Report for the Course Modelling in Computational Science, HT23

Project 1: The Potts model

Theo Koppenhöfer

Group 4 (Anna Rockstroh, Carmen Lopez)

Lund

September 18, 2023

Introduction

The following report is part of the first project of the Modelling in Computational Science course at Lund University, HT2023. In this project we implemented a Monte-Carlo simulation of the q-state Potts model. In the first part of this report we will describe the Potts model, the Monte-Carlo algorithms and some technicalities regarding the implementation. In the second part we will describe several experiments and the results they yielded. The main source for the report were [1] and [3]. The report and the Python implementation can be found online under [2].

The setup

The Potts model

The following is a brief introduction to the Potts model from statistical mechanics which describes the spin of a particle on a grid. A state of the q-state Potts model of a $L \times L$ flat grid is given by a mapping from the grid points to the spin states

$$s\colon \{1,\dots,L\} \times \{1,\dots,L\} \to \{1,\dots,q\}\,.$$

One assigns this state an energy via

$$E(s) = -J \sum_{\substack{i,j \text{ neighbouring} \\ \text{grid points}}} \delta(s_i, s_j)$$

where δ denotes the Kronecker-delta and $J = 1$ is the coupling strength. As given in the problem setting we assume periodic boundary conditions on the grid. The aim is to evaluate the integral

$$\langle E \rangle = \int E(s) p(s)\,\mathrm{d}s\,. \tag{1}$$

Here p is the density corresponding to the Boltzmann distribution, i.e.

$$p(s) = \frac{\exp(-E(s)/T)}{Z}$$

with Z a normalisation constant and T the temperature. To evaluate the integral in (1) we will use random sampling of s with the samples distributed according to the Boltzmann distribution. In other words, we will use a Monte-Carlo simulation.

The Monte-Carlo algorithms

To determine the statistical behaviour of the Potts model we use Monte-Carlo simulations. The first method we implemented was the Metropolis algorithm. The steps for the Metropolis algorithm are given in figure 1. A single iteration of this algorithm is performed by the function `MC_step_fast` in the implementation. In the implementation we used the pseudo-random number generators from `numpy.random` for the

Input : Initial data s, L, T, q

for $k = 0, 1, \ldots$ **do**
  Pick a point on the grid.
  Propose a new random spin value for this point.
  Calculate the change of energy ΔE that this spin flip would cause.
  Accept this spin flip with probability $\min\{1, \exp(-\Delta E/T)\}$.
end

Algorithm 1: Metropolis

Input : Initial data s, L, T, q

for $k = 0, 1, \ldots$ **do**
  Pick a point on the grid.
  Pick a spin value for this point with a probability given by the distribution

$$p(s_i) = C \exp\left(\frac{1}{T} \sum_{j \text{ neighbours } i} \delta(s_i, s_j)\right).$$

  Here C is a normalisation constant.
end

Algorithm 2: Heat-bath

proposed spins, spin states and for determining if the choice should be accepted. For more information on the Metropolis Monte-Carlo simulation and the Potts model see [1].

Analogously to the Metropolis algorithm the steps of the heat-bath algorithm are given in figure 2. Its step differs from the Metropolis algorithm only after a point on the grid was randomly chosen. In the implementation a single iteration is performed by the function `Gibbs_step`. In the implementation for this algorithm we also used the `numpy.random` functions. For more details on the heat-bath algorithm see e.g. [3].

