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.

Data structures for storing a graph

A graph $G = (E, V)$ is determined by its *edge set* and its *vertices*. There are two main ways in which we could store the data about a graph:

- We might think most straightforwardly of using an adjacency matrix, an $N \times N$ matrix ($N = |V|$) filled with zeros for nodes that are not connected and ones for those that are connected.
- Another way of storing graphs is by the use of edge lists, which are lists of the tuples referring to the connected nodes.
- We could also store the information of graphs by enumerating the neighbourhoods of each node.

The sparsity of the graph (i.e. usually most nodes are not connected) gives us a hint that we should prefer the last two data structures. This will be the case in most of the examples, but sometimes it will also be useful to have an adjacency matrix (when we will resort to the use of tools from linear algebra to study 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

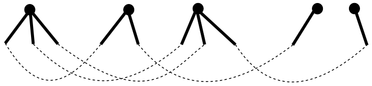


Figure: Connecting stubs in the configuration model

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

```
 1: for $k_i$ in degree sequence $\vec{k}$ do
 2:     Generate node with $k_i$ stubs
 3: end for
 4: while Graph $G$ not generated do
 5:     while Stub list is not exhausted do
 6:         Pick two random stubs
 7:         Create a candidate edge between them
 8:         Check whether they are not multiedge or self-edge
 9:     end while
10: end while
```

Examples of random graphs

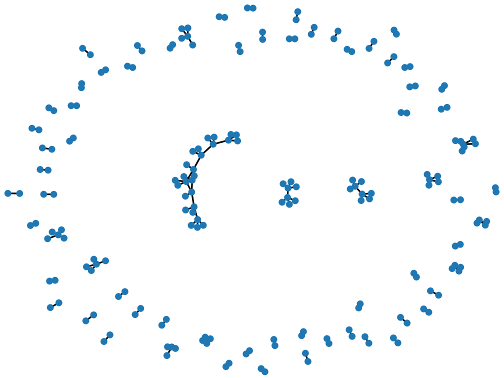


Figure: Instance of a random graph for $\pi = 0.1$

Examples of random graphs

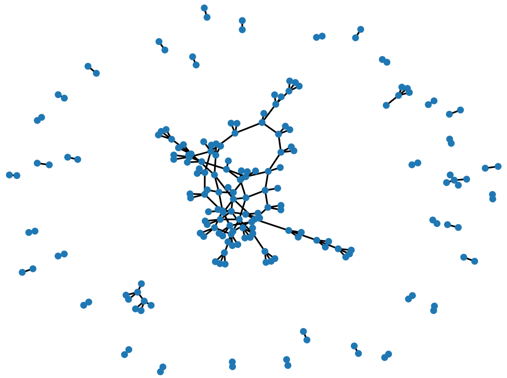


Figure: Instance of a random graph for $\pi = 0.3$

Examples of random graphs

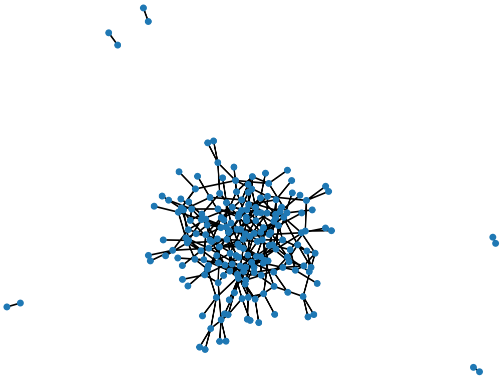


Figure: Instance of a random graph for $\pi = 0.7$

Detecting the giant component in an instance

We explore the connected components with a Breadth-First Search (BFS) Algorithm 2.

Once all connected components have been detected, they are stored in a dictionary-like object and the size of the biggest is given as an output.

BFS algorithm

Algorithm 2 Breadth-First Search algorithm implementation

```
1:  All nodes are unvisited
2:  while Set of unvisited nodes is nonempty do
3:      Initialize queue with unvisited node
4:      while Queue is nonempty do
5:          Assign last node of queue to component $t$ and remove from the
            queue
6:          for All neighbours of the node do
7:              if The neighbour doesn't belong to component $t$ then
8:                  Add to component $t$
9:                  Add to queue
10:             end if
11:         end for
12:     end while
13: end while
```

The onset of criticality

On an intuitive ground, we can argue that in an ER graph the number of nodes visited after a k steps walk is on average given by $c^k = e^{k \log c}$. Then for $c < 1$ the number gets smaller and smaller while for $c > 1$ it becomes extensive.

On the same line of reasoning, we can replace c with the average number of nearest neighbours which is given by the average excess degree, hence we get that the onset of criticality happens for

$$\frac{\left\langle d^2 \right\rangle - \langle d \rangle}{\langle d \rangle} \overset{!}{=} 1 \tag{1}$$

The onset of criticality

Then

$$\left\langle d^2 \right\rangle - 2 \left\langle d \right\rangle \overset{!}{=} 0$$

which in our case is given by

$$(1-\pi) + 16\pi - 2[(1-\pi) + 4\pi] \overset{!}{=} 0$$

The final result is that the criticality threshold is given by

$$\pi = \frac{1}{9} \tag{2}$$

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 [2, p. 56].

