"Critical slowing down" and cluster Monte Carlo methods

The importance of autocorrelation becomes clear when we wish to simulate the Ising model at low temperatures. The mean magnetization $\langle m \rangle$ is zero on any finite system, since there is a degeneracy between the two spin-reversed low-energy configurations.

If, however, we run the single spin-flip Metropolis algorithm at low temperatures and starting from one of the two configurations it will take extremely long time for all spins to be flipped.

This problem appears as soon as we get close to the critical temperature, where it was observed that the autocorrelation time diverges as

$$\tau \propto [\min(\xi, L)]^z$$

with a dynamical critical exponent $z \approx 2$ for all local update methods like the single spin-flip algorithm.

The solution to this problem was found by introducing so-called cluster update techniques, i.e. it is now proposed to flip big clusters of spins and choose them in a clever way so that the probability of flipping these clusters is large.

The Swendsen-Wang "multicluster" algorithm

The multicluster algorithm of Swendsen and Wang (1987) proceeds as follows:

- each bond in the lattice is assigned a label "connected" or "disconnected":

$c = \uparrow\uparrow$	$c = \uparrow\downarrow$	$c = \downarrow\uparrow$	$c = \downarrow\downarrow$
connected	disconnected	disconnected	connected

  - two aligned spins are connected with a probability

$$1 - \exp(-2\beta J)$$

  - two antiparallel spins are never connected

- The connected bonds group the spins into clusters of aligned spins. A cluster labeling algorithm, like the Hoshen-Kopelman algorithm, is used to identify clusters of connected spins.

- Perform measurements ($\rightarrow$ improved estimators)
- Each cluster is flipped with probability $1/2$.
- Go back to step 1.

The Swendsen-Wang algorithm gets less efficient in dimensions higher than two as the majority of the clusters will be very small ones, and only a few large clusters exist.

The Wolff algorithm (single cluster)

The Wolff algorithm is similar to the Swendsen-Wang algorithm, but builds only one cluster starting from a randomly chosen point. As the probability of this point being on a cluster of size s is proportional to s the Wolff algorithm builds preferably larger clusters.

It works as follows:

① • choose a random spin as the initial cluster

② • if a neighboring spin is parallel to the initial spin it will be added to the cluster with probability $1-\exp(-2\beta J)$

• repeat step ② for all points newly added to the cluster and repeat this procedure until no new points can be added.

• perform measurements ($\rightarrow$ improved estimators)

• flip all spins in the cluster

• go to step ①

The linear cluster size diverges with the correlation length ξ and the average number of spins in a cluster is just χT. Thus the algorithm adapts optimally to the physics of the problem and the dynamical critical exponent can be reduced to $z \approx 0$, so that there is no longer any critical slowing down.

Close to criticality these algorithms are many orders of magnitude (a factor L^2) better than the local update methods.

Away from criticality a hybrid method – mixing cluster updates and single spin-flips – can sometimes be the ideal approach.

Ising critical exponents

exponent	Ising $d=2$	Ising $d=3$	Ising $d \geq 4$
β	$1/8$	0.3258 (44)	$1/2$
γ	$7/4$	1.2390 (25)	1
ν	1	0.6294(2)	$1/2$
α	0	0.110(1)	0

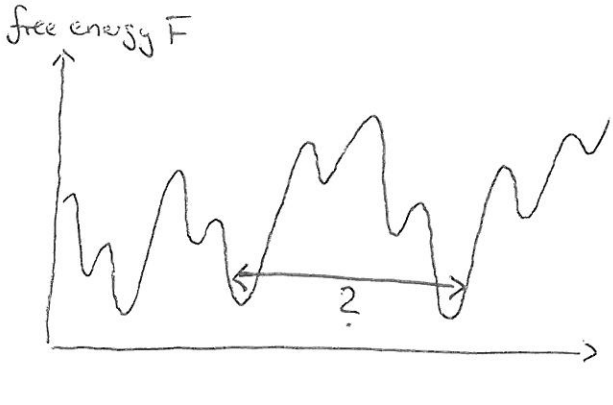
$$T_c = \frac{2}{\ln(1+\sqrt{2})}$$

Heisenberg critical exponents

exponent	$d=3$
β	0.3639(35)
γ	1.3873 (85)
ν	0.7048(30)
α	0.1144 (90)

