Basic Monte Carlo (chapter 3)

Algorithm

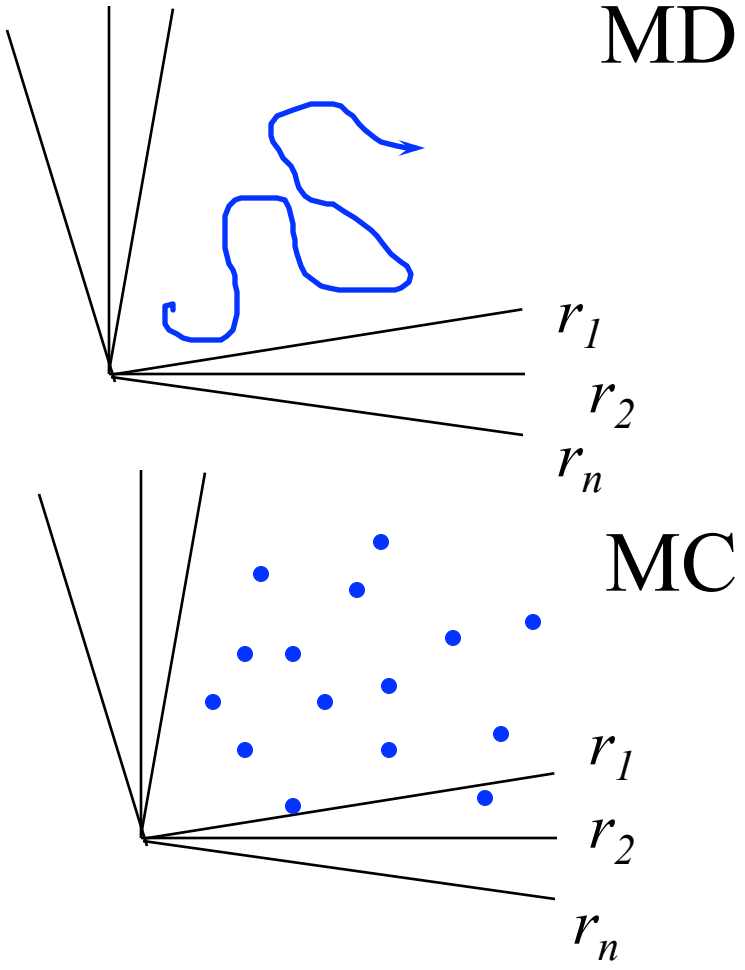
Detailed Balance

Other points

Molecular Simulations

- Molecular dynamics: solve equations of motion

- Monte Carlo: importance sampling



Statistical Thermodynamics

Partition function

$$Q_{NVT} = \frac{1}{\Lambda^{3N} N!} \int \mathrm{d}\mathrm{r}^N \exp\left[-\beta U\left(\mathrm{r}^N\right)\right]$$

Ensemble average

$$\left\langle A \right\rangle_{NVT} = \frac{1}{Q_{NVT}} \frac{1}{\Lambda^{3N} N!} \int \mathrm{d}\mathrm{r}^N A\left(\mathrm{r}^N\right) \exp\left[-\beta U\left(\mathrm{r}^N\right)\right]$$

Probability to find a particular configuration

$$\mathrm{N}\left(\mathrm{r}^N\right) = \frac{1}{Q_{NVT}} \frac{1}{\Lambda^{3N} N!} \int \mathrm{d}\mathrm{r}'^N \delta\left(\mathrm{r}'^N - \mathrm{r}^N\right) \exp\left[-\beta U\left(\mathrm{r}'^N\right)\right] \propto \exp\left[-\beta U\left(\mathrm{r}^N\right)\right]$$

Free energy

$$\beta F = -\ln\left(Q_{NVT}\right)$$

Monte Carlo

Aim : compute thermal averages (to obtain thermodynamics)

$$\langle A \rangle = \frac{\sum_i \exp(-\beta \epsilon_i) A_i}{\sum_i \exp(-\beta \epsilon_i)}$$

$$A_i = \langle i | A | i \rangle$$

Where i labels all eigenstates of the system, and

Phase space integral

In the classical limit we replace the sum over quantum states by an integral over phase space

$$\langle A \rangle = \frac{\int dp^N dr^N A(p^N, r^N) \exp(-\beta \mathcal{H}(p^N, r^N))}{\int dp^N dr^N \exp(-\beta \mathcal{H}(p^N, r^N))}$$

r is position vector and p is momentum vector.

N is number of particles

H is the Hamiltonian of the system

and $\beta = 1/k_B T$

In replacing the sum by an integral, we have attributed a "volume" h^{3N} to every quantum state

Hamiltonian is sum of potential and kinetic energy

$$H(p^N, r^N) = U(r^N) + \sum_i \frac{p_i^2}{2m_i}$$

If A does not depend on momenta the integration of momenta can be done analytically (Gaussian integral)

$$\langle A \rangle = \frac{\int dr^N A(r^N) \exp(-\beta \mathcal{U}(r^N))}{\int dr^N \exp(-\beta \mathcal{U}(r^N))}$$

Problems with this integral

- We cannot compute the sum over all quantum states because there are so many
- And we cannot compute the classical integral either (except the integration over momenta).
- Consider “normal” numerical integration of 100 particles, 3 dimensions, 10 points in every direction.
- Requires 10^{300} points for a very poor estimate of the integral…

Daan Frenkel's analogy

A similar but much less serious problem:

Measure the depth of the Nile by quadrature.