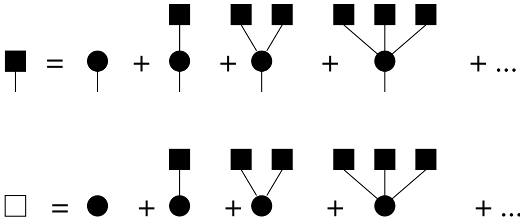


Figure: Scheme for the iterative solution to find the probability that a node belongs to the giant component and the excess probability

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

$$\begin{cases} \mu = q_1 + q_2\mu + q_3\mu^2 + \dots \\ 1 - \gamma = p_0 + p_1\mu + p_2\mu^2 + \dots \end{cases} \tag{3}$$

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

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

where

$$q_1 = \frac{1-\pi}{1+3\pi} \qquad q_4 = \frac{4\pi}{1+3\pi}$$

We can solve the first third order equation in Eq. 4 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

$$\gamma = 1 - (1-\pi)\frac{1-\sqrt{\pi}}{2\sqrt{\pi}} - \pi\left(\frac{1-\sqrt{\pi}}{2\sqrt{\pi}}\right)^4 \tag{5}$$

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

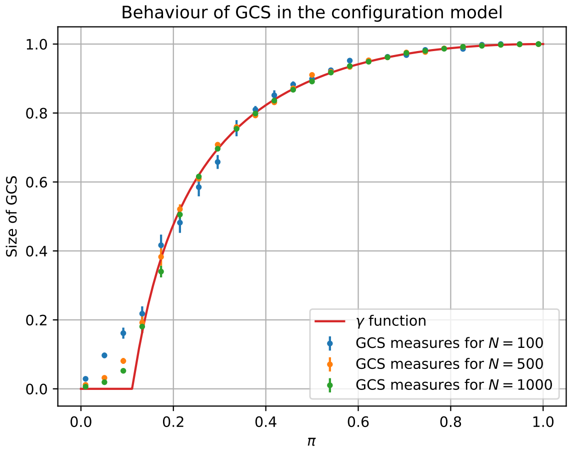


Figure: Comparison between theoretical value and measures from random instances of the size of the giant component

The deterministic q-core algorithm

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

Algorithm 3 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 4 Algorithm for stochastic q-core detection

```
1: while Nodes with $d < q$ are present do
2:     Select randomly a node with degree $d < q$
3:     if The node has degree $d < q$ then
4:         Remove node
5:     end if
6: 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 4 can be described by rate equations using the Wormwald method. [2, 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

$$p_d(t) = \frac{N_d(t)}{N(t)} = \frac{N_d(t)}{N[1-t]} \tag{6}$$

ODE for the rate equation

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

$$N_d(T+1) - N_d(T) = -\frac{\chi_d p_d}{\overline{\chi}_d} + \frac{\overline{\chi_d d}}{\overline{\chi}_d c(t)}[(d+1)p_{d+1}(t) - dp_d(t)] \quad (7)$$

where χ_d is the indicator function

$$\chi_d = \begin{cases} 1 & \text{if } d < q \\ 0 & \text{if } d \geq q \end{cases} \quad (8)$$

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

$$c(t) = \sum_d dp_d(t) \quad (9)$$

ODE for the rate equation

Starting from Eq. 7 we can find the equations for $p_d(t)$

$$\partial_t[(1-t)p_d(t)] = -\frac{\chi_d p_d}{\overline{\chi}_d} + \frac{\overline{d\chi_d}}{c(t)\overline{\chi}_d}[(d+1)p_{d+1}(t) - dp_d(t)] \qquad (10)$$

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

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

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

$$\frac{\mathrm{d}\vec{p}(t)}{\mathrm{d}t} = M(t)\vec{p}(t) \tag{11}$$

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 11 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 [3] the function value as

