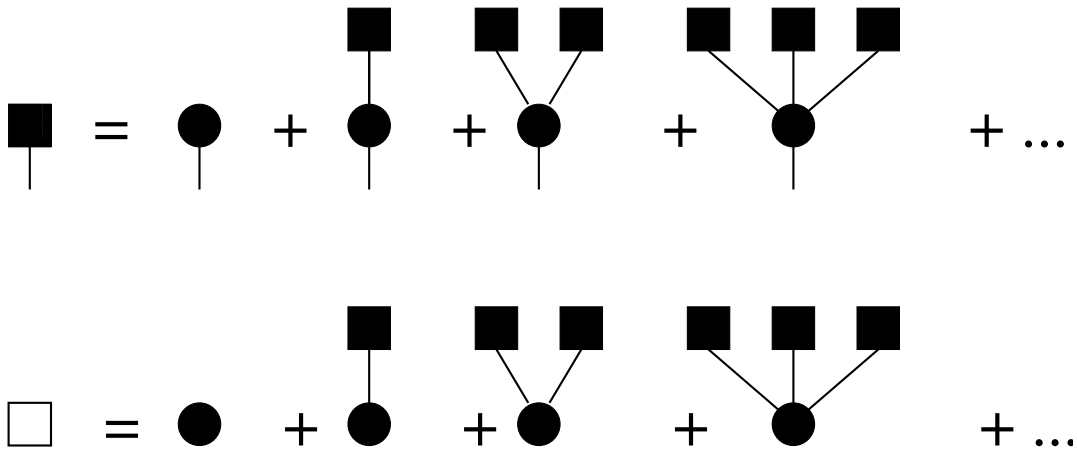

Figure: Scheme for the iterative solution to find the probability that a node belongs to the giant component and the excess probability

Excess probability that node belongs to GC

The scheme reads as follows. The excess proability μ, which is the probability that given a randomly selected edge one of its end vertices is not connected with a GC, is given by

- the probability that the excesss node has degree 1, i.e. there would not exist any other edge that connects it to a GC
- the probability that the excess node is connected to a node that does not belong to the GC
- the probability that the excess node is connected to two nodes that do not belong to the GC
- etc.

Probability that node belongs to GC

The same goes for the probability that a randomly selected node does not belong to the GC. The proability $1-\gamma$, is given by

- the probability that this node is isolated
- the probability that this node is connected to a node that does not belong to the GC
- the probability that this node is connected to two nodes that do not belong to the GC
- etc.

Iterative equations

We then find the iterative equations, respectively

$$\begin{cases} \mu = q_1 + q_2\mu + q_3\mu^2 + ... \\ 1 - \gamma = p_0 + p_1\mu + p_2\mu^2 + ... \end{cases} \tag{1}$$

Since our degree distribution has only two nonzero terms for $d = 1$ and $d = 4$, sd can rewrite them as

$$\begin{cases} \mu = q_1 + q_4\mu^3 \\ 1 - \gamma = p_1\mu + p_4\mu^4 \end{cases} \tag{2}$$

We can solve the first third order equation in Eq. 2 and we get

$$\mu = 1 \quad \vee \quad \mu = \frac{-1-\sqrt{\pi}}{2\sqrt{\pi}} \quad \vee \quad \mu = \frac{1-\sqrt{\pi}}{2\sqrt{\pi}}$$

where the first solution is the trivial solution that does not have any giant component, and the second solution is negative (so it cannot be a probability). If we plug the third solution in the second equation we get

$$\gamma = 1 - (1-\pi)\frac{1-\sqrt{\pi}}{2\sqrt{\pi}} - \pi\left(\frac{1-\sqrt{\pi}}{2\sqrt{\pi}}\right)^4 \tag{3}$$

which gives us the theoretical value of the probability of a node being part of the GC, hence also the expectation value over the size of the GC.