Not very efficient…

© D. Frenkel

The solution

A better strategy:

measure depth while walking around in the Nile

Importance sampling

© D. Frenkel

Importance sampling

We wish to perform a **random walk** in configuration space, such that the number of times that each point is visited, is proportional to its Boltzmann weight

$$\mathcal{N}(r^N) = c \exp[-\beta U(r^N)]$$

Then

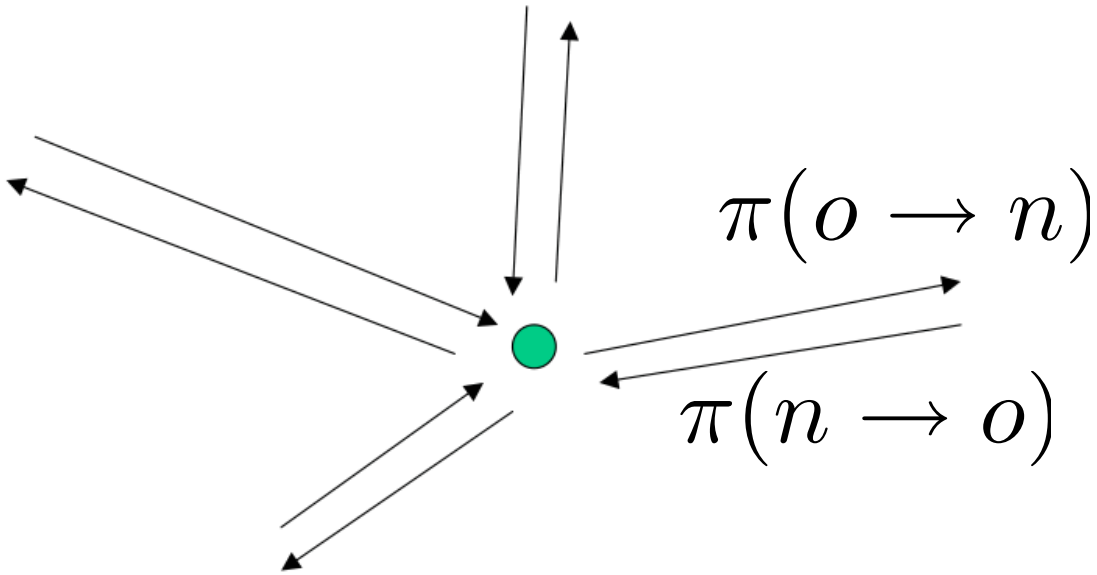
$$\langle A \rangle \approx \frac{1}{L} \sum_{1=1}^{L} A(r_i^N)$$

how to achieve such a random walk?

Balance

Whatever our rule is for moving from one point to another, it should not destroy the equilibrium distribution.

That is: in equilibrium we must have balance



Detailed balance

denoting ***o*** old configuration and ***n*** the new state reachable from ***o*** this balance condition is

$$\mathcal{N}(o)\sum_n \pi(o \to n) = \sum_n \mathcal{N}(n)\pi(n \to o)$$

A stronger condition is

$$\mathcal{N}(o)\pi(o \to n) = \mathcal{N}(n)\pi(n \to o)$$

for each pair of states o, n

Detailed balance (automatically implies balance)

Importance Sampling Random Walk

A move starting from one point consist of generating a **trial move** and accept or reject such a move.

Transition probabilities are a product of the **generation probability** and the **acceptance probability**

$$\pi(o \to n) = \alpha(o \to n) \times \mathrm{acc}(o \to n)$$

Detailed balance implies

$$\mathcal{N}(o)\alpha(o \to n) \times \mathrm{acc}(o \to n) =$$
$$\mathcal{N}(n)\alpha(n \to o) \times \mathrm{acc}(n \to o)$$

Metropolis algortihm

Generation probabilities are often chosen symmetric.

$$\alpha(o \to n) = \alpha(n \to o)$$

Therefore

$$\mathcal{N}(o) \times \mathrm{acc}(o \to n) = \mathcal{N}(n) \times \mathrm{acc}(n \to o)$$

$$\frac{\mathrm{acc}(o \to n)}{\mathrm{acc}(n \to o)} = \frac{\mathcal{N}(n)}{\mathcal{N}(o)} = e^{-\beta[U(n)-U(o)]}$$

the choice of Metropolis, Rosenbluth, Rosenbluth,Teller and Teller (1953)

$$\mathrm{acc}(o \to n) = \min\left(1, e^{-\beta[U(n)-U(o)]}\right)$$

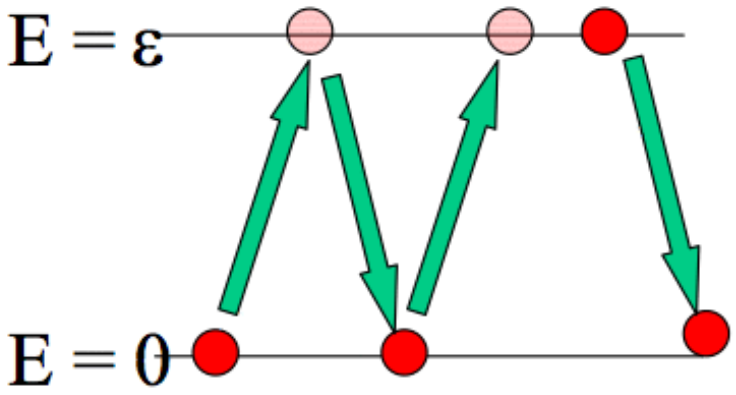
Importance Sampling Random Walk

A move starting from one point consist of generating a **trial move** and **accept or reject** such a move.