$$\begin{aligned} K_1 &= hf(t, u) && (12) \\ K_2 &= hf(t + h/2, u + K_1/2) && (13) \\ K_3 &= hf(t + h/2, u + K_2/2) && (14) \\ K_4 &= hf(t + h, u + K_3) && (15) \\ u(t + h) &\approx u + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4) && (16) \end{aligned}$$

Example of the ODE: $\pi = 0.3$

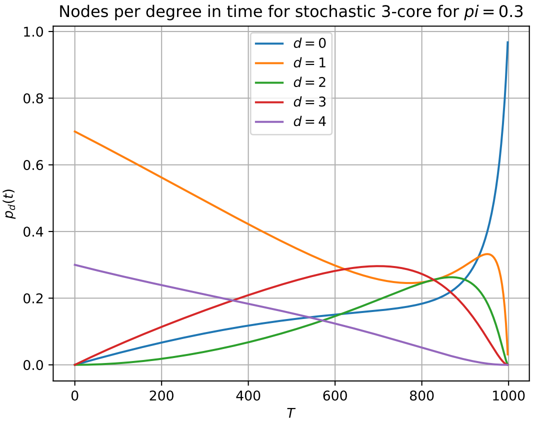


Figure: Evolution of the ODE for $\pi = 0.3$

Example of the ODE: $\pi = 0.8$

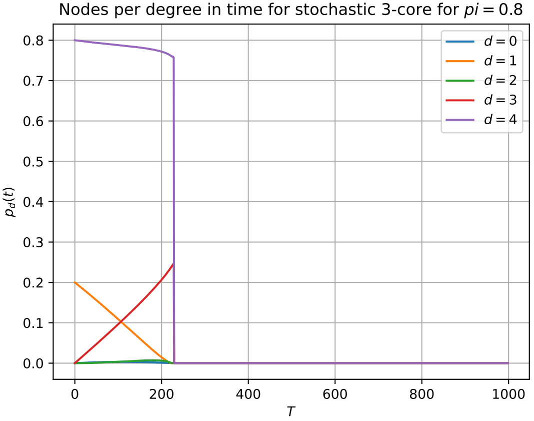


Figure: Evolution of the ODE for $\pi = 0.8$, the time where all rates go to zero is the *halting time* T_f

The size of the 3-core

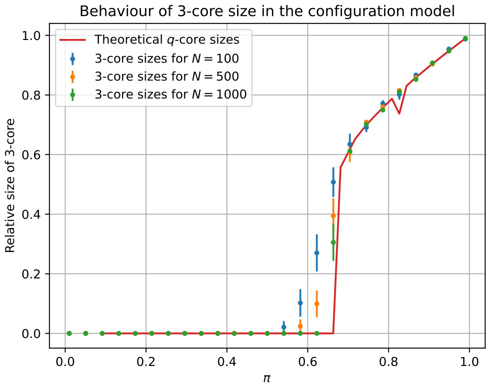


Figure: Comparison between theoretical value and measures from random instances of the size of the 3-core. It emerges in $\pi \approx 0.68$ and jumps to a size of ≈ 0.56.

The ferromagnetic Ising model

The ferromagnetic Ising model is a model described on the graph structure. Its behaviour is described by the Hamiltonian

$$\mathcal{H} = - \sum_{(ij) \in E} \sigma_i \sigma_j \tag{17}$$

where

$$\sigma_i = \pm 1 \qquad \forall i \in 1, ..., N,$$

E is the edge set and σ_i are binary values defined on the i-th node.

What do we expect

Let us think about what can we expect from the ferromagnetic Ising model given the knowledge about the structure.

- At small values of π we have only small components. Since they are all independent, they will have on average zero magnetization.
- Below a certain criticality threshold we get an extensive giant component: these spins are correlated.

At low temperature we get the ground state. But since only spins that belong to a connected component try to align, we expect that on average the magnetizations of spins that do not belong to the GC will be zero and that the total average magnetization will depend on the size of the GC.

MCMC

In order to simulate a ferromagnetic Ising model on a random graph sampled from $\mathcal{G}$, we might use Monte Carlo Markov Chains.

MCMCs are a class of algorithms that, given an equilibrium probability distribution, are able to converge to it using a Markov chain.

The idea is that we are able, using a set of rules that satisfy detailed balance, to reach after a certain number of steps a satisfactory equilibrium configuration. [4, pp. 31–42].

Usually these rules encode the information whether the new state is “better” or “worse” than the old one, and decide how to proceed.

The Metropolis-Hastings algorithm

The Metropolis-Hastings flips one spin per step and the difference in energy between configurations and the temperature of the system is evaluated in order to accept or reject the step. [4, p. 46]

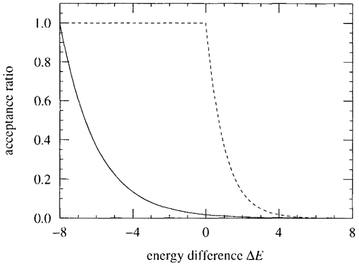


Figure: Metropolis-Hastings acceptance ratios (dashed line) from [4, p. 48]

The Metropolis-Hastings algorithm

Algorithm 5 Metropolis-Hastings algorithm