Extended ensemble simulations

The problem: Thermal equilibrium in systems with rough (free) energy landscape is hugely suppressed.



example systems:
- glasses
- frustrated magnets
- proteins
- any system close to phase transition

The goal: Improve equilibration in numerical simulations
How can this be achieved?

Every Monte Carlo step consists of two parts:

1) suggest a configuration update
   → improvements: non-local updates (clusts, worms, ...)
2) accept/reject this update
   → improvements: extended statistical ensembles

The algorithms:

1) Wang-Landau algorithm
2) Ensemble optimization
3) Parallel tempering

Simulation of Markov chains

Typically, we think of Monte Carlo simulations as sampling configurations in some high-dimensional configuration space

$$C_1 \longrightarrow C_2 \longrightarrow C_3 \longrightarrow \dots \longrightarrow C_i \longrightarrow C_{i+1} \longrightarrow \dots$$

A widely used algorithm to generate such a Markov chain is the Metropolis algorithm:

- propose a (small) change to a configuration

- accept/reject the update with probability

$$p(C_i \to C_{i+1}) = \min\left(1, \frac{\omega(C_{i+1})}{\omega(C_i)}\right)$$

↑ statistical weights

- Common choice for the statistical weights/ensemble is the canonical one:

$$\omega(C_i) = \omega(E_i) = \exp(-\beta E_i)$$

↑ energy $E_i = H(C_i)$.

But we could have chosen all sorts of other weights - we will see some of these other choices later on.

Here:

$$p(C_i \to C_{i+1}) = \min\left(1, \exp(-\beta(E_{i+1} - E_i))\right).$$

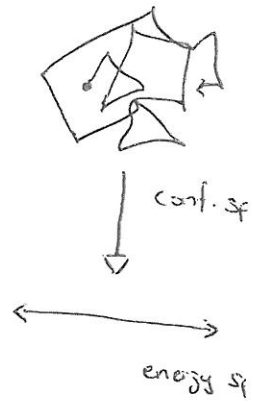
Markov chains (cont'd)

In making our decision about whether or not we accept an update solely based on the energy difference of the two configurations, we neglect a lot of information.
In particular, we <u>project</u> the random walk in configuration space onto one in energy space:

$$C_1 \to C_2 \to \dots \to C_i \to C_{i+1} \to \dots$$

$$\downarrow$$

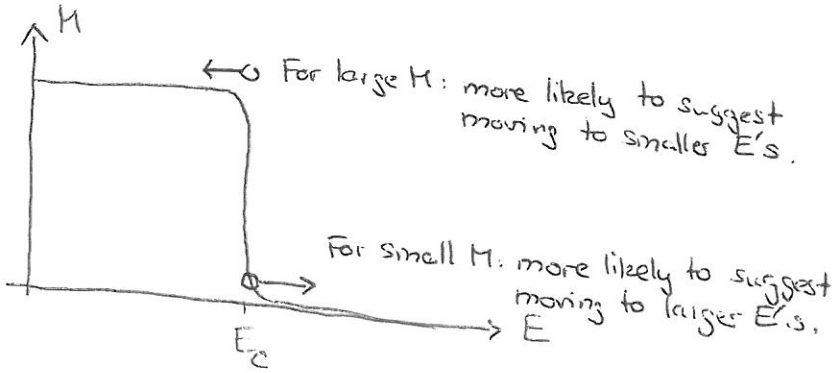
$$E_1 \to E_2 \to \dots \to E_i \to E_{i+1} \to \dots$$



Configuration space: Random walk is biased (by stat ensemble) but Markovian.