$$\mathrm{acc}(o \to n) = \min\left(1, e^{-\beta[U(n)-U(o)]}\right)$$

Metropolis, Rosenbluth, Rosenbluth,Teller and Teller (1953)

- try to change energy state
- compute $\Delta E = E_{new} - E_{old}$
- accept new state if ran < exp(ΔE/kT)
- reject otherwise
- sample the state of the system
- repeat



Does MC give an ensemble average?

$$\langle A \rangle_{NVT} = \frac{1}{Q_{NVT}} \frac{1}{\Lambda^{3N} N!} \int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N) \exp\left[-\beta U(\mathbf{r}^N)\right]$$

$$= \int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N) P(\mathbf{r}^N) = \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N) P(\mathbf{r}^N)}{\int \mathrm{d}\mathbf{r}^N P(\mathbf{r}^N)}$$

$$= \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N) C \exp\left[-\beta U(\mathbf{r}^N)\right]}{\int \mathrm{d}\mathbf{r}^N C \exp\left[-\beta U(\mathbf{r}^N)\right]}$$

$$P(\mathbf{r}^N) = \frac{\exp\left[-\beta U(\mathbf{r}^N)\right]}{Q_{NVT} \Lambda^{3N} N!}$$

$$= \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N) \exp\left[-\beta U(\mathbf{r}^N)\right]}{\int \mathrm{d}\mathbf{r}^N \exp\left[-\beta U(\mathbf{r}^N)\right]}$$

Generate configuration using MC:

$$\left\{\mathbf{r}_1^N, \mathbf{r}_2^N, \mathbf{r}_3^N, \mathbf{r}_4^N \cdots, \mathbf{r}_M^N\right\}$$

$$\bar{A} = \frac{1}{M} \sum_{i=1}^{M} A(\mathbf{r}_i^N) = \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N) P^{MC}(\mathbf{r}^N)}{\int \mathrm{d}\mathbf{r}^N P^{MC}(\mathbf{r}^N)}$$

with

$$P^{MC}(\mathbf{r}^N) = C^{MC} \exp\left[-\beta U(\mathbf{r}^N)\right]$$

$$= \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N) C^{MC} \exp\left[-\beta U(\mathbf{r}^N)\right]}{\int \mathrm{d}\mathbf{r}^N C^{MC} \exp\left[-\beta U(\mathbf{r}^N)\right]}$$

$$= \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N) \exp\left[-\beta U(\mathbf{r}^N)\right]}{\int \mathrm{d}\mathbf{r}^N \exp\left[-\beta U(\mathbf{r}^N)\right]}$$

MC in practice

Algorithm 1 (Basic Metropolis Algorithm)

```
PROGRAM mc                                  basic Metropolis algorithm

do icycl=1,ncycl                            perform ncycl MC cycles
   call rcmove                              displace a particle
   if (mod(icycl,nsamp).eq.0)
+     call sample                           sample averages
enddo
end
```

Comments to this algorithm:

1. *Subroutine* `rcmove` *attempts to displace a randomly selected particle (see Algorithm 2).*
2. *Subroutine* `sample` *samples quantities every* `nsamp`*th cycle.*

Algorithm 2 (Attempt to Displace a Particle)

```
SUBROUTINE nomove                    attempts to displace a particle

o=int(ranf()*npart)+1                select a particle at random
call ener(x(o),eno)                  energy old configuration
xn=x(o)+(ranf()-0.5)*delx            give particle random displacement
call ener(xn,enn)                    energy new configuration
if (ranf().lt.exp(-beta              acceptance rule (3.2.1)
+    *(enn-eno)) x(o)=xn             accepted: replace x(o) by xn
return
end
```

$$\mathrm{acc}(o \to n) = \min\left(1, e^{-\beta[U(n)-U(o)]}\right)$$

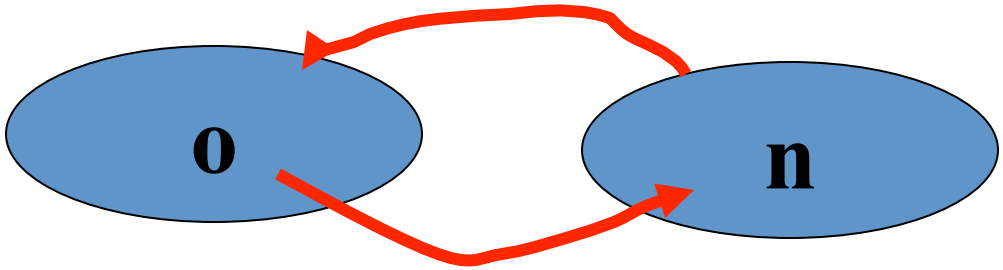
Comments to this algorithm:

1. *Subroutine* `ener` *calculates the energy of a particle at the given position.*
2. *Note that, if a configuration is rejected, the old configuration is retained.*
3. *The* `ranf()` *is a random number uniform in* $[0, 1]$.