```
1: Initialize spins $\vec{\sigma}$
2: while System not at equilibrium do
3:     Choose a random $i$-th site
4:     Compute the $\Delta E = \mathcal{H}(\vec{\sigma}^{\text{new}}) - \mathcal{H}(\vec{\sigma}^{\text{old}})$
5:     Generate a random number $r \in [0, 1]$
6:     if $r < \exp(-\beta \Delta E)$ then
7:         Flip the spin $\sigma_i^{\text{new}} = -\sigma_i^{\text{old}}$
8:     else
9:         Do nothing
10:    end if
11: end while
```

The Wolff algorithm

The Wolff algorithm picks a random spin and probabilistically grows the cluster adding spins that are aligned. It then flips the entire cluster at once. [4, p. 91]

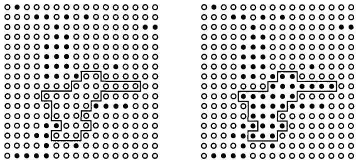


Figure: Cluster spin flip using Wolff algorithm from [4, p. 91]

The Wolff algorithm

The Wolff algorithm picks a random spin and probabilistically grows the cluster adding spins that are aligned. It then flips the entire cluster at once. [4, p. 91]

Algorithm 6 Wolff algorithm

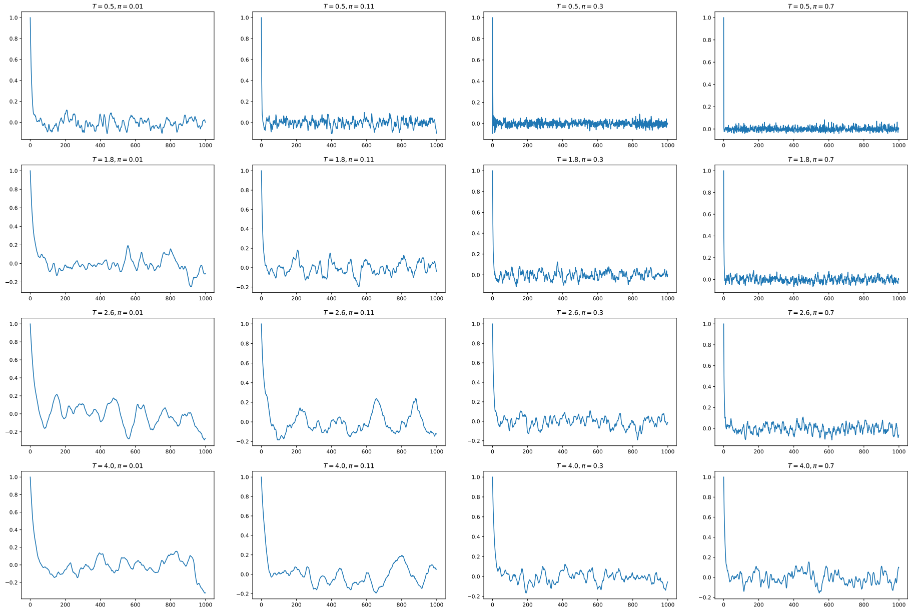
```
 1: Initialize spins $\vec{\sigma}$
 2: while System not at equilibrium do
 3:     while Aligned neighbourhood is not exhausted do
 4:         Look at neighbours
 5:         if They are aligned to the first spin then
 6:             Add to cluster with probability $P_{\text{add}} = 1 - e^{-2\beta J}$
 7:         end if
 8:     end while
 9:     Flip the spins of the cluster
10: end while
```

Sampling equilibrium states of Ising on RG

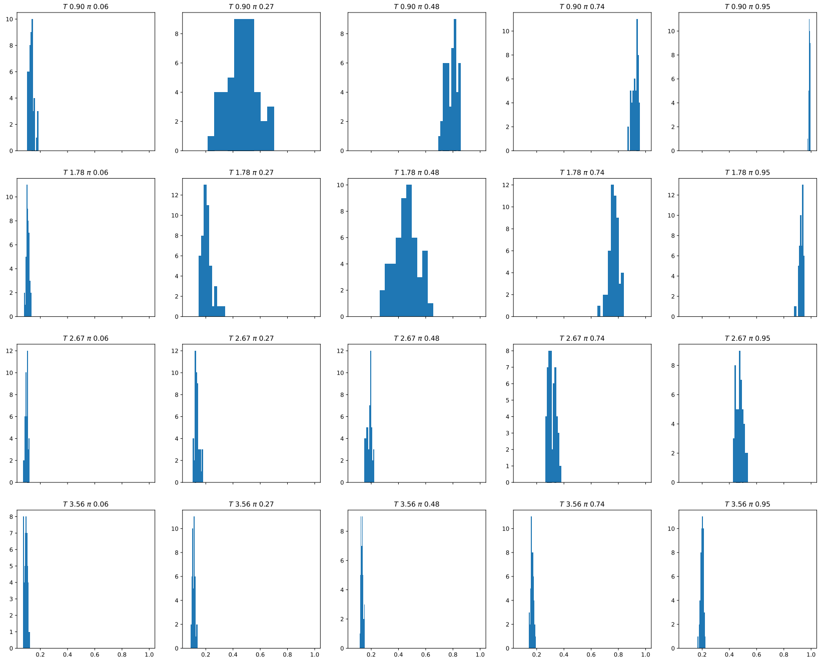
We choose to use the Wolff algorithm for speeding up the sampling also in the critical phase. The implementation is similar to the 2D Ising case, taking care of the definitions of neighbourhoods.

Some runs over sets of (π, T) parameters are performed in order to understand how much time does the system take to equilibrate in terms of magnetization $m = \sum_i \sigma_i / N$.

After the equilibration time τ_{eq} has passed, the magnetization m is measured every τ_{sample} steps. The samples are then averaged and yield the final result for each RG instance.



$T = 0.5, \pi = 0.01$
$T = 0.5, \pi = 0.11$
$T = 0.5, \pi = 0.3$
$T = 0.5, \pi = 0.7$
$T = 1.8, \pi = 0.01$
$T = 1.8, \pi = 0.11$
$T = 1.8, \pi = 0.3$
$T = 1.8, \pi = 0.7$
$T = 2.6, \pi = 0.01$
$T = 2.6, \pi = 0.11$
$T = 2.6, \pi = 0.3$
$T = 2.6, \pi = 0.7$
$T = 4.0, \pi = 0.01$
$T = 4.0, \pi = 0.11$
$T = 4.0, \pi = 0.3$
$T = 4.0, \pi = 0.7$



T 0.90 π 0.06
T 0.90 π 0.27
T 0.90 π 0.48
T 0.90 π 0.74
T 0.90 π 0.95
T 1.78 π 0.06
T 1.78 π 0.27
T 1.78 π 0.48
T 1.78 π 0.74
T 1.78 π 0.95
T 2.67 π 0.06
T 2.67 π 0.27
T 2.67 π 0.48
T 2.67 π 0.74
T 2.67 π 0.95
T 3.56 π 0.06
T 3.56 π 0.27
T 3.56 π 0.48
T 3.56 π 0.74
T 3.56 π 0.95
0.2
0.4
0.6
0.8
1.0

Belief Propagation

Another way to study the Ising model is by using the belief propagation equations.

Given a random graph instance, belief propagation allows for an algorithmic way to study its properties.

If a graph is tree-like, then we can compute the cavity marginals $\mu_{i\to j}(\sigma_i)$, which are the probabilities that the i-th site has spin σ_i given its neighbourhood ∂i except j. Then we can find the true marginal $\mu_i(\sigma_i)$.