The size of the giant component

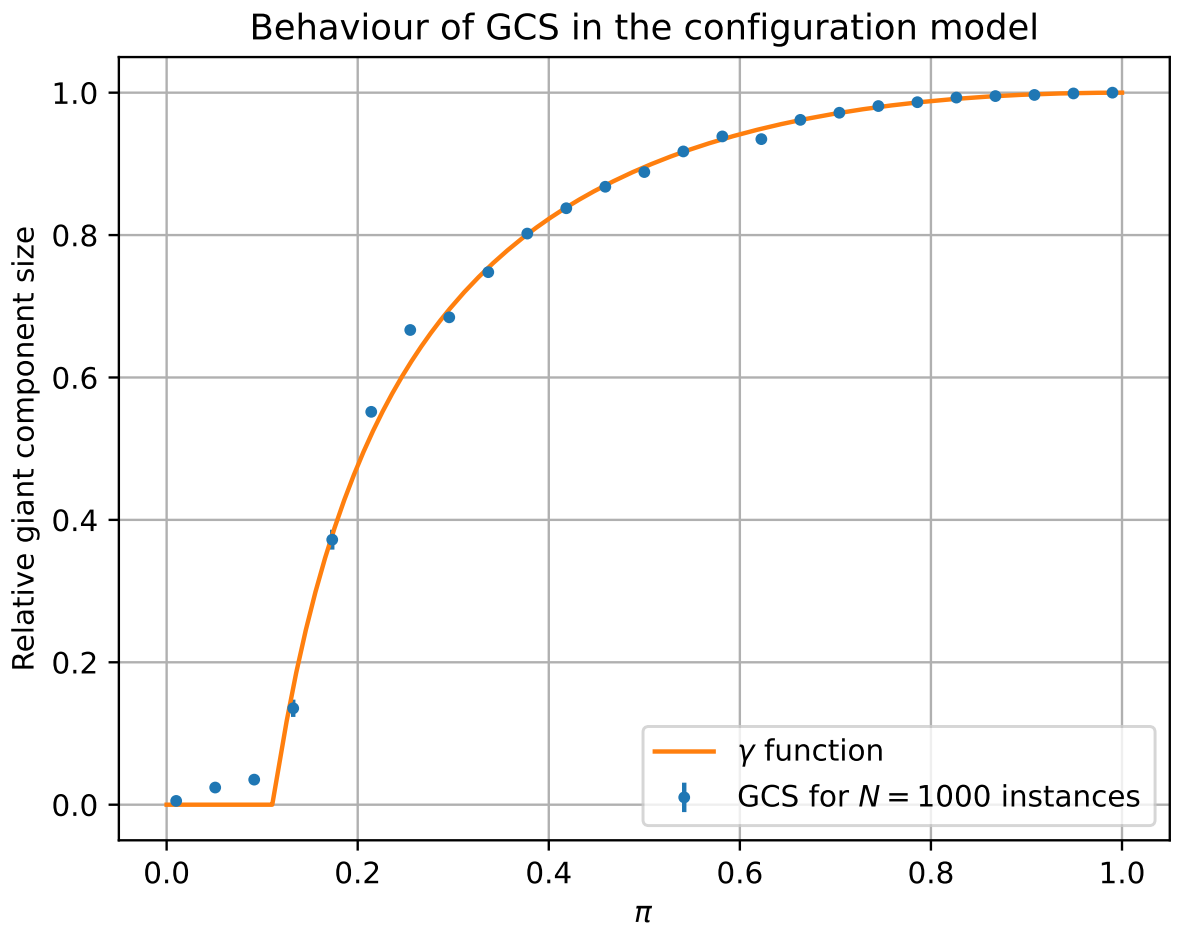

Figure: Comparison between theoretical value and measures from random instances of the size of the giant component

The deterministic q-core algorithm

The q-core of a graph can be found using a simple algorithm.

Algorithm 2 Algorithm for deterministic q-core detection

1: **while** Nodes with $d < q$ are present **do**
2: Remove all nodes with $d < q$
3: **end while**

We will use this algorithm to find the size of a q-core.

The stochastic q-core algorithm

There is a stochastic version of this algorithm.