Some implementation details

Determining when the Energy has plateaued

In order for the simulation to start sampling we needed a criteria to determine the time t_0 when the system reaches an equilibrium state. To determine t_0 we calculate in every step i moving averages ma_1 and ma_2 over n energies. The construction is shown in figure 1. If we start in the hot state then the energy will tend to decrease until we reach an equilibrium. Hence we can use the condition

$$\text{ma}_2 \leq \text{ma}_1 \tag{2}$$

to determine t_0. If on the other hand we start in cold state the energy will in general increase and we have to reverse the inequality in equation (2). After reaching an

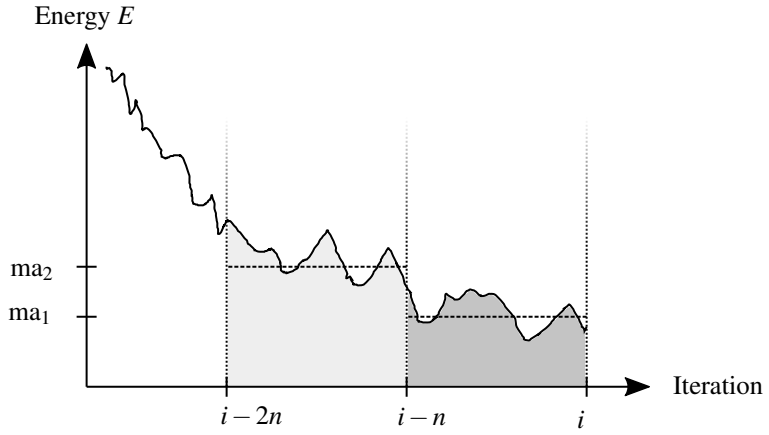

Figure 1: A visualisation of the moving averages.

equilibrium the simulation runs for a further `M_sampling` steps. It is over these samples we take the mean and the standard deviation.

Improving performance

While simulating we run into performance issues due to the fact that Python is in general rather slow even with heavy use of `numpy`. We resolved this issue with the help of `numba` which precompiles functions and thus increased performance substantially. The downside is that the code becomes quite unaesthetic because `numba` does not support all `python` and `numpy` features. In hindsight it would probably been better to have written the iteration in a language other than python. In the experiments we also preferred to use the Metropolis algorithm because our implementation of this algorithm ran faster than the heat-bath algorithm.

The experiments

A brief sanity check

In order to check that our code was indeed doing what it was supposed to we designed some sanity checks. The energy during the simulation is updated with the calculated value for ΔE. Hence we checked with the function `test_energies` if the energy of the end state of the simulation is the same as the energy calculated during the simulation. Indeed, the last time I checked this was the case.

We also ran the simulation for both the Metropolis and the heat-bath algorithms and for a hot start and a cold start and compared the results. In the hot start the state s is randomly initialised and in the cold start s is initialised to have a constant value. We set the parameter $q = 2$ and the grid size to $L = 100$. In a first experiment the temperature was set to $T = 100$. The result for the energies per spin is plotted in figure 2. One can see that for both initialisations and for both algorithms the energy converges to a

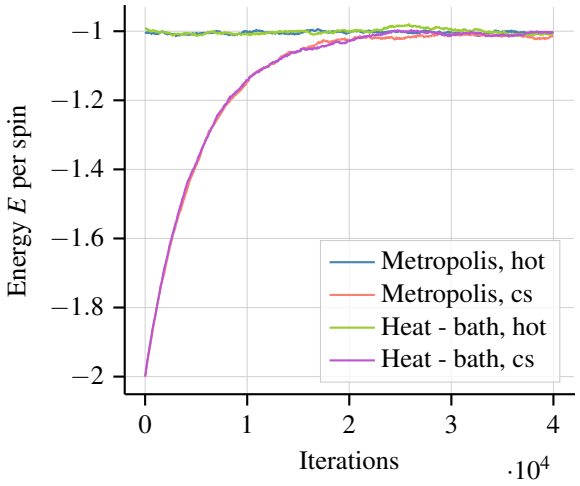

Figure 2: Energy evolution for the temperature $T = 100$

fixed value close to -1. Since the temperature is relatively 'hot' the hot start reaches equilibrium faster than the cold start. The energy per spin value of almost -1 means that two neighbouring points will almost surely differ in their spin as would be expected for hot temperatures. In a second experiment the temperature was set to $T = 0.1$ and the result can be seen in figure 3. Here too the energies per spin start to converge to a fixed lower value close to -2 but for the hot starts this process takes far longer. The energy per spin value of almost -2 means that two given neighbouring points will almost surely agree in their spins. This should be expected for cold temperatures.