The algorithm becomes approximate in the case of graphs that are not tree-like and can be solved by finding a fixed point for the cavity marginals.

BP equations for Ising

The BP equations are

$$\mu_{i\to j}(\sigma_i) = \frac{1}{\tilde{Z}_{i\to j}} \prod_{k\in\partial i\setminus j} \sum_{\sigma_k} e^{\beta\sigma_i\sigma_k} \mu_{k\to i}(\sigma_k) \tag{18}$$

$$\mu_i(\sigma_i) = \frac{1}{\tilde{Z}_i} \prod_{k\in\partial i} \sum_{\sigma_k} e^{\beta\sigma_i\sigma_k} \mu_{k\to i}(\sigma_k) \tag{19}$$

and in the case of the Ising model we get

$$h_{i\to j} = \frac{1}{2\beta} \sum_{k\in\partial i\setminus j} \log\left[\frac{\cos\beta(h_{k\to i}+1)}{\cos\beta(h_{k\to i}-1)}\right] \tag{20}$$

$$h_i = \frac{1}{2\beta} \sum_{k\in\partial i} \log\left[\frac{\cos\beta(h_{k\to i}+1)}{\cos\beta(h_{k\to i}-1)}\right] \tag{21}$$

BP equations for Ising

The Equation 20 is then run iteratively. We just initialize randomly the fields, and run

$$h_{i\to j}^{\text{new}} = \frac{1}{2\beta} \sum_{k\in\partial i\setminus j} \log\left[\frac{\cos\beta(h_{k\to i}^{\text{old}}+1)}{\cos\beta(h_{k\to i}^{\text{old}}-1)}\right]$$

many times until we reach convergence (e.g. we can check that the elementwise difference between cavity fields are less than a given threshold). Once the fixed point has been found, we run Eq. 21 just once and we get the fields. These can be plugged into

$$m_i = \tanh(\beta h_i) \tag{22}$$

to find the local magnetizations. The average magnetization is the computed as $m = \sum_i m_i/N$.

Distribution of cavity and effective fields

Belief propagation is an algorithmic scheme that applies instance-wise. We might ask ourselves what happens instead in a typical case.

It might also be desirable that this description does not depend on the size of an instance or the underlying structure.

Distribution of cavity and effective fields

The cavity and effective fields have respectively the following distributions

$$\eta_{\mathrm{cav}}(h) = \sum_{d=0}^{\infty} q_{d+1} \int \left(\prod_{k=1}^{d} \mathrm{d}h_k \, \eta_{\mathrm{cav}}(h_k) \right) \delta\left[h - \frac{1}{2\beta} \sum_{k=1}^{d} \log \frac{\cosh \beta(h_k + 1)}{\cosh \beta(h_k - 1)} \right] \tag{23}$$

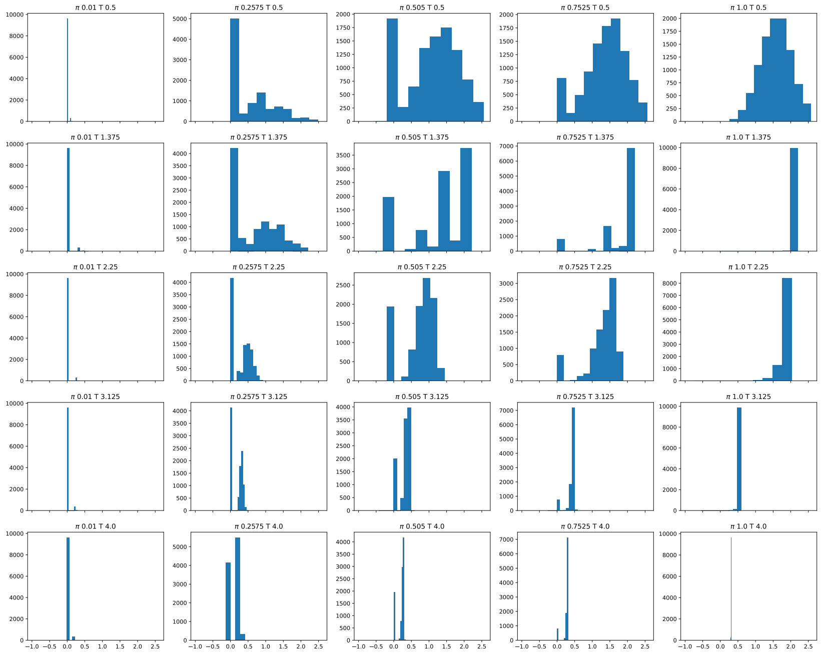
$$\eta(h) = \sum_{d=0}^{\infty} p_d \int \left(\prod_{k=1}^{d} \mathrm{d}h_k \, \eta_{\mathrm{cav}}(h_k) \right) \delta\left[h - \frac{1}{2\beta} \sum_{k=1}^{d} \log \frac{\cosh \beta(h_k + 1)}{\cosh \beta(h_k - 1)} \right] \tag{24}$$

Population dynamics

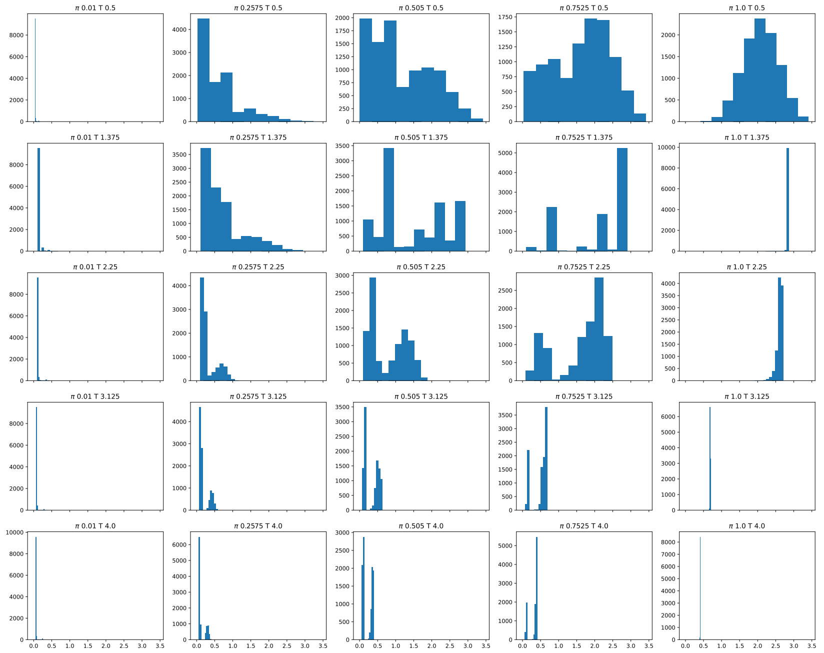
We can use a population dynamics algorithm in order to find the distribution of cavity and effective fields, as well as the magnetization distribution.

- Initialize a population of L fields $\vec{h}$ randomly
- At each step pick d elements of the population $\vec{h}$ randomly, compute the function inside the argument of the delta and replace another random member of the population with the computed value.
- After many steps, the population should converge *in distribution* to a fixed (distribution) point. We can check this by monitoring the moments.