Energy space: Random walk is non-Markovian, 'memory' is stored in configuration space.

Example: Ising model



The Wang-Landau algorithm

Goal: Calculate density of states, $g(E)$, by iteratively approximating weights, $\omega(E) = 1/g(E)$, and sample a flat histogram, $h(E)$.

The algorithm:

- Initialize $h(E) = 0$ and $g(E) = 1$ for all energies E.
- Set 'modification factor' $f = f_0 = e$.
- Start from a random configuration C_1.

main loop:

- Propose configuration update $C_1 \to C_2$ (e.g. a spin flip) with corresponding energies $E_1 = H(C_1)$, $E_2 = H(C_2)$.
- Accept update with probability

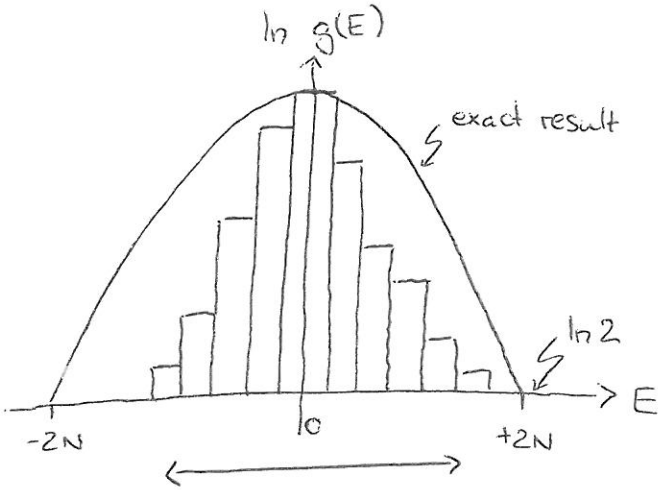
$$p(C_1 \to C_2) = \min\left(1, \frac{g(E_1)}{g(E_2)}\right),$$

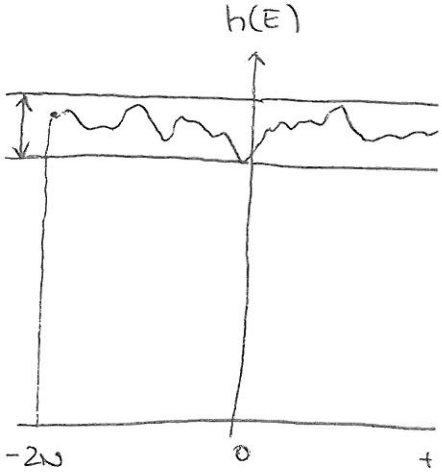
  if accepted $E_1 \leftarrow E_2$, $C_1 \leftarrow C_2$.
- Update density of states, $g(E_1) \leftarrow g(E_1) \cdot f$ and histogram, $h(E_1) \leftarrow h(E_1) + 1$.
- If histogram is "flat", update $f \leftarrow \sqrt{f}$, and reset $h(E) = 0$ for all energies.
- Iterate until f becomes smaller than some threshold, e.g. $f_{final} = 1.000\,001$.

The Wang-Landau algorithm (cont'd)

Example: 2D Ising model on a square lattice with N sites.

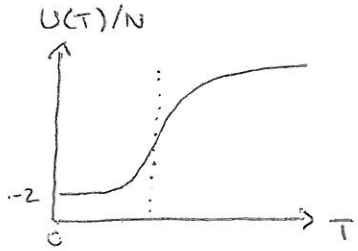
Wang-Landau iterations

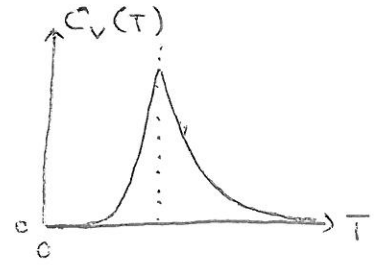


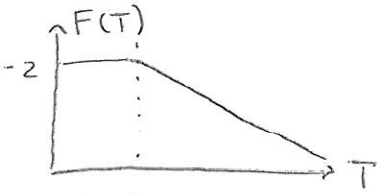


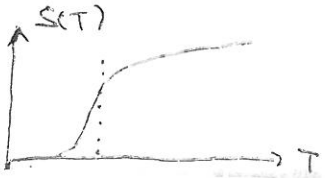
With estimated density of states calculate:

- $Z = \sum_E g(E) \cdot e^{-\beta E}$
- $U(T) = \frac{1}{Z} \cdot \sum_E E \cdot g(E)\, e^{-\beta E} = \langle E \rangle_T$
- $C_v(T) = \frac{\partial U}{\partial T} = \frac{\langle E^2 \rangle_T - \langle E \rangle_T^2}{T^2}$
- $F(T) = -kT \cdot \log Z$
- $S(T) = \frac{U(T) - F(T)}{T}$









Limitations of flat-histogram sampling

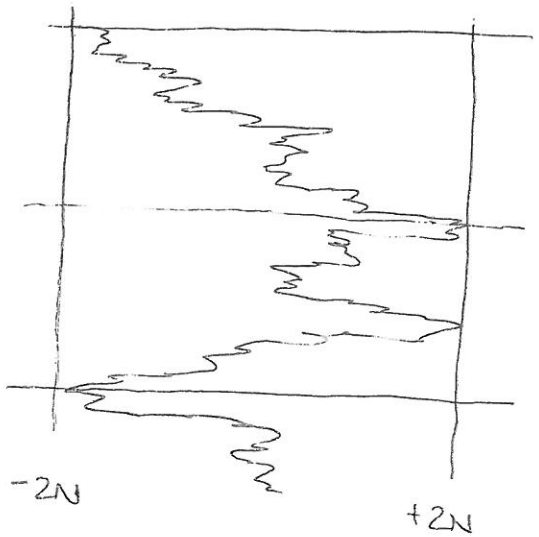
Is sampling a flat histogram really the best we can do to explore a system with a rough energy landscape?

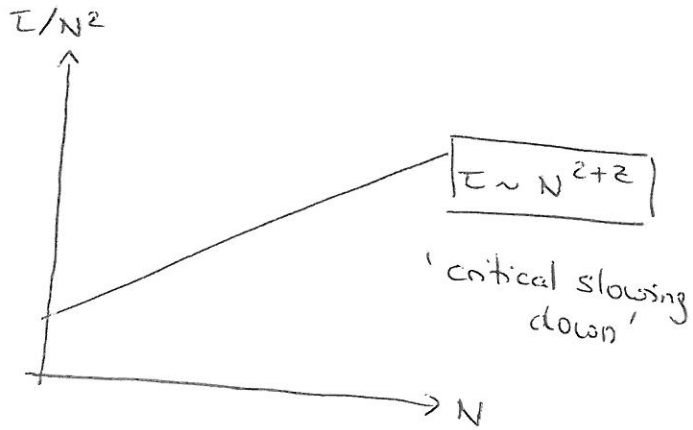
No - one reason we have just seen: Not all temperature / energy regimes are the same. So, why should this not be reflected in the histogram / statistical ensemble?

A more complex answer: Carefully study the random walk in energy (temperature) space as opposed to the random walk in configuration space.

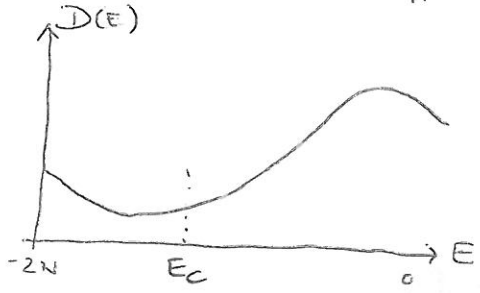
Something we have already seen: Markovian vs. non-Markovian behavior.

A further manifestation of the non-Markovian behavior is found in the scaling behavior of round-trip times.





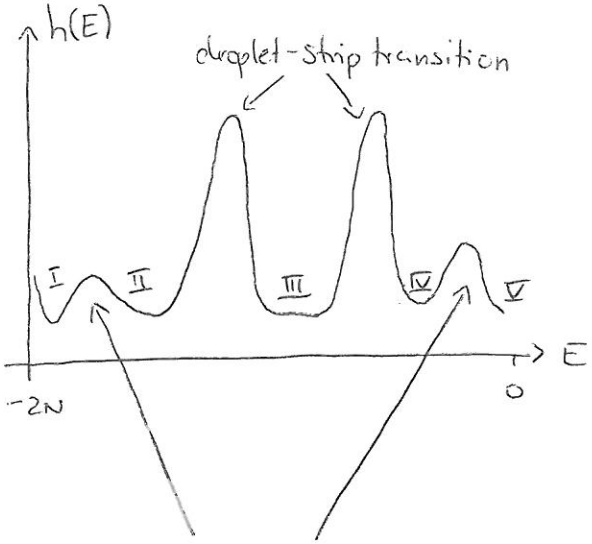
What originates this slowing down? A (local) diffusivity that is modulated in (energy) space.

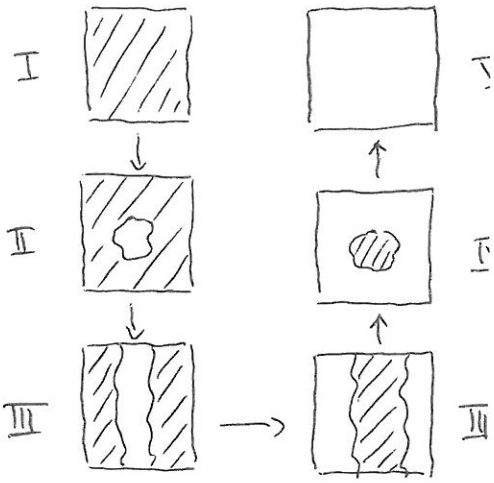


Ensemble optimization (cont'd)

Example: 2D Potts model (first-order transition)

$$\mathcal{H} = -\sum_{\langle ij \rangle} \delta(\sigma_i, \sigma_j) \qquad \sigma_i \in \{1, 2, \ldots, Q\}$$



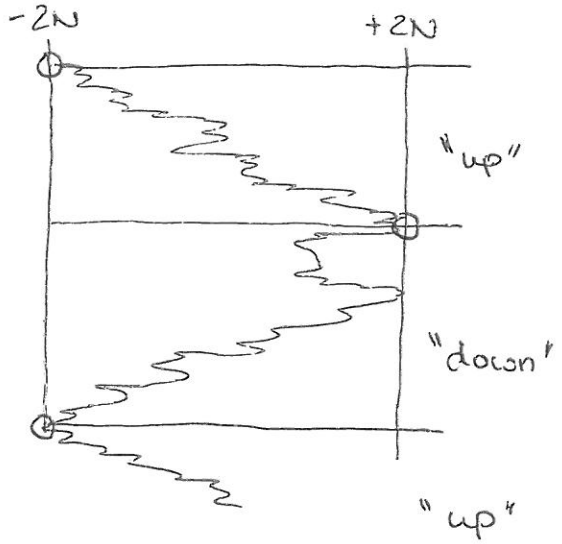


Potts model

$$T^* = \frac{1}{\ln(1+\sqrt{Q})}$$

Ensemble optimization

Can we further improve the simulated statistical ensemble? To achieve this we want to minimize the round-trip times between extremal energies.

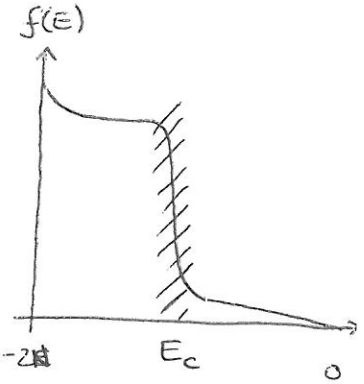


Split random walk into two:
steady-state currents from $E_{min} \to E_{max}$ and $E_{max} \to E_{min}$ which exactly cancel (since we are in total equilibrium)

These steady-state currents can be estimated

$$j = \mathcal{D}(E) \cdot h_\omega(E) \cdot \frac{df}{dE}$$

$\mathcal{D}(E)$: local diffusivity; $h_\omega(E)$: histogram; $\frac{df}{dE}$: derivative of fraction $f(E) = \frac{h^+_\omega(E)}{h_\omega(E)}$



Minimize round-trip time by maximizing this current:

$$\tau \propto \frac{1}{j} = \int_{E_{min}}^{E_{max}} dE \left(\frac{1}{\mathcal{D}(E) \cdot h_\omega(E)} + \lambda h_\omega(E) \right)$$

Lagrange multiplier
$h_\omega(E)$ must remain normalized

The 'optimal' histogram thus becomes:

$$h_\omega(E) \propto \frac{1}{\sqrt{\mathcal{D}(E)}} \propto \frac{df}{dE}$$

Ensemble optimization (cont'd)

Feedback algorithm:

- Start with some trial weights $\omega(E)$, e.g. from Wang-Landau algorithm.

→ - Initialize $h(E) = 0$, $h^+(E) = 0$, $h^-(E) = 0$

- Run simulations with acceptance rates

$$p(C_1 \to C_2') = \min\left(1, \frac{\omega(C_2')}{\omega(C_1)}\right) = \min\left(1, \frac{\omega(E_2')}{\omega(E_1)}\right)$$

and record the histograms $h(E)$, $h^+(E)$ and $h^-(E)$.

- Estimate local diffusivity

$$\mathcal{D}(E) \propto \frac{1}{h(E)\frac{df}{dE}} \quad , \text{ where } f(E) = \frac{h^+(E)}{h(E)}$$

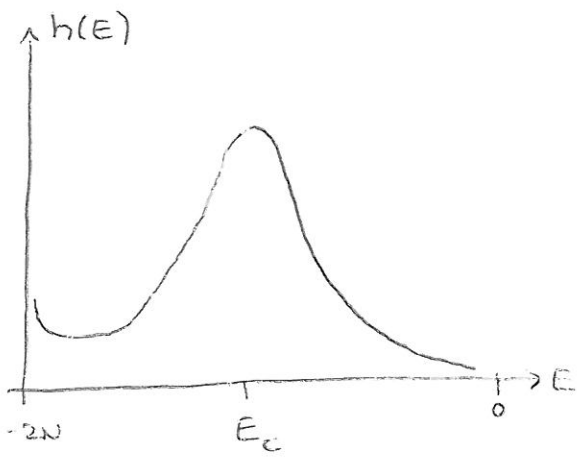
- Optimize weights as

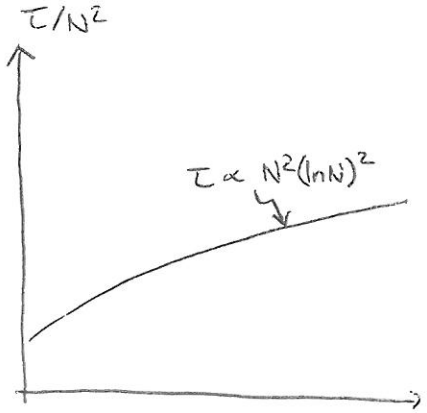
$$\omega(E) \leftarrow \omega(E) \cdot \sqrt{\frac{1}{h(E)}\frac{df}{dE}}$$

main loop

- Continue feedback until weights converge

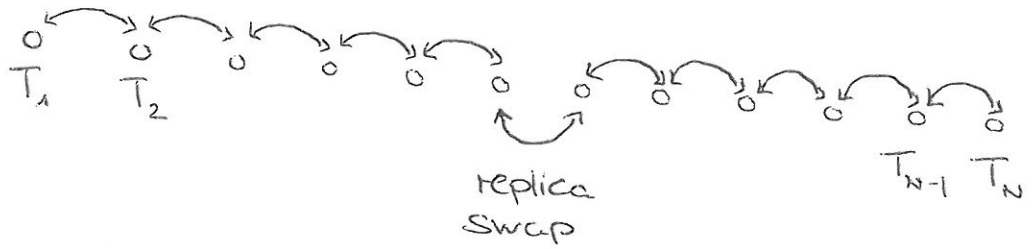
Example: 2D Ising model





The parallel tempering algorithm

Goal: Improve thermal equilibration by simulating multiple replicas of the system at various temperatures and swapping replicas between adjacent temperatures.



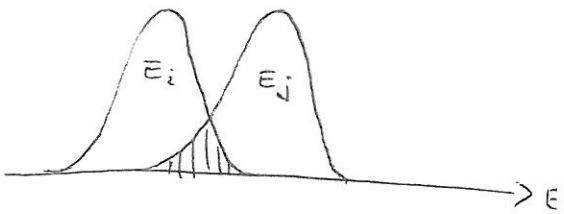
temperature ⟶

Accept replica swap with probability

$$p(E_i, T_i \rightarrow E_j, T_j) = \min\left(1, \exp(\Delta\beta\, \Delta E)\right)$$

$$\Delta\beta = \beta_j - \beta_i = \frac{1}{T_j} - \frac{1}{T_i}$$

$$\Delta E = E_j - E_i$$



From perspective of given temperature:
Replica swaps (may) result in global configuration update.

From perspective of given replica:
Replica swaps result in random walk in T-space.

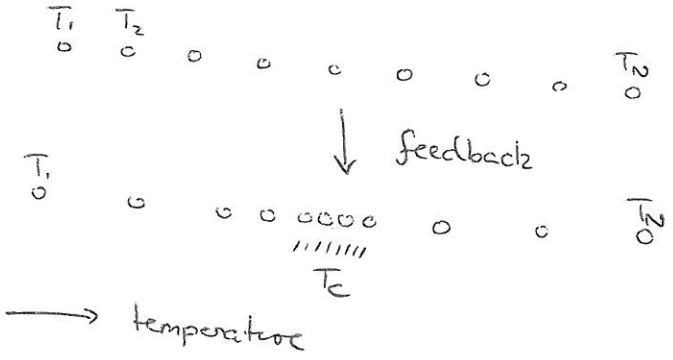
Parallel tempering (cont'd)

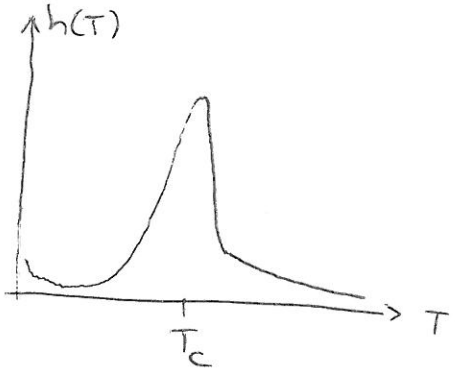
How do you choose the set of temperature points?

Again, we want to shift (numerical) resources towards the bottleneck of the simulation, i.e. those temperature regimes where the (local) diffusivity (in temperature space) is suppressed.

$$h(T) \propto \frac{1}{\sqrt{D(T)}} \quad , \text{where } h(T) \propto \frac{1}{\Delta T}$$

Example: 2D Ising model





Note: This is <u>not</u> the same as a uniform acceptance rates for swap moves.