State plots

We also plotted the state of the system for a system size $L = 20$ with parameter $q = 5$ after $M = 10^4$ iterations for the temperatures $T = 0.1$, $T = 1$ and $T = 100$. Figure 4 shows the state for the low temperature, figure 5 for the medium temperature and figure 6 for the high temperature. We can see that as the temperature increases the state of the system gradually becomes more chaotic. For low temperatures there are larger patches of the same state. This corresponds to a lower energy configuration. In the implementation one can also see an animated version of the evolution of the system state for the different temperatures.

Distribution of energies in equilibrium

In the following numerical experiment we set the grid size to $L = 50$, the number of spin states to $q = 10$ and the temperature to $T = 1$. We then run the simulation until the system reached equilibrium. After that we varied the number of steps M to be 10^5, 10^6, $4 \cdot 10^6$ and 10^7. The distribution of the energies can be seen in figures 7 to 10. One sees that as M increases the distribution approaches a distribution which looks similar to the

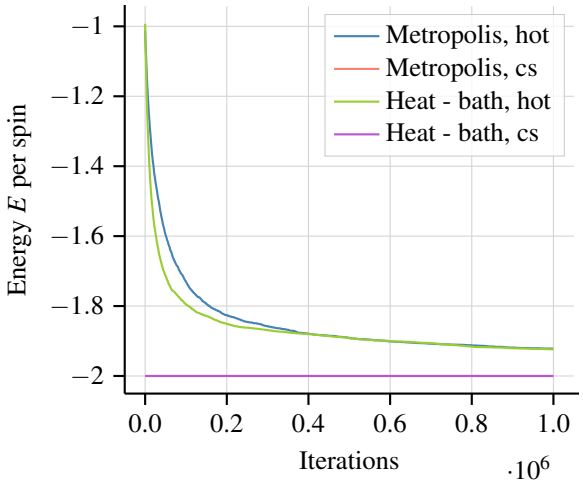

Figure 3: Energy evolution for the temperature $T = 0.1$

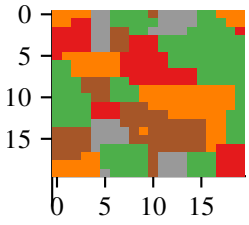

Figure 4: State for temperature $T = 0.1$.

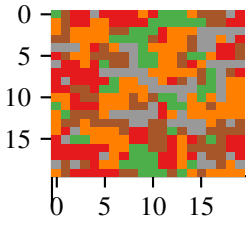

Figure 5: State for temperature $T = 1$.

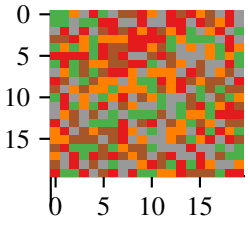

Figure 6: State for temperature $T = 100$.