Once the fixed point is reached, we can now find a new L elements *effective fields* population, by running the same algorithm but leaving the cavity fields population unchanged.



π 0.01 T 0.5
π 0.2575 T 0.5
π 0.505 T 0.5
π 0.7525 T 0.5
π 1.0 T 0.5
π 0.01 T 1.375
π 0.2575 T 1.375
π 0.505 T 1.375
π 0.7525 T 1.375
π 1.0 T 1.375
π 0.01 T 2.25
π 0.2575 T 2.25
π 0.505 T 2.25
π 0.7525 T 2.25
π 1.0 T 2.25
π 0.01 T 3.125
π 0.2575 T 3.125
π 0.505 T 3.125
π 0.7525 T 3.125
π 1.0 T 3.125
π 0.01 T 4.0
π 0.2575 T 4.0
π 0.505 T 4.0
π 0.7525 T 4.0
π 1.0 T 4.0



π 0.01 T 0.5
π 0.2575 T 0.5
π 0.505 T 0.5
π 0.7525 T 0.5
π 1.0 T 0.5
π 0.01 T 1.375
π 0.2575 T 1.375
π 0.505 T 1.375
π 0.7525 T 1.375
π 1.0 T 1.375
π 0.01 T 2.25
π 0.2575 T 2.25
π 0.505 T 2.25
π 0.7525 T 2.25
π 1.0 T 2.25
π 0.01 T 3.125
π 0.2575 T 3.125
π 0.505 T 3.125
π 0.7525 T 3.125
π 1.0 T 3.125
π 0.01 T 4.0
π 0.2575 T 4.0
π 0.505 T 4.0
π 0.7525 T 4.0
π 1.0 T 4.0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5

Linear stability analysis and phase diagram

We know that the cavity fields have a distribution given by Eq. 23 and also we know that the paramagnetic solution $\delta(h)$ is a fixed point of the BP equations. We can perform a linear stability analysis in order to understand how a small perturbation of the system affects its distribution.

We shall be able to get also the phase diagram at the end of the analysis.

Linear stability analysis

A small perturbation of the field is given by

$$\eta^{(0)}(h) = \delta(h - \epsilon_0) \tag{25}$$

if we replace it in Eq. 23 we get that at the next iteration

$$\begin{aligned}
\epsilon_1 &= \int \mathrm{d}h\, \eta^{(1)}(h) h \qquad (26)\\
&= \sum_{d=0}^{\infty} q_{d+1} \frac{d}{2\beta} \log\left[\frac{\cosh\beta(\epsilon_0+1)}{\cosh\beta(\epsilon_0-1)}\right] \qquad (27)\\
&\approx \sum_{d=0}^{\infty} q_{d+1} d\epsilon_0 \tanh\beta \qquad (28)
\end{aligned}$$

Phase diagram

Recall that since the excess degree distributions are

$$q_1 = \frac{1-\pi}{1+3\pi} \qquad q_4 = \frac{4\pi}{1+3\pi}$$

then we can write

$$\begin{aligned} \frac{\epsilon_1}{\epsilon_0} &= 3q_4 \tanh\beta && (29) \\ &= \frac{12\pi}{1+3\pi}\tanh\beta && (30) \end{aligned}$$

The system is stable if the ratio is less than one, hence bringing us back to the paramagnetic solution. Otherwise, the system will go in the ferromagnetic phase.

Phase diagram

This means that we can draw the phase diagram, given that for $\epsilon_1/\epsilon_0 = 1$ we get the critical line. It corresponds to the function

$$T = \frac{1}{\operatorname{arctanh}\left(\frac{1+3\pi}{12\pi}\right)}$$

that has an onset in

$$\pi_0 = \lim_{T\to 0^+} \frac{1}{12\tanh T^{-1} - 3} = \frac{1}{9}$$

which is exactly the threshold for the onset of the giant component.

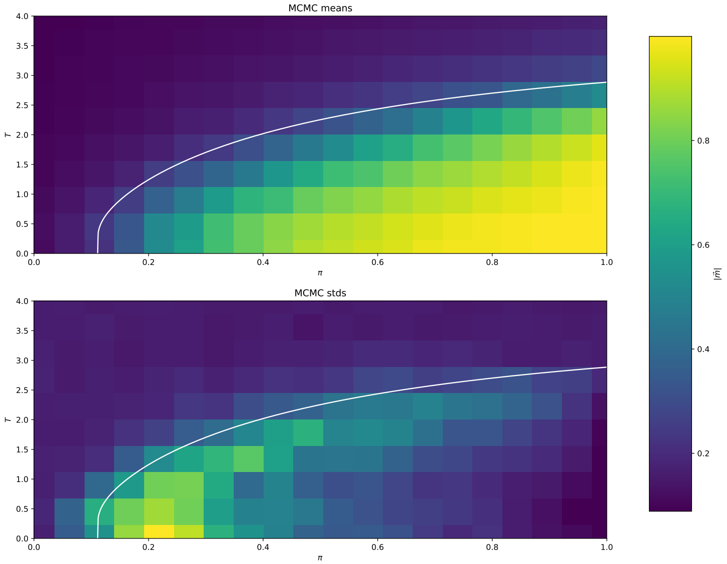


Figure: Heatmap for MCMC sampled magnetization means and stds

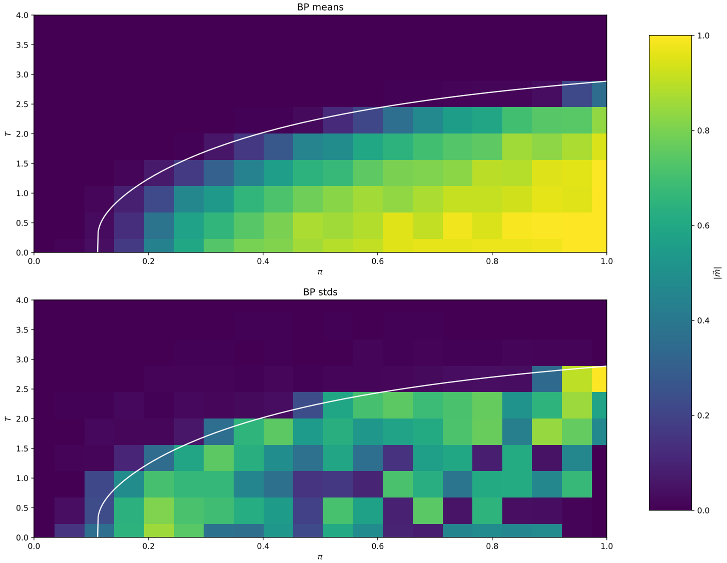


Figure: Heatmap for BP sampled magnetization means and stds

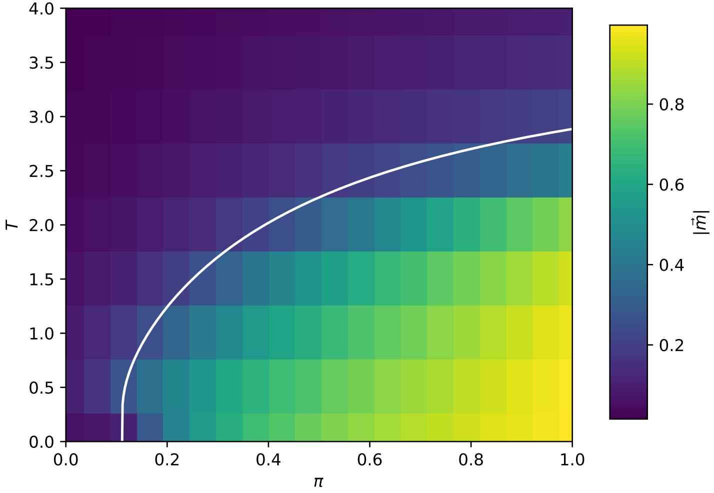


Figure: Heatmap for magnetization given by PD algorithm

Inverse problem

Once we have a graph instance G sampled from $\mathcal{G}$, and we have defined a spin model on the graph (e.g. a ferromagnetic Ising model), we might want to understand whether we are able to reconstruct the topology of the graph (i.e. the edges) just by looking at the behaviour of the single components (i.e. the nodes).

This task is hence that of estimating “the rule” once a set of data is given, and it is called an inverse problem.

Inverse problem

In particular we will construct the dataset as follows:

- Pick a random $G \in \mathcal{G}$
- Initialize spins over G and perform an MCMC run in order to reach equilibrium at a temperature $T > T_c$ (paramagnetic phase)
- Repeat this M times in order to get M samples of equilibrium spin configurations for G

Then we will attempt to perform inference over the dataset: in particular we will study its correlation matrix and the naive mean field approximation inference.

Correlations

We can, first of all, look at correlations between components. Given the Σ matrix of spin configurations (an $M \times N$ matrix), we can compute the correlation matrix

$$C_{ij} = \langle \sigma_i \sigma_j \rangle_m - \langle \sigma_i \rangle_m \langle \sigma_j \rangle_m \tag{31}$$

Then we can compute the positive predictive value (PPV) by sorting the values of the correlation matrix in descending order, taking the greatest n values and checking how many of the corresponding pairs of nodes are really connected by edges in the original graph.

$$\mathrm{PPV}(n) = \frac{\#(i,j) \in E \text{ for } (i,j) \in n\text{-th ranking of highest correlations}}{n} \tag{32}$$

Boltzmann machines

In order to perform network reconstruction on categorical variables (such our case, i.e. $\sigma_i = \pm 1$) we can use the Boltzmann machine approach.

By enforcing the maximisation of the log-likelihood over the fields parameters h_i and the interactions parameter J_{ij} we get respectively the moment-matching conditions between one- and two-points statistics of the data and the model. [1, p. 89]

$$\frac{\partial \mathcal{L}}{\partial h_i} \overset{!}{=} 0 \quad \Rightarrow \quad \mu_i = \langle \sigma_i \rangle \tag{33}$$

$$\frac{\partial \mathcal{L}}{\partial J_{ij}} \overset{!}{=} 0 \quad \Rightarrow \quad \mu_{ij} = \langle \sigma_i \sigma_j \rangle \tag{34}$$

Boltzmann Machine Learning

Solving directly Equations 33 and 34 is nontrivial, since finding the averages over the model distributions is hard.

The problem can be recasted into an ML problem (Boltzmann machine learning), where we would iteratively have h_i and J_{ij}, and we would perform an MCMC run until equilibration, then we would update the parameters

$$h_i^{t+1} = h_i^t + \eta_h(\mu_i - \langle \sigma_i \rangle)$$
$$J_{ij}^{t+1} = J_{ij}^t + \eta_J(\mu_{ij} - \langle \sigma_i \sigma_j \rangle)$$

and so on, for many epochs until we have learned the parameters. This amounts to a gradient descent process over the log-likelihood.

This process still requires many MCMC runs and also explores a high-dimensional space, which makes the learning harder.