Algorithm 3 Algorithm for stochastic q-core detection

```
while Nodes with d < q are present do
    Select randomly a node with degree d < q
    if The node has degree d < q then
        Remove node
    end if
end while
```

This algorithm does not remove an extensive amount of nodes for each timestep and its dynamic can be described by an ODE.

Rate equations for the q-core

The behaviour of q-cores in Algorithm 3 can be described by rate equations using the Wormwald method. [1, p. 58].

The algorithmic time $N(T) = N - T$ describes the amount of nodes that are still available, where N is the total number of nodes and T is the number of nodes that we have removed.

We can introduce a rescaled time $t = T/N$, which will span from 0 to 1.

The variable that we want to study in time is the degree distribution at time t

$$p_d(t) = \frac{N_d(t)}{N(t)} = \frac{N_d(t)}{N[1-t]} \tag{4}$$

ODE for the rate equation

We can write the difference between two consecutive timesteps of the number of nodes of degree d as

$$N_d(T+1) - N_d(T) = -\frac{\chi_d p_d}{\overline{\chi}_d} + \frac{\overline{\chi_d d}}{\overline{\chi}_d c(t)}[(d+1)p_{d+1}(t) - dp_d(t)] \quad (5)$$

where χ_d is the indicator function

$$\chi_d = \begin{cases} 1 & \text{if } d < q \\ 0 & \text{if } d \geq q \end{cases} \quad (6)$$

and $c(t)$ is the average connectivity at time t

$$c(t) = \sum_d d p_d(t) \quad (7)$$

ODE for the rate equation

Starting from Eq. 5 we can find the equations for $p_d(t)$

$$\partial_t[(1-t)p_d(t)] = -\frac{\chi_d p_d}{\overline{\chi}_d} + \frac{\overline{d\chi_d}}{c(t)\overline{\chi}_d}[(d+1)p_{d+1}(t) - dp_d(t)] \tag{8}$$

Since we start with finite $p_d(0)$ for $d = 1, 4$, we will have nonzero terms for $t > 0$ for $d = 0, 1, 2, 3, 4$. In fact, by randomly removing a node and its attached edges, we might decrease the degrees of its previously attached nodes.

ODE for the rate equation

For our specific problem, if we define

$$A(t) = \frac{1}{p_0(t) + p_1(t) + p_2(t)} \qquad B(t) = \frac{p_1(t) + 2p_2(t)}{c(t)} A(t)$$

we get the system of equations

$$\begin{aligned}
\frac{\mathrm{d}p_0(t)}{\mathrm{d}t} &= \frac{1 - A(t)}{1 - t} p_0(t) + \frac{B(t)}{1 - t} p_1(t) \\
\frac{\mathrm{d}p_1(t)}{\mathrm{d}t} &= \frac{1 - A(t) - B(t)}{1 - t} p_1(t) + \frac{2B(t)}{1 - t} p_2(t) \\
\frac{\mathrm{d}p_2(t)}{\mathrm{d}t} &= \frac{1 - A(t) - 2B(t)}{1 - t} p_2(t) + \frac{3B(t)}{1 - t} p_3(t) \\
\frac{\mathrm{d}p_3(t)}{\mathrm{d}t} &= \frac{1 - 3B(t)}{1 - t} p_3(t) + \frac{4B(t)}{1 - t} p_4(t) \\
\frac{\mathrm{d}p_4(t)}{\mathrm{d}t} &= \frac{1 - 4B(t)}{1 - t} p_4(t)
\end{aligned}$$

where

$$p_0(0) = 0 \quad p_1(0) = 1 - \pi \quad p_2(0) = 0 \quad p_3(0) = 0 \quad p_4(0) = \pi$$

ODE for the rate equation

We can rephrase the ODE as

$$\frac{\mathrm{d}\vec{p}(t)}{\mathrm{d}t} = M(t)\vec{p}(t) \tag{9}$$

where

$$M(t) = \frac{1}{1-t}\begin{pmatrix} 1-A(t) & B(t) & 0 & 0 & 0 \\ 0 & 1-A(t)-B(t) & 2B(t) & 0 & 0 \\ 0 & 0 & 1-A(t)-2B(t) & 2B(t) & 0 \\ 0 & 0 & 0 & 1-3B(t) & 4B(t) \\ 0 & 0 & 0 & 0 & 1-4B(t) \end{pmatrix}$$

Equation 9 is an initial value problem and it can be solved numerically with the Runge-Kutta method.