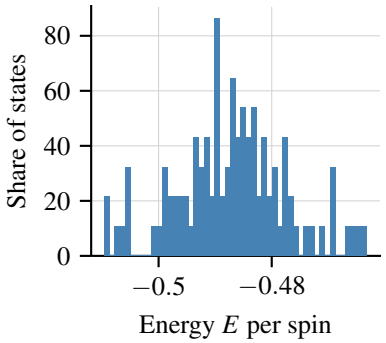

Figure 7: For $M = 10^5$

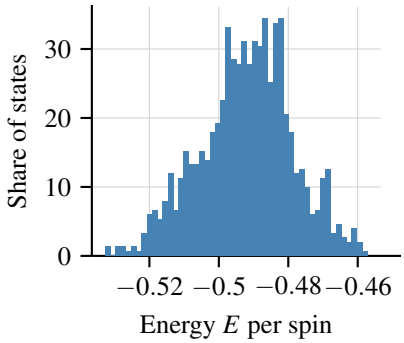

Figure 8: For $M = 10^6$

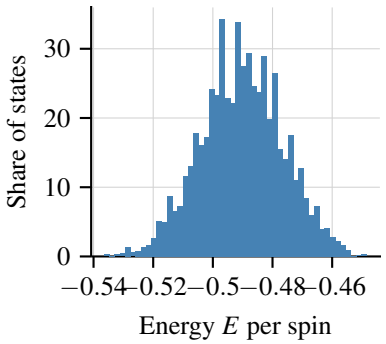

Figure 9: For $M = 4 \cdot 10^6$

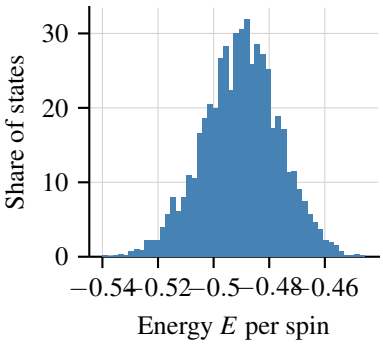

Figure 10: For $M = 10^7$

Maxwell-Boltzmann distribution with a peak at an energy per spin of $E \approx -0.49$.

Energies in dependence of the temperature

In the main experiment we plotted the energies in dependence of the temperature for a system with parameters $q = 2$ and $q = 10$. The aim was to observe how the energy depended on the temperature around the Curie temperature

$$T_c = \frac{1}{\ln\left(1+\sqrt{q}\right)}.$$

For $q = 2$ we have a Curie temperature of $T_c \approx 1.13$ and for $q = 10$ a Curie temperature of $T_c \approx 0.701$. Thus we decided to plot the energies for 40 different temperatures in the range $0.1 \leq T \leq 2$. We also wanted to simulate a system of maximal size. Since a sampling size beyond 10^7 was computationally unreasonable and the temperature would be in the region of $T \approx 1$ the previous experiment suggested a maximal system size of $L = 50$. For larger systems the distribution plot shown in figure 10 would become

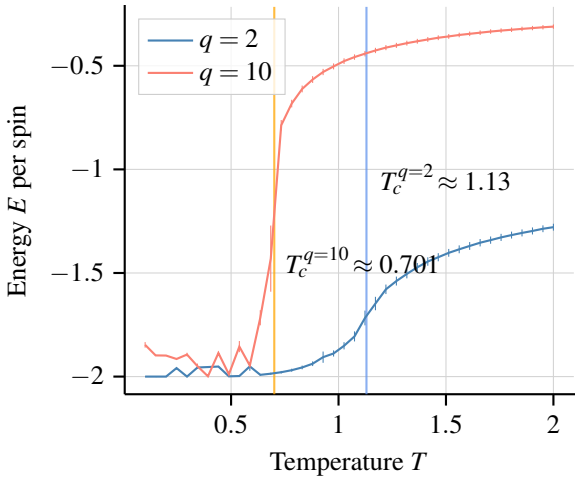

Figure 11

degenerate. So we run the experiment with a sampling size of `M_sampling=10E7` and chose the parameter n for the moving averages to be 10^6. The results of the experiment are shown in figure 11. The error bars in the plot are the standard deviation of the energies per spin. One can see that for the $q = 2$ case the energy gradually increases as the temperature increases around T_c. For $q = 10$ on the other hand the energy dramatically increases around T_c. We also observe that the standard deviation becomes large at the simulation with the value of T closest to T_c. This is consistent with the fact that the phase transition for $q = 10$ on infinite lattice size is discontinuous at T_c.