Questions

- How can we prove that this scheme generates the desired distribution of configurations?
- Why make a random selection of the particle to be displaced?
- Why do we need to take the old configuration again?
- How large should we take: `delx`?

Detailed balance



$$K(o \to n) = K(n \to o)$$

$$K(o \to n) = N(o) \times \pi(o \to n)$$

$$K(o \to n) = N(o) \times \alpha(o \to n) \times \mathrm{acc}(o \to n)$$

$$K(n \to o) = N(n) \times \alpha(n \to o) \times \mathrm{acc}(n \to o)$$

$$\frac{\mathrm{acc}(o \to n)}{\mathrm{acc}(n \to o)} = \frac{N(n) \times \alpha(n \to o)}{N(o) \times \alpha(o \to n)} = \frac{N(n)}{N(o)}$$

NVT-ensemble

$$N(n) \propto \exp\left[-\beta U\left(n\right)\right]$$

$$\frac{\mathrm{acc}(o \rightarrow n)}{\mathrm{acc}(n \rightarrow o)} = \frac{N(n)}{N(o)}$$

$$\frac{\mathrm{acc}(o \rightarrow n)}{\mathrm{acc}(n \rightarrow o)} = \exp\left[-\beta\left[U\left(n\right) - U\left(o\right)\right]\right]$$

Questions

- How can we prove that this scheme generates the desired distribution of configurations?
- Why make a random selection of the particle to be displaced?
- Why do we need to take the old configuration again?
- How large should we take: `delx`?

Algorithm 2 (Attempt to Displace a Particle)

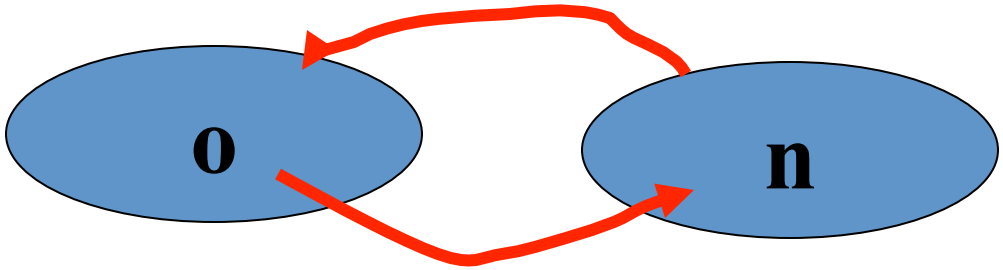
```
SUBROUTINE nomove                    attempts to displace a particle

o=int(ranf()*npart)+1                select a particle at random
call ener(x(o),eno)                  energy old configuration
xn=x(o)+(ranf()-0.5)*delx            give particle random displacement
call ener(xn,enn)                    energy new configuration
if (ranf().lt.exp(-beta              acceptance rule (3.2.1)
+    *(enn-eno)) x(o)=xn             accepted: replace x(o) by xn
return
end
```

Comments to this algorithm:

1. *Subroutine* `ener` *calculates the energy of a particle at the given position.*
2. *Note that, if a configuration is rejected, the old configuration is retained.*
3. *The* `ranf()` *is a random number uniform in* [0, 1].

Detailed balance



$$K(o \to n) = K(n \to o)$$

$$K(o \to n) = N(o) \times \alpha(o \to n) \times \mathrm{acc}(o \to n)$$

$$K(n \to o) = N(n) \times \alpha(n \to o) \times \mathrm{acc}(n \to o)$$

$$\frac{\mathrm{acc}(o \to n)}{\mathrm{acc}(n \to o)} = \frac{N(n) \times \alpha(n \to o)}{N(o) \times \alpha(o \to n)} = \frac{N(n)}{N(o)}$$

$$\alpha(o \to n) = \alpha(n \to o)$$

Hence, we have assumed backward move is equally likely to be generated as forward move : RANDOM selection!

Questions

- How can we prove that this scheme generates the desired distribution of configurations?
- Why make a random selection of the particle to be displaced?
- Why do we need to take the old configuration again?
- How large should we take: `delx`?