Runge-Kutta method for the IVP

An IVP can be solved numerically with the Runge-Kutta method.

For each timestep from t to $t+h$ we can compute [2] the function value as

$$K_1 = hf(t, u) \tag{10}$$
$$K_2 = hf(t + h/2, u + K_1/2) \tag{11}$$
$$K_3 = hf(t + h/2, u + K_2/2) \tag{12}$$
$$K_4 = hf(t + h, u + K_3) \tag{13}$$
$$u(t+h) \approx u + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4) \tag{14}$$

Example of the ODE: $\pi = 0.3$

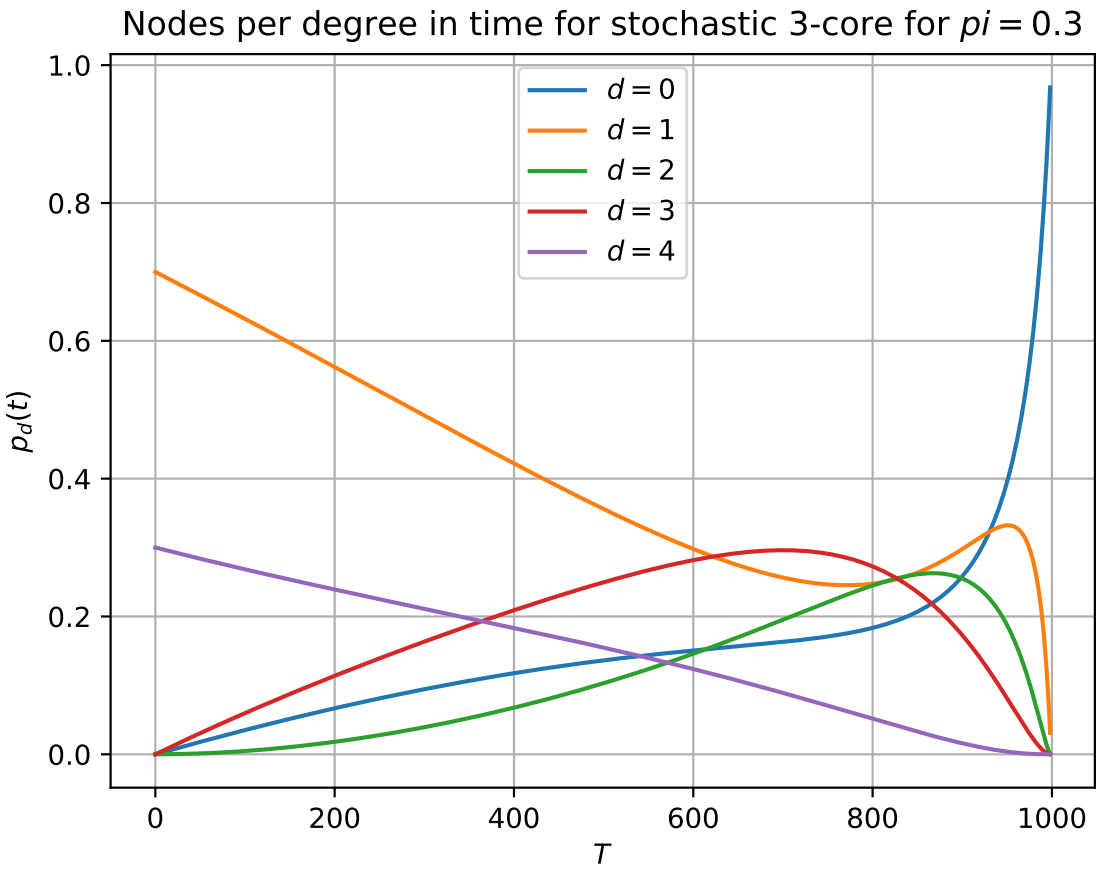

Figure: Evolution of the ODE for $\pi = 0.3$

Example of the ODE: $\pi = 0.8$

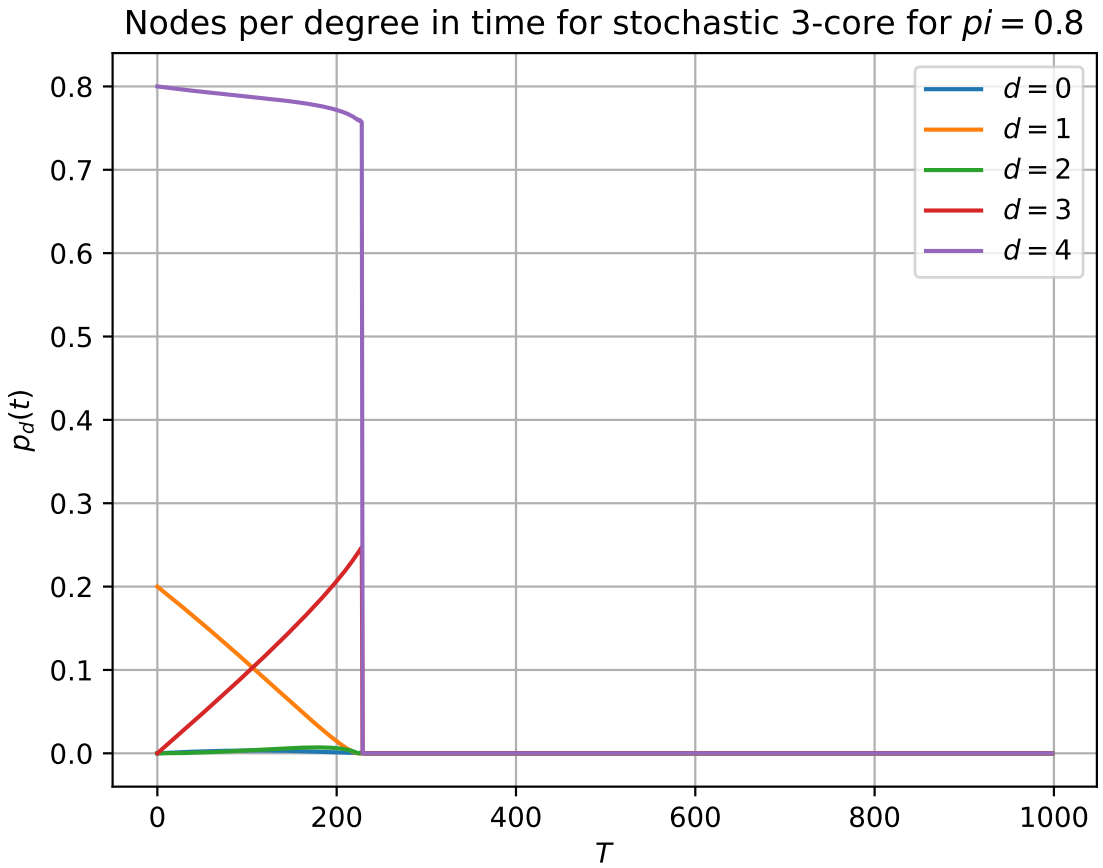

Figure: Evolution of the ODE for $\pi = 0.8$, the time where all rates go to zero is the *halting time* T_f