In figure 12 one sees the number of iterations t_0 it took the simulation to reach an equilibrium. For this it takes the simulation in most cases roughly $0.25 \cdot 10^7$ steps. However, for cold temperatures with $q = 10$ this takes far longer since the system starts in the 'hot' state similar to the state previously depicted in figure 6 and must first reach the 'cold' state.

In another series of experiments we wanted to see how the energy dependence of the temperature depends on the system size L. The results are shown in figures 13 and 14 for system sizes $L = 5$ and $L = 10$ respectively. Both look quite similar to the case $L = 50$. One notices that as the state size L decreases the standard deviation for the energies per spin increases. This is probably due to the fact that in a smaller system a change of some spins and thus a change of energy is has a greater impact in relation to the total energy of the system. It is also noticeable that the curve for the parameter $q = 10$ smooths out as the system size decreases.

Plotting the energy distribution around the Curie temperature

In a final experiment we plotted the distribution of energies around the Curie temperature for a grid size of $L = 50$ and parameter $q = 10$. The number of sampling steps was chosen as `M_sampling=10E7`. In figure 17 we see that the distribution of energies splits up at

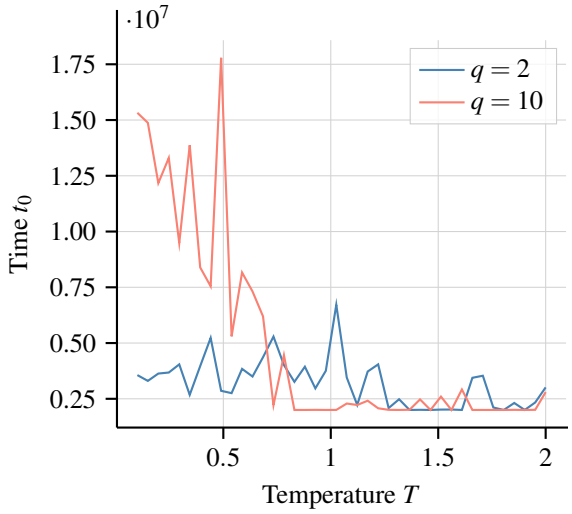

Figure 12: Time t_0 until the equilibrium is reached.

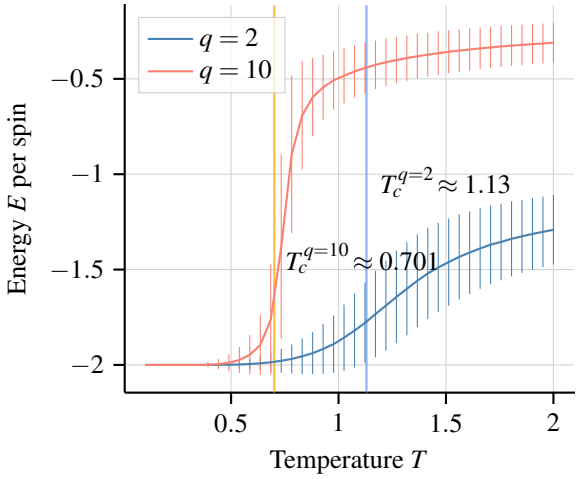

Figure 13: $L = 5$.

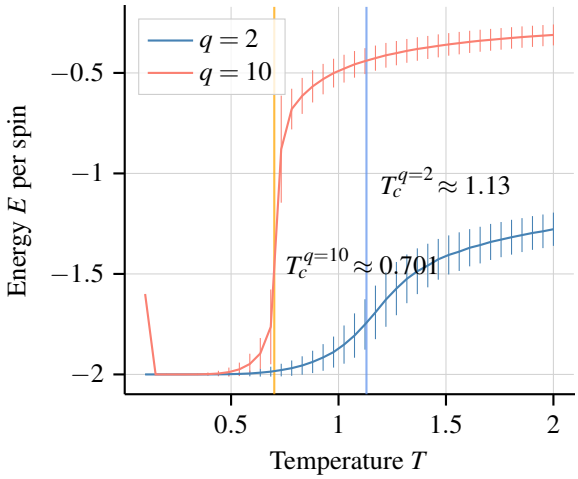

Figure 14: $L = 10$.

the temperature $T = 0.697$. In figures 16 and 18 the distribution lies around the energy -1 for the slightly different temperatures $T = 0.696$ and $T = 0.698$. For the slightly lower temperature $T = 0.695$ in 15 the distribution hovers around the significantly lower energy value of -1.7. This is characteristic behaviour for the instability of the system around the Curie temperature.

Conclusion

We have seen that one can use the Metropolis Monte-Carlo and the heat-bath algorithm to effectively simulate the Potts model. Both algorithms yield consistent results for different initial values and the distribution of energy approximates a distribution similar to the Maxwell-Boltzmann distribution. We also saw that the energy for the parameter $q = 10$ jumped around the Curie temperature. This jump became more pronounced for increased system size. We also saw this jump when plotting the distribution of the energy for a temperature around the Curie temperature. In the case of $q = 2$ the energy on the other hand increased gradually with increasing temperature. We have seen in the project that the number of steps a simulation is able to run for is critical for the ability to produce meaningful results. Here we discovered the efficiency of the code played a crucial role.

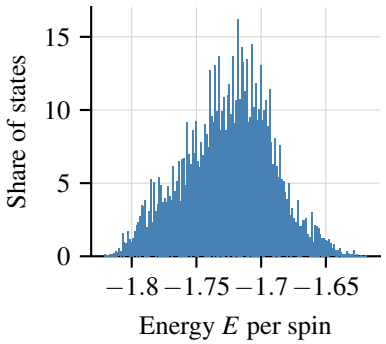

Figure 15: Distribution of energies for the temperature $T = 0.695$.

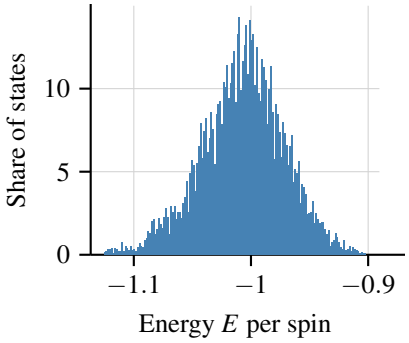

Figure 16: Distribution of energies for the temperature $T = 0.696$.

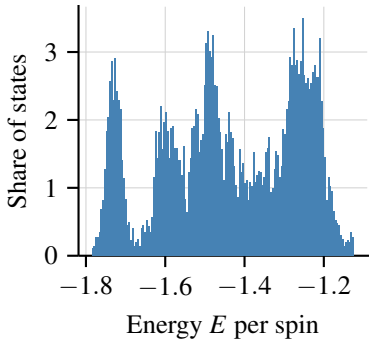

Figure 17: Distribution of energies for the temperature $T = 0.697$.

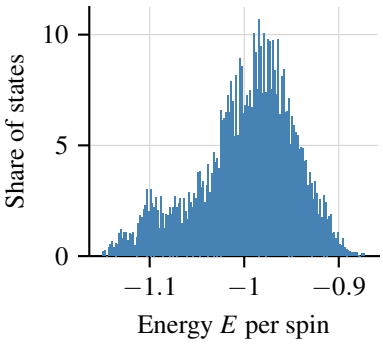

Figure 18: Distribution of energies for the temperature $T = 0.698$.

Bibliography

[1] A. Irsbäck, *Two lectures on random variables and markov chain monte carlo*, BERN01, University of Lund, Aug. 2023.

[2] computational-science-HT23, *Github repository to the project.* Online, 2023. [Online]. Available: `https://github.com/TheoKoppenhoefer/computational-science-HT23`.

[3] A. Irsbäck, *Project: Monte carlo simulation of the q-state potts model*, BERN01, University of Lund, Aug. 2023.