Mathematical reason

Transition probability:

$$\pi(o \to n) = \alpha(o \to n) \times \mathrm{acc}(o \to n)$$

$$\sum_n \pi(o \to n) = 1$$

Probability to accept the old configuration:

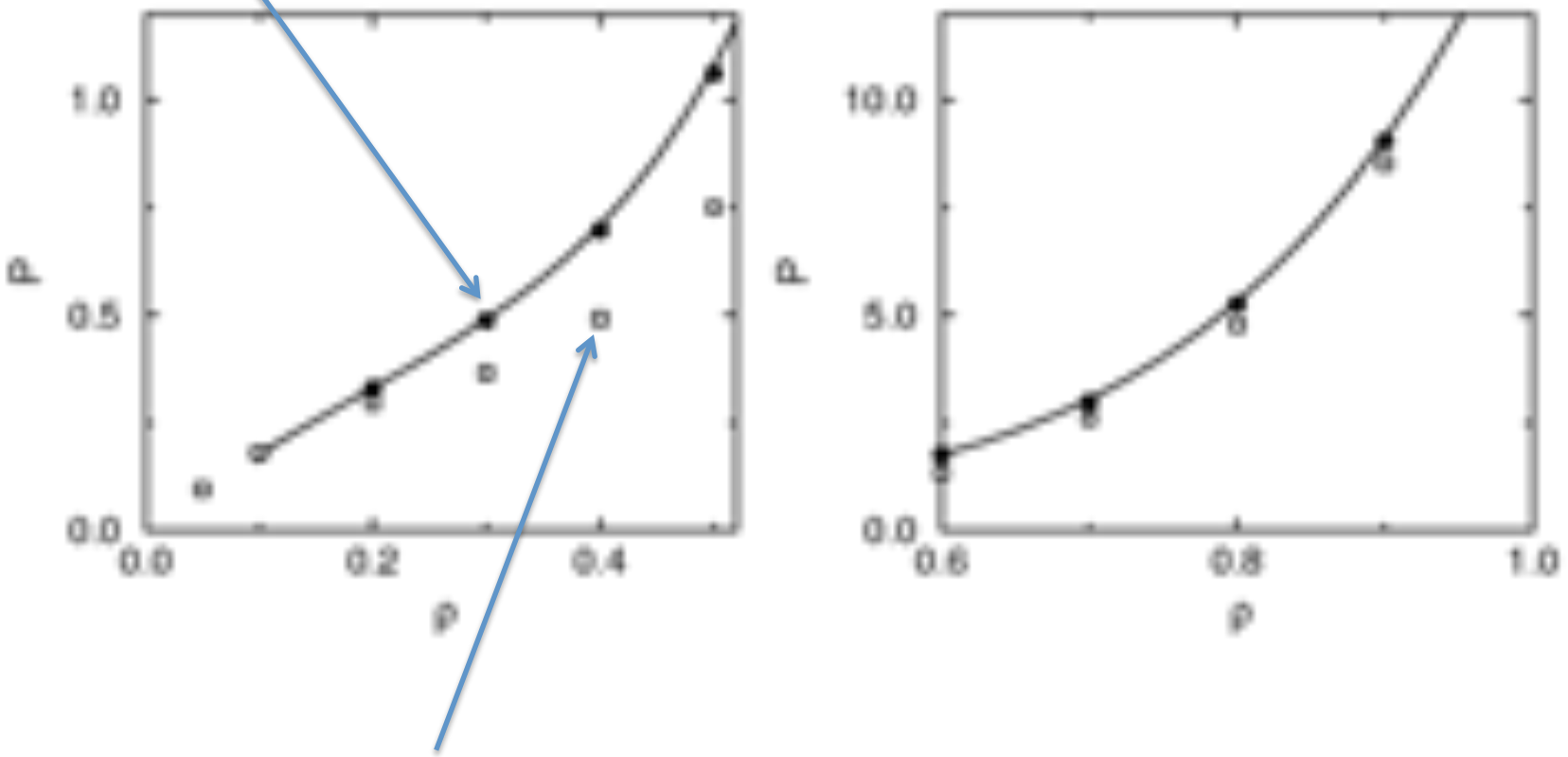
$$\pi(o \to o) = 1 - \sum_{n \neq o} \pi(o \to n)$$

≠0

Keeping old configuration?

Recounting old configuration

Lennard Jones equation of state

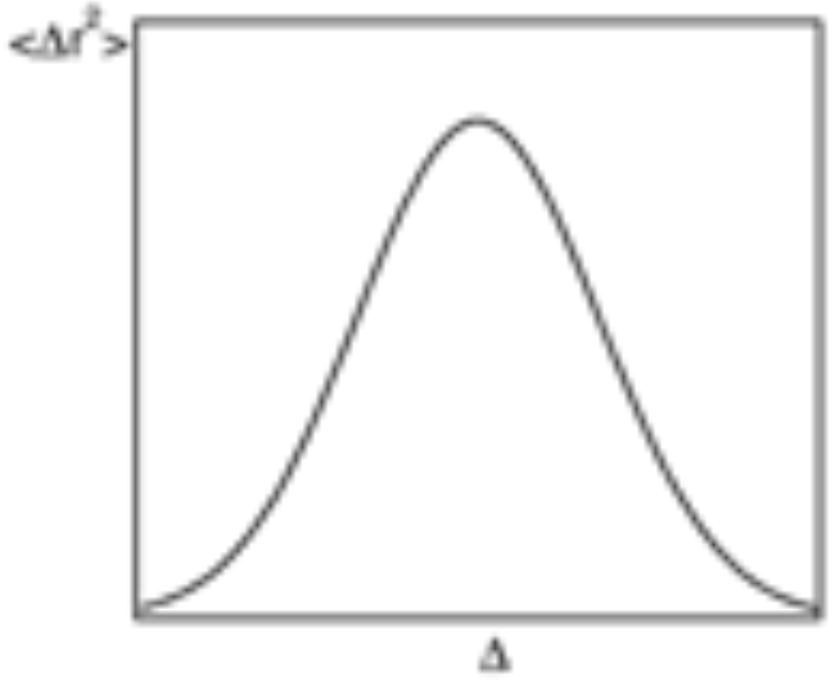


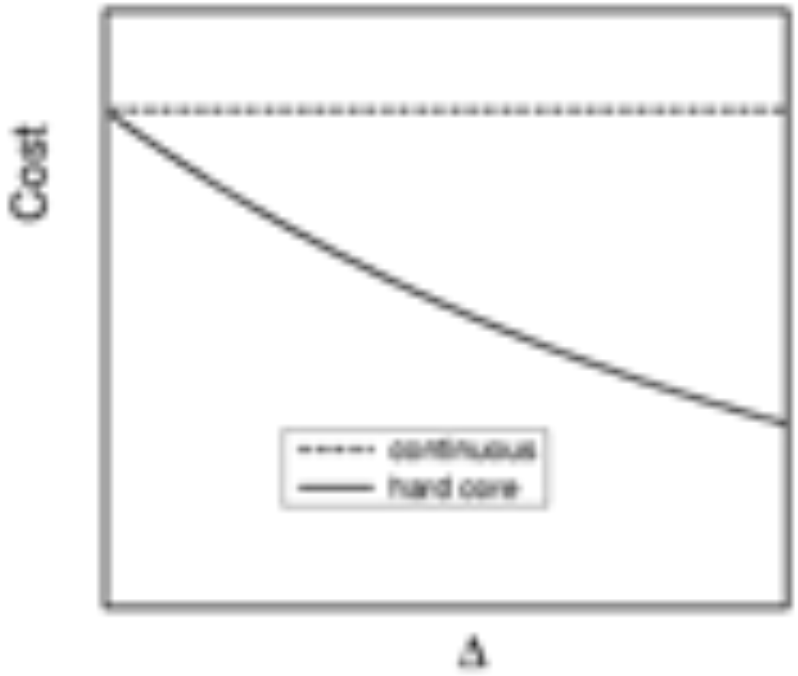
Not recounting old configuration

Questions

- How can we prove that this scheme generates the desired distribution of configurations?
- Why make a random selection of the particle to be displaced?
- Why do we need to take the old configuration again?
- How large should we take: `delx`?

Not too small, not too big!



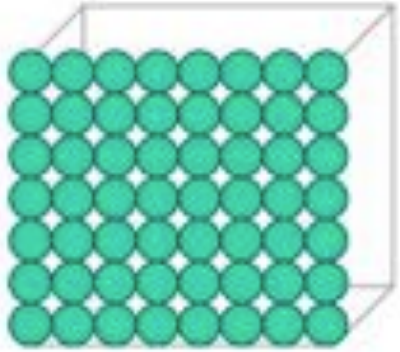


Acceptance probability is usually optimal around 0.5, but can be smaller e.g. for hard cores around 0.2

Practical issues

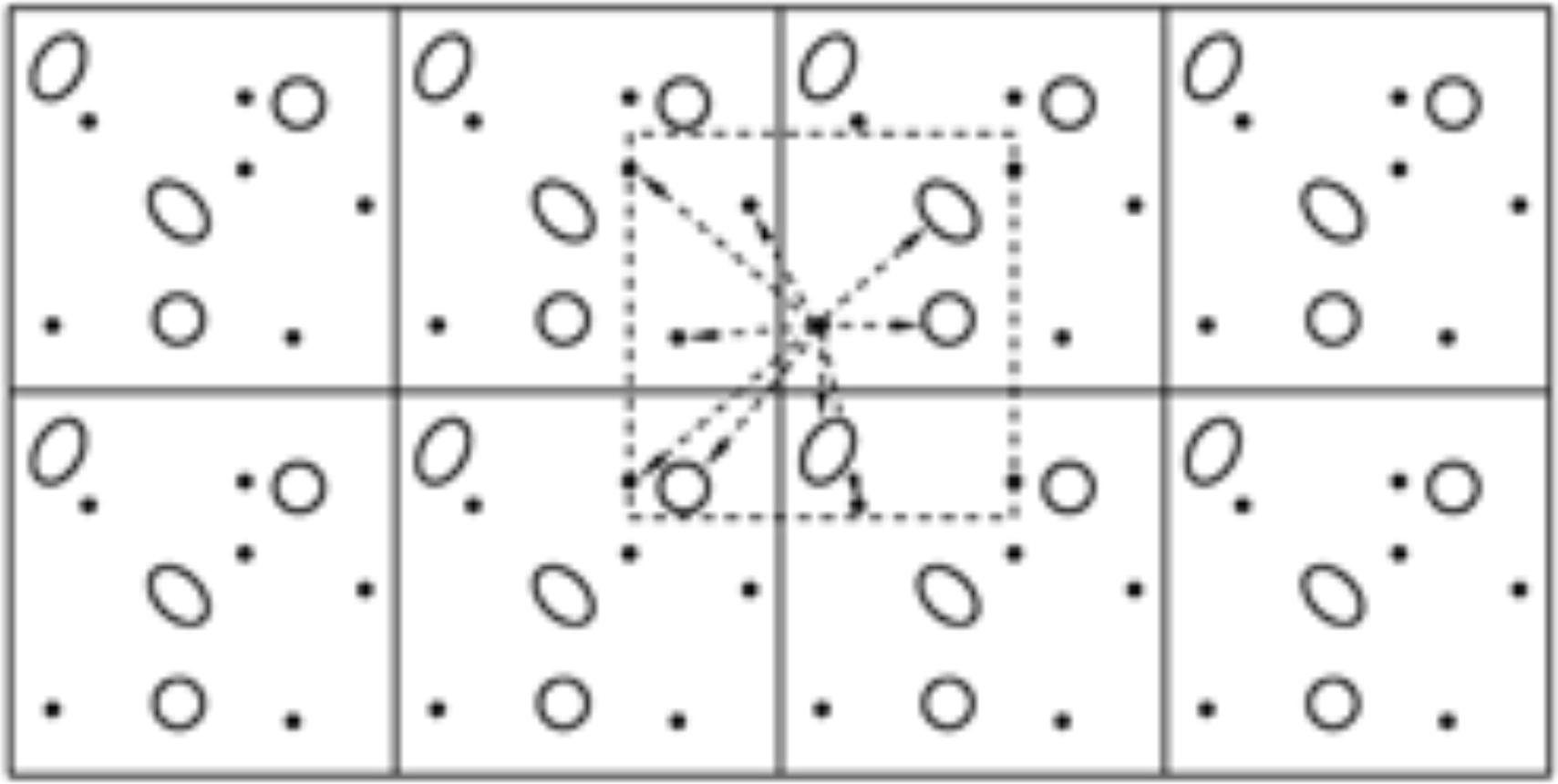
- Boundaries
- CPU saving methods
- Reduced units
- Long ranged forces

Boundary effects



- In small systems, boundary effects are always large.
- 1000 atoms in a simple cubic crystal – 488 boundary atoms.
- 1000000 atoms in a simple cubic crystal – still 6% boundary atoms.

Periodic boundary conditions



Algorithm 5 (Calculation of the energies)

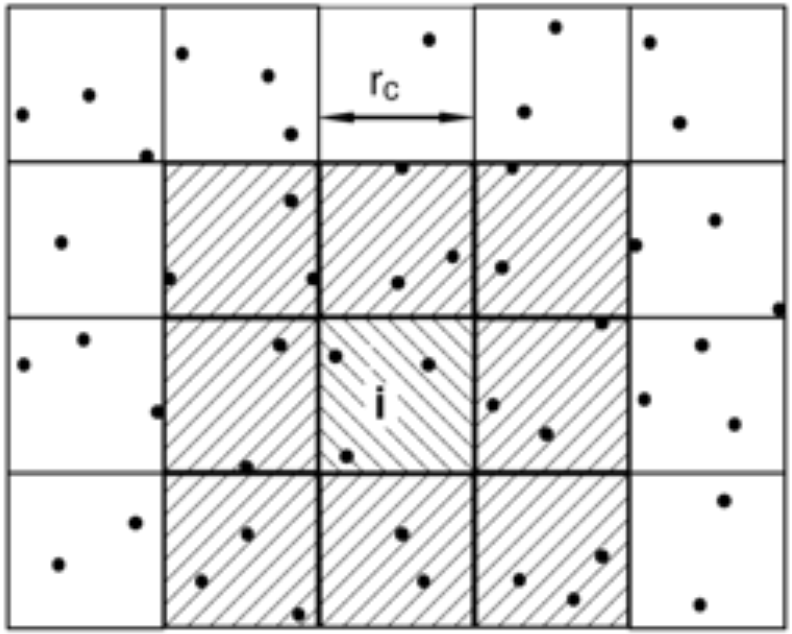
```
subroutine ener(x,en)                    determine the force
en=0                                     and energy
do i=1,npart
   f(i)=0                                set forces to zero
enddo
do i=1,npart-1
  do j=i+1,npart                         loop over all pairs
    xx=x(i)-x(j)
    xx=xx-box*nint(xx/box)               periodic boundary conditions
    r2=xx**2
    if (r2.lt.rc2) then                  test cutoff
      r2i=1/r2
      r6i=r2i**3
      ff=48*r2i*r6i*(r6i-0.5)            Lennard-Jones potential
      f(i)=f(i)+ff*xx                    update force
      f(j)=f(j)-ff*xx
      en=en+4*r6i*(r6i-1)-ecut           update energy
    endif
  enddo
enddo
return
end
```

Energy evaluation costs!

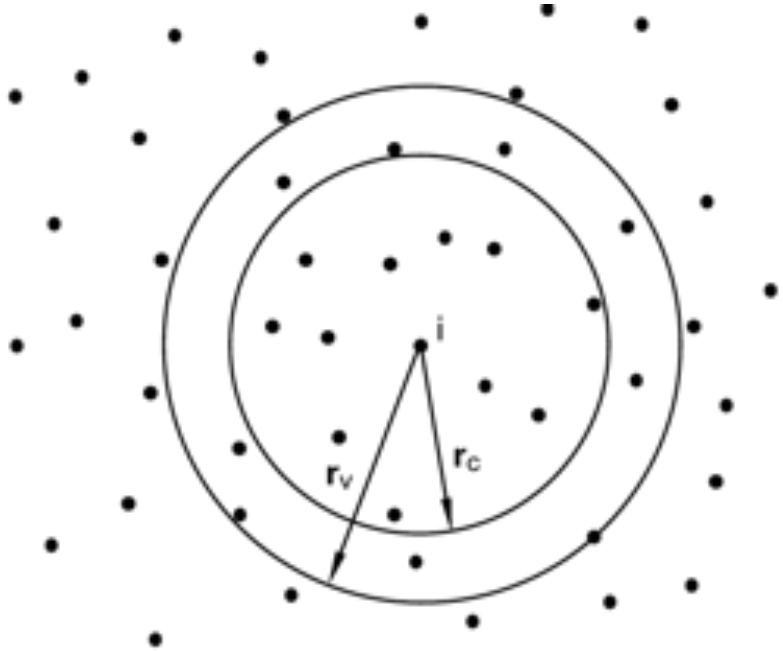
- The most time-consuming part of any simulation is the evaulation of all the interactions between the molecules.
- In general: N(N-1)/2 = O(N2)
- But often, intermolecular forces have a short range:
- Therefore, we do not have to consider interactions with far-away atoms.

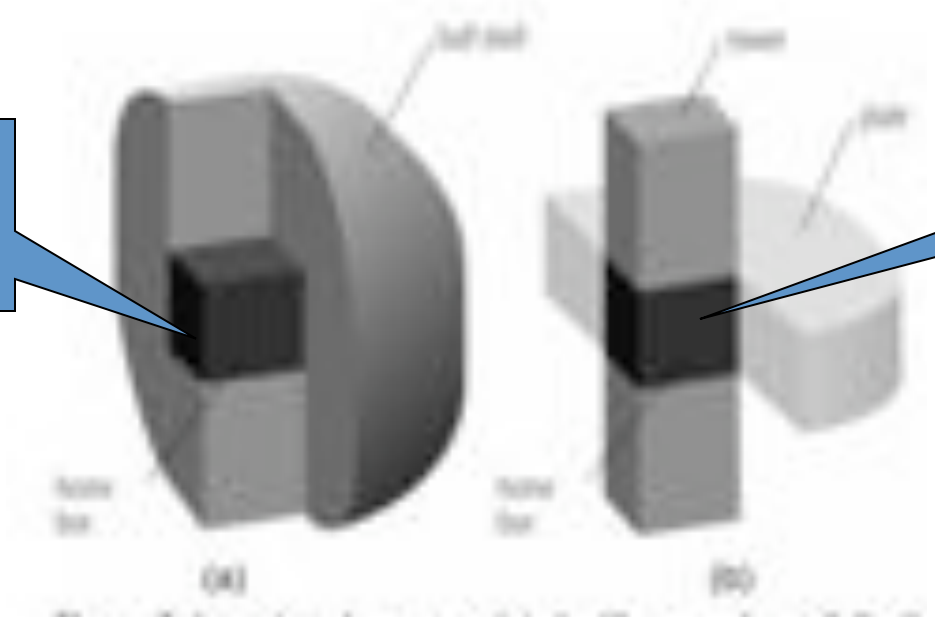
Saving CPU

- Cell list



Verlet List





New decomposition : "Neutral Territory" method

(Bowers, Dror & Shaw: J. Chem. Phys, 124: 184109 (2006)).

The **black** box maps onto one processor.

Each processor handles interaction between all particles i that are within the *xy* coordinates of the box and with its *z*-range, with all particles j that are within the *z* coordinates of the box and within its *xy* range.

Application: Lennard Jones potential

- The Lennard-Jones potential

$$u^{LJ}(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

- The truncated Lennard-Jones potential

$$u(r) = \begin{cases} u^{LJ}(r) & r \le r_c \\ 0 & r > r_c \end{cases}$$

- The truncated and shifted Lennard-Jones potential

$$u(r) = \begin{cases} u^{LJ}(r) - u^{LJ}(r_c) & r \le r_c \\ 0 & r > r_c \end{cases}$$

Algorithm 5 (Calculation of the energies)

```
subroutine ener(x,en)                    determine the force
en=0                                     and energy
do i=1,npart
   f(i)=0                                set forces to zero
enddo
do i=1,npart-1
  do j=i+1,npart                         loop over all pairs
     xx=x(i)-x(j)
     xx=xx-box*nint(xx/box)              periodic boundary conditions
     r2=xx**2
     if (r2.lt.rc2) then                 test cutoff
        r2i=1/r2
        r6i=r2i**3
        ff=48*r2i*r6i*(r6i-0.5)          Lennard-Jones potential
        f(i)=f(i)+ff*xx                  update force
        f(j)=f(j)-ff*xx
        en=en+4*r6i*(r6i-1)-ecut         update energy
     endif
  enddo
enddo
return
end
```

Reduced units

Example: Particles with mass **m** and pair potential:

