Spin glasses

Santiago Sanz Wuhl

April 11, 2025

1 Introduction

A (classical) spin system with n particles is governed by the Edwards-Anderson Hamiltonian

$$H = -\sum_{1\leq j<i}^{N} J_{ij} s_i s_j, \tag{1}$$

with J_{ij} the matrix elements of a 2-dimensional symmetric matrix, dictating the interactions between the particle at the site i and the particle at the site j, and s_i is the spin of the particle at the site i. In the fully connected version, the interactions are given between every two spin pairs decaying as a power law

$$J_{ij} \sim \frac{\phi_{ij}}{r_{ij}^{1+\sigma}}, \tag{2}$$

following notation from Beyer et al. [2012], $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ and ϕ_{ij} is a standard normal random variable. The spins are organized in a 1-dimensional chain with periodic boundary conditions.

1.1 Dilute spin models

Simulations of these systems are very computationaly expensive, as at each 'time-step', calculations are of complexity $\mathcal{O}(N^2)$. A usual simplification of such problems is the Ising model, in which the sum in (1) is performed only over nearest neighbors, which only approximates short-range interactions. The Ising Hamiltonian is Luijten [2006]

$$\mathcal{H}_{\text{Ising}} = -J\sum_{\langle ij\rangle}^{N} s_i s_j,$$

where J is now a constant. This model is in a sense uninteresting, as it has been shown that 1-dimensional spin glasses with finite non-zero J_{ij} (e.g. the Ising model) does not present phase transitions Rushbrooke and Ursell [1948]. Long range interactions however, have been found to present phase transitions Kot.

In order to study long range interactions and avoid the $\mathcal{O}(N^2)$ computational cost of computing every interaction, a dilute spin model is used Leu. In this model, the interaction J_{ij} is set distance independent, but the probability of having an interaction decays with $\frac{1}{r_{ij}^{\sigma}}$. The coordination number is fixed, and therefore so is the total number of bonds $N_l \leq \frac{N(N-1)}{2}$.

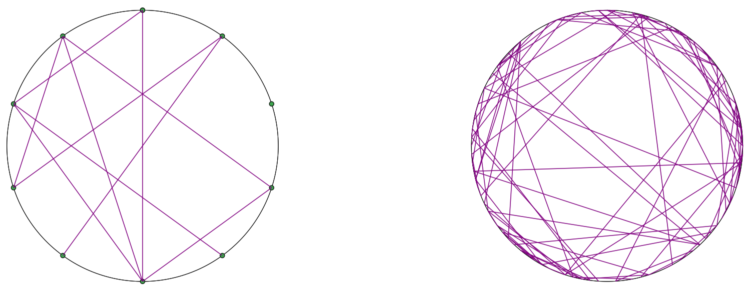

Figure 1: Partially connected graphs of different N and N_l, but same $\sigma = 5$. In the left panel, $N = N_l = 10$, and in the right panel $N = N_l = 100$.

2 Methods

2.1 Walker's Alias method

In order to generate the connections J_{ij} for the dilute spin model, Walker's Alias Walker [1974] method is used. This method allows us to sample random non-uniformly distributed integer numbers, i.e. rolling unfair dice in $\mathcal{O}(1)$ computation complexity with $\mathcal{O}(N^2)$ memory complexity, by splitting the sampling into rolling a fair die and then flipping a biased coin.

The connection set algorithm We will only be interested in the indices i, j for which J_{ij} is non-zero. Furthermore, since J_{ij} is symmetric, we characterize connections by the set $\{i, j\}$. Different connections $\{i, j\}$ are drawn until N_l connections have been drawn. If the connection $\{i, j\}$ exists, discard it and try again. The algorithm to draw random connections is straight-forward: Draw an *uniformly* distributed integer from 1 to N to choose the first particle. The second particle is then chosen by generating a random number $m \in [1, N-1]$, where the probability of rolling m is $P_m = m^{-(1+\sigma)}$.

The sampled connection is thus $\{n, \mod(n+m, N)\}$, with $\{\}$ indicating the data structure *set*. This set is stored in another set, wihch is motivated by their $\mathcal{O}(1)$ lookup time in Julia as was tested by a user Kamiński [2022]. Furthermore, they are not orderered and do not have repeated elements.

The `Julia` code used to generate the non-zero J_{ij} is found in Appendix A, and examples of diluted spin models with $N = N_l$ and $\sigma = 5$ are displayed in 1, with $N = 10$ on the left panel and $N = 100$ on the right panel.

2.2 Cluster counting

In applying Monte Carlo algorithms to calculate thermodynamical averages, a method was introduced by Swendsen and Wang [1987] in which the single spin flip seen in famous models e.g. Ising is extended to flipping whole clusters. To do so, a clustering algorithm is explained here, based on a depth first search (DFS) algorithm.

In order to apply this algorithm, the connections set must be given a new format so that the DFS algorithm can be easier applied. An array `connectionArray` of N entries is created, where each of its entries is an empty array. The array `connectionArray[i]` contains the indices of the nodes to which the node `i` is connected. An example of a `connecionArray` corresponding to a fully connected graph with $N = 4$ is

```
connectionArray = [[2,3,4], [1,3,4], [1,2,4], [1,2,3].
```

The DFS algorithm is in the worst case $\mathcal{O}(V + E)$, with V the number of vertices of the graph and E the number of edges (of each cluster). Two sets are defined

1. `clusters`: The identified clusters
2. `visited`: The visitied nodes

Starting (arbitrarily) at the first node `n=1`. Initialize the array `stack` containing only said node, and an empty set `cluster`. Then,

1. While the stack is non-empty: pop the stack to the variable `curr = pop(stack)`,
2. If `curr` is not in `visited`, add it to `cluster`, and to the `visited` set.
3. Retrieve the „neighbors" of `curr` from `connectionArray[curr]`, and add them to the stack.
4. Repeat step 1.

Reaching an empty stack means there are not anymore non-visited nodes in the current cluster, and therefore `cluster` is returned and added to the set `clusters`.

The procedure is repeated for every node, if it is not in `visited`.

The corresponding `Julia` code can be found in A.2

3 Some preliminary results

With a given probability of forming a bond between two nodes that are at a distance r away from each other to be $r^{-\sigma}$, we expect the correlation function $\xi(r)$(i.e. the distribution of distances at which two nodes are bound) to follow $\sim r^{-\sigma}$. We show in the figure 2 the measured average (black, solid line) correlation function and its standard deviation (black, dashed lines) after generating 1000 realizations of the connection graph using Walker's Alias algorithm to draw the bonds among the nodes. In pink, overlayed on top of the black curve, the fit of the average $\xi(r)$ is shown. We fit $\xi(r)$ to the curve $y = (ar + b)^{-\sigma}$ and obtain a perfect overlay between the data (the average $\xi(r)$) and the fit curve.

Using these algorithms, we first study the number of clusters for different configurations of the system. Figure 3 shows the average number of clusters as a function of the number of bonds, fixing the total number of modes, $N = 1000$ and the $\sigma = 2$.

We also show in figures the distribution of the lengths of the clusters for fixed number of nodes but varying number of bonds and σ. Here we see how a small number of bonds leads to (as one would expect) all clusters being of length 1 (i.e. a very disconnected graph). The distribution skews towards higher cluster sizes with higher number of modes, and finally for a high enough (nBonds=200) number of bonds almost all nodes are connected.

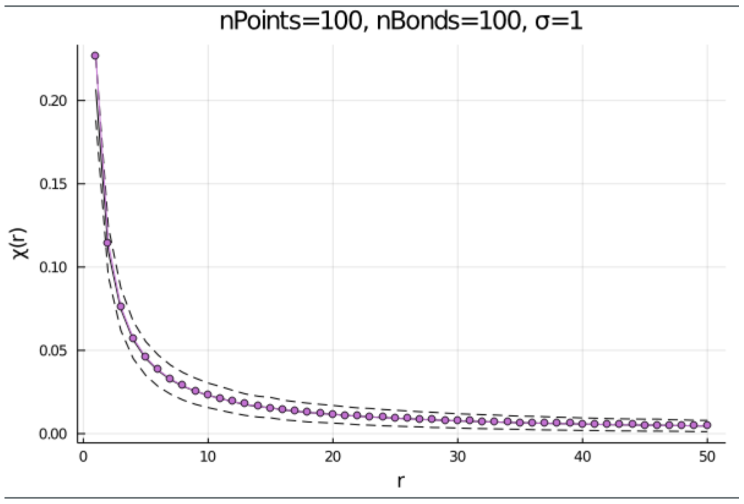

Figure 2: The average correlation function (black) $\xi(r)$ and its standard deviation (dashed lines) of 1000 realizations of a connectivity graph with $N = 100, nBonds = 100, \sigma = 1$. In pink, dotted, the fit of the average correlation function to the curve $y = (4.5r + 0.5)^{-1}$.

4 Miscellaneous notes

Notes on Monte Carlo algorithms Given a partition function

$$Z = \sum_s \exp\left(-i\beta E_s\right), \tag{3}$$

the calculation of thermodynamic observables is extremely computationally expensive, as one has to integrate over the whole phase space. To avoid this, the Monte Carlo method of integration is used, based on the idea of trial and error, from Markov chains. The key characteristic of these chains is that each element **only** depends on the previous element.

Starting from a configuration s_i with a non vanishing Boltzmann factor p_i, a new trial configuration s_j is created with Boltzmann factor p_j. From each state s_i to s_j there is a transition probability represented by a transition matrix π_{ij}. One looks for the transition matrix yielding the equilibrium distribution p_j, so that

$$\sum_i p_i \pi_{ij} = p_j. \tag{4}$$

One looks for the solutions π_{ij} by imposing the condition of *microscopic reversibility* or *detailed balance*

$$p_i \pi_{ij} = p_j \pi_{ji}. \tag{5}$$

Equations (4) and (5) are equivalent if

$$\sum_i \pi_{ji} = 1. \tag{6}$$

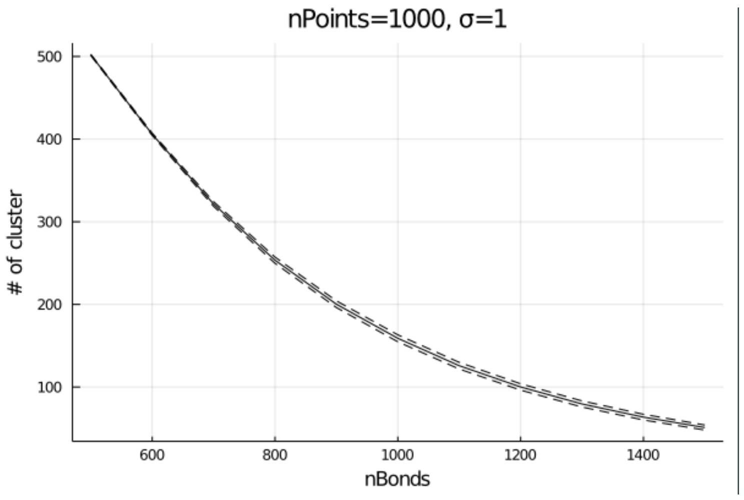

Figure 3: Average number of clusters and standard deviation as a function of number of bonds, for fixed number of nodes and σ.

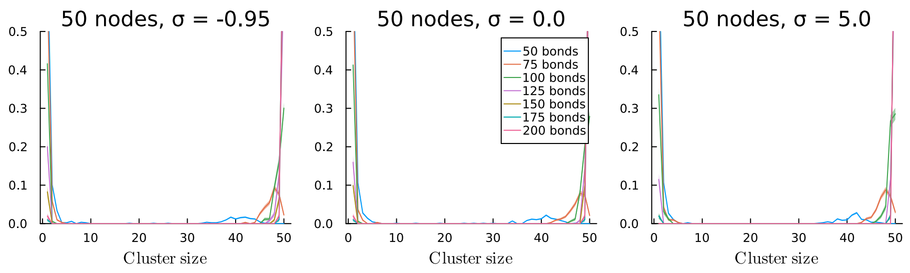

Figure 4: Normalized distributions of cluster sizes for fixed number of nodes $N = 50$ and varying σ values, as indicated at the top of each panel, for different number of bonds among the nodes, as indicated in the legend of the middle panel.

We split each π_{ij} as the product of an *a priori* transition probability α_{ij} of generating s_j from s_i and an acceptance probability P_{ij} of accepting s_j as the new state. We thus write (5) as

$$p_i \alpha_{ij} P_{ij} = p_j \alpha_{ji} P_{ji}. \tag{7}$$

If α is symmetric,

$$\frac{P_{ij}}{P_{ji}} = \exp[-\beta(E_j - E_i)]. \tag{8}$$

This does not uniquely define P. Metropolis et al. [1953] suggests

$$\frac{P_{ij}}{P_{ji}} = \begin{cases} \exp -\beta(E_j - E_i) & E_j > E_i \\ 1 & E_j \leq E_i. \end{cases} \tag{9}$$

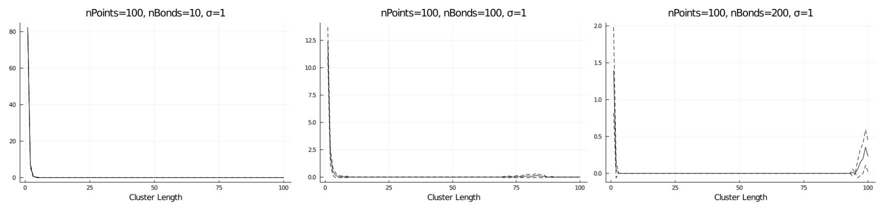

Figure 5: The distribution of the cluster Length for fixed number of nodes $N = 100$ and $\sigma = 1$, for different number of bonds among the nodes. The panels are arranged as in the figure 4

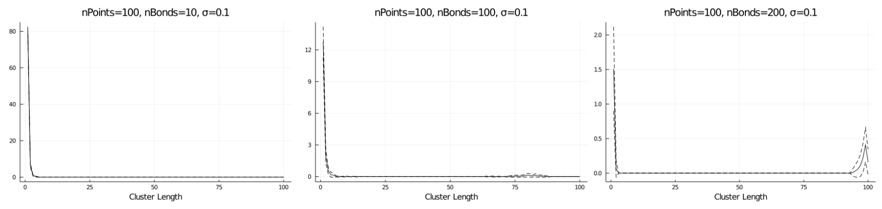

Figure 6: The distribution of the cluster Length for fixed number of nodes $N = 100$ and $\sigma = 0.1$, for different number of bonds among the nodes. The panels are arranged as in the figure 4.

This way, thermodynamic averages are calculated by generating a sequence of M configurations $\{s_1, \dots s_M\}$,

$$\langle A \rangle \approx \frac{1}{M} \sum_{n=1} M A_n. \tag{10}$$

The Ising model is modelled by the Hamiltonian

$$\mathcal{H}_{\text{Ising}} = -J \sum_{\langle ij \rangle} s_i s_j. \tag{11}$$

), where the sum runs over all pairs of neares neighbors, coupled via ferromagnetic coupling with strength $J > 0$. Local trial moves correspond to flipping single spins.

Swendsen and Wang [1987] changes the Monte carlo algorithm from a "single-spin flip". The recipe of this algorithm is as follows:

1. A "bond" is formed between every pair of nearest neighbors,aligned with a probability $p_{ij} = 1 - \exp(-i\beta J)$, with J the coupling constant.
2. Connected bonds (directly or indirectly) belong to the same cluster[1]
3. Spins in each cluster are flipped collectively with probability $\frac{1}{2}$.
4. Delete all bonds, perform step (1) again.

[1]They supposedly have the same spin? From Luijten [2006]: "The bond assignment procedure divides the system into clusters of *parallel* spins (a so-called cluster decomposition) [...] two spins of the same sign need not belong to the same cluster, even if these spins are adjacent on the lattice.