The size of the 3-core

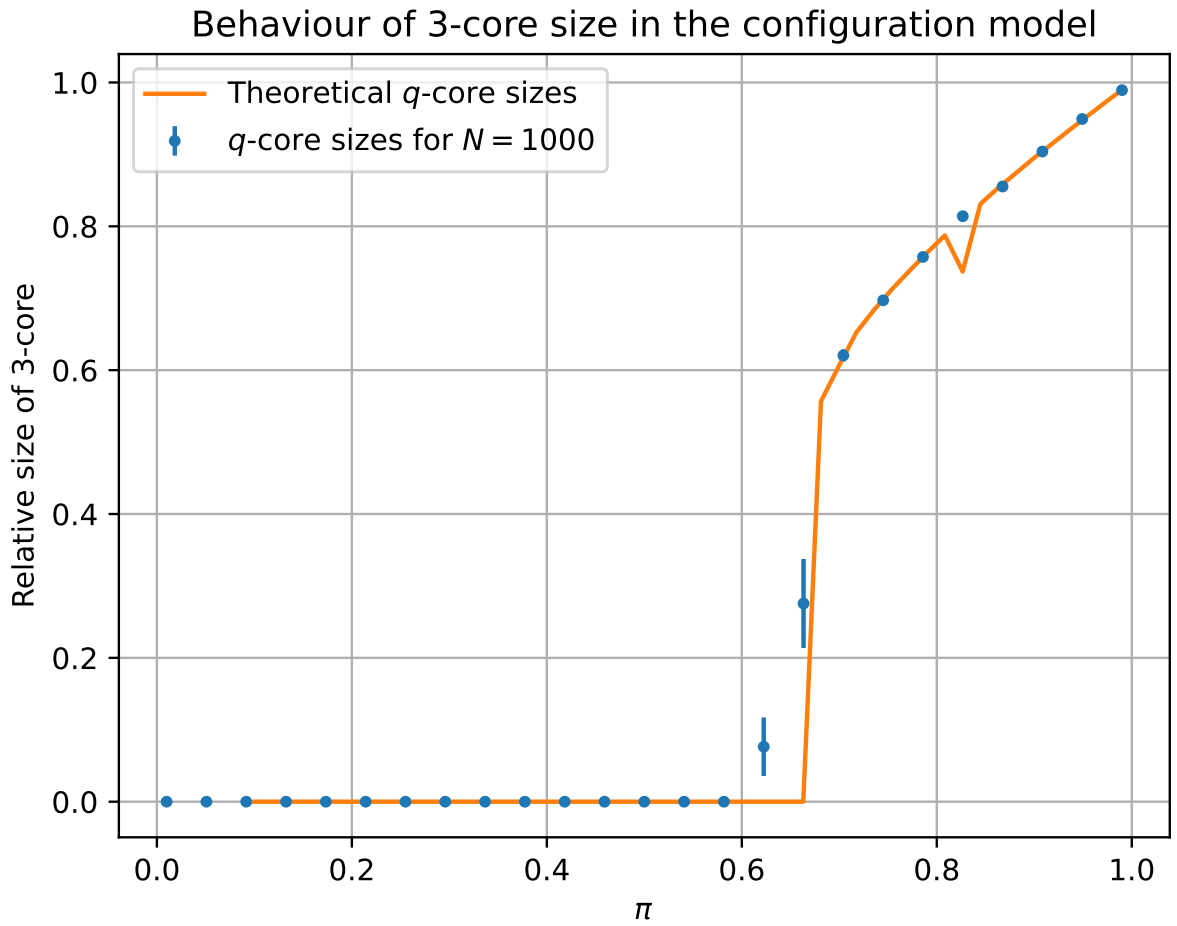

Figure: Comparison between theoretical value and measures from random instances of the size of the 3-core

Bibliography

[1] Alexander K. Hartmann and Martin Weigt. *Phase transitions in combinatorial optimization problems: basics, algorithms and statistical mechanics*. Weinheim: Wiley-VCH, 2005. 348 pp. ISBN: 978-3-527-40473-5.

[2] Brenton LeMesurier. "Initial Value Problems for ODEs, Part 2: Runge-Kutta Methods". In: *Elementary Numerical Analysis with Python*. URL: https://lemesurierb.people.cofc.edu/elementary-numerical-analysis-python/notebooks/ODE-IVP-2-Runge-Kutta-python.html.