Naive Mean Field Approximation

We might want to find an approximation in which we can express fields and couplings analytically.

If we make a mean field ansatz (i.e. the Boltzmann distribution factorizes in the sites) we can recast the problem of maximising a likelihood into that of inverting a matrix.

In fact we can take the mean field free energy

$$G^{\mathrm{mf}}(\vec{m}, J) = -\sum_{i<j} J_{ij} m_i m_j + \sum_i \left[\frac{1+m_i}{2} \log \frac{1+m_i}{2} + \frac{1-m_i}{2} \log \frac{1-m_i}{2} \right] \tag{35}$$

Naive Mean Field Approximation

If we derive Eq. 35 twice wrt m_i we get that the coupling matrix J_{ij}^{mf} is equal to minus the inverse correlation matrix $[C]_{ij}^{-1}$, which is much easier to compute than the maximum log-likelihood.

$$J_{ij}^{\text{mf}} = -[C]_{ij}^{-1} \tag{36}$$

Then we could compute the fields since $h_i^{\text{mf}} = \frac{\partial G}{\partial m_i}$ hence

$$h_i^{\text{mf}} = -\sum_{j \neq i} J_{ij}^{\text{mf}} m_j + \text{arctanh} m_i \tag{37}$$

[5, p. 17]

Results for different dataset sizes

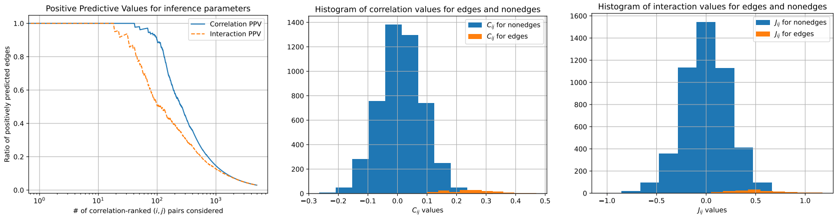


Figure: PPV and distribution of J and C values for $M = 200$

Results for different dataset sizes

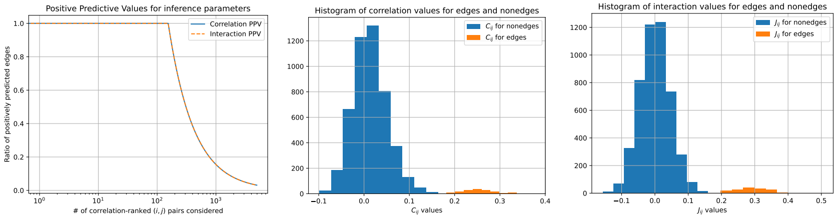


Figure: PPV and distribution of J and C values for $M = 1000$

Comments on inverse problem techniques

We can see that

- as M increases, we get better results since the PPV stay high for a longer ordered amount of largest values of C or J
- in the $M = 200$ case the performances of the NMF interaction matrix J are slightly worse than those of the correlation matrix
- in the $M = 2000$ case we see that the PPV is substantially the same for C and J
- we can also evaluate the ability to reconstruct the network by the ease of telling apart the pairs that are in the edge set and those that are not (C and J perform approximately the same)

Correlations are usually deemed to be a bad variable for reconstructing network edges. However, since the interactions encoded in the Ising RG are pairwise, in this particular case correlations work well.

Bibliography I

[1] Simona Cocco et al. *From Statistical Physics to Data-Driven Modelling: with Applications to Quantitative Biology*. Oxford, New York: Oxford University Press, Sept. 9, 2022. 192 pp. ISBN: 978-0-19-886474-5.

[2] 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.

[3] 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.

Bibliography II

[4] Mark E. J. Newman and Gerard T. Barkema. *Monte Carlo methods in statistical physics*. Reprinted (with corr.) Oxford: Clarendon Press [u.a.], 2010. 475 pp. ISBN: 978-0-19-851797-9 978-0-19-851796-2.

[5] H. Chau Nguyen, Riccardo Zecchina, and Johannes Berg. "Inverse statistical problems: from the inverse Ising problem to data science". In: *Advances in Physics* 66.3 (July 3, 2017). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00018732.2017.1341604, pp. 197–261. ISSN: 0001-8732. DOI: 10.1080/00018732.2017.1341604. URL: https://doi.org/10.1080/00018732.2017.1341604 (visited on 03/29/2024).