This algorithm suppresses the dynamic slowing down near critical points. Near critical points (continuous phase transitions), the relaxation time of thermodynamic properties depends on the correlation length

$$\tau \propto \xi^z, \tag{12}$$

with $z \approx 2$ the so-called dynamical critical exponent. The correlation length diverges as

$$\xi \propto |T - T_c|^{-\nu}, \tag{13}$$

with $\nu > 0$. As $T \to T_c$, we encounter a *critical slowing down.* Larger systems with larger correlation lengths present larger correlation times, and it therefore becomes increasingly difficult to generate statistically independent configurations.

The above mentioned algorithm destroys nonlocal correlations, and the dynamical exponent z is lowered to a much smaller value.

A Code snippets

A.1 Code for the generation of connection sets

```julia
using AliasTable

function generateConnectionsSet(N, N_l, sigma)

    distancesArray = 1:(N-1)
    chooseAT = AliasTable(ones(nPoints))
    distanceAliasTable = AliasTable(
                    1 ./ distancesArray .^ (1+sigma)
                    )

    # Stores connections
    connectionsSet = Set()

    # Stops if N_l is reached
    while len(connectionsSet) < N_l
        # Uniform distribution
        particle1Choice = rand(chooseAT, 1)[1]

        # Rolls integer m with probability P_m = m^(-(1+sigma))
        particle2Addition = rand(distanceAliasTable, 1)[1]
        particle2Choice = (particle2Addition + particle1Choice) % N

        # Stores the sampled connection. Since sets do not repeat elements
        # and are also unordered, if the connection already existed,
        # it will not be stored.
        push!(connectionsSet,
                Set([particle1Choice, particle2Choice]))
    end
    connectionsSet
end
```

A.2 Code for the clustering algorithms

```julia
function clusterIdentification(connectivityArray)
    nPoints = length(connectivityArray)
    visited = Set{Int}()
    clusters = []

    function dfs(node, cluster)
        stack = [ node ]
        while !isempty(stack)
            curr = pop!(stack)
            if !(curr in visited)
```

```
                push!(visited, curr)
                push!(cluster, curr)
                for neighbor in connectivityArray[curr]
                    if !(neighbor in visited)
                        push!(stack, neighbor)
                    end
                end
            end
        end
        cluster
    end

        for i in 1:nPoints
            if !(i in visited)
                cluster = Set{Int}()
                cluster = dfs(i, cluster)
                append!(clusters, [cluster])
            end
        end
        length(clusters)
end
```

References

F. Beyer, M. Weigel, and M. A. Moore. One-dimensional infinite-component vector spin glass with long-range interactions. *Phys. Rev. B*, 86:014431, Jul 2012. doi: 10.1103/PhysRevB.86.014431. URL `https://link.aps.org/doi/10.1103/PhysRevB.86.014431`.

B. Kamiński, 2022. URL `"https://www.juliabloggers.com/set-vs-vector-lookup-in-julia-a-closer-look/`.

E. Luijten. *Introduction to Cluster Monte Carlo Algorithms*, pages 13–38. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-35273-0. doi: 10.1007/3-540-35273-2_1. URL `https://doi.org/10.1007/3-540-35273-2_1`.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 06 1953. ISSN 0021-9606. doi: 10.1063/1.1699114. URL `https://doi.org/10.1063/1.1699114`.

G. S. Rushbrooke and H. D. Ursell. On one-dimensional regular assemblies. *Mathematical Proceedings of the Cambridge Philosophical Society*, 44(2): 263–271, 1948. doi: 10.1017/S0305004100024221.

R. H. Swendsen and J.-S. Wang. Nonuniversal critical dynamics in monte carlo simulations. *Phys. Rev. Lett.*, 58:86–88, Jan 1987. doi: 10.1103/PhysRevLett.58.86. URL `https://link.aps.org/doi/10.1103/PhysRevLett.58.86`.

A. Walker. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters*, 10:127–128, 1974. doi: 10.1049/el:19740097. URL `https://digital-library.theiet.org/doi/abs/10.1049/el%3A19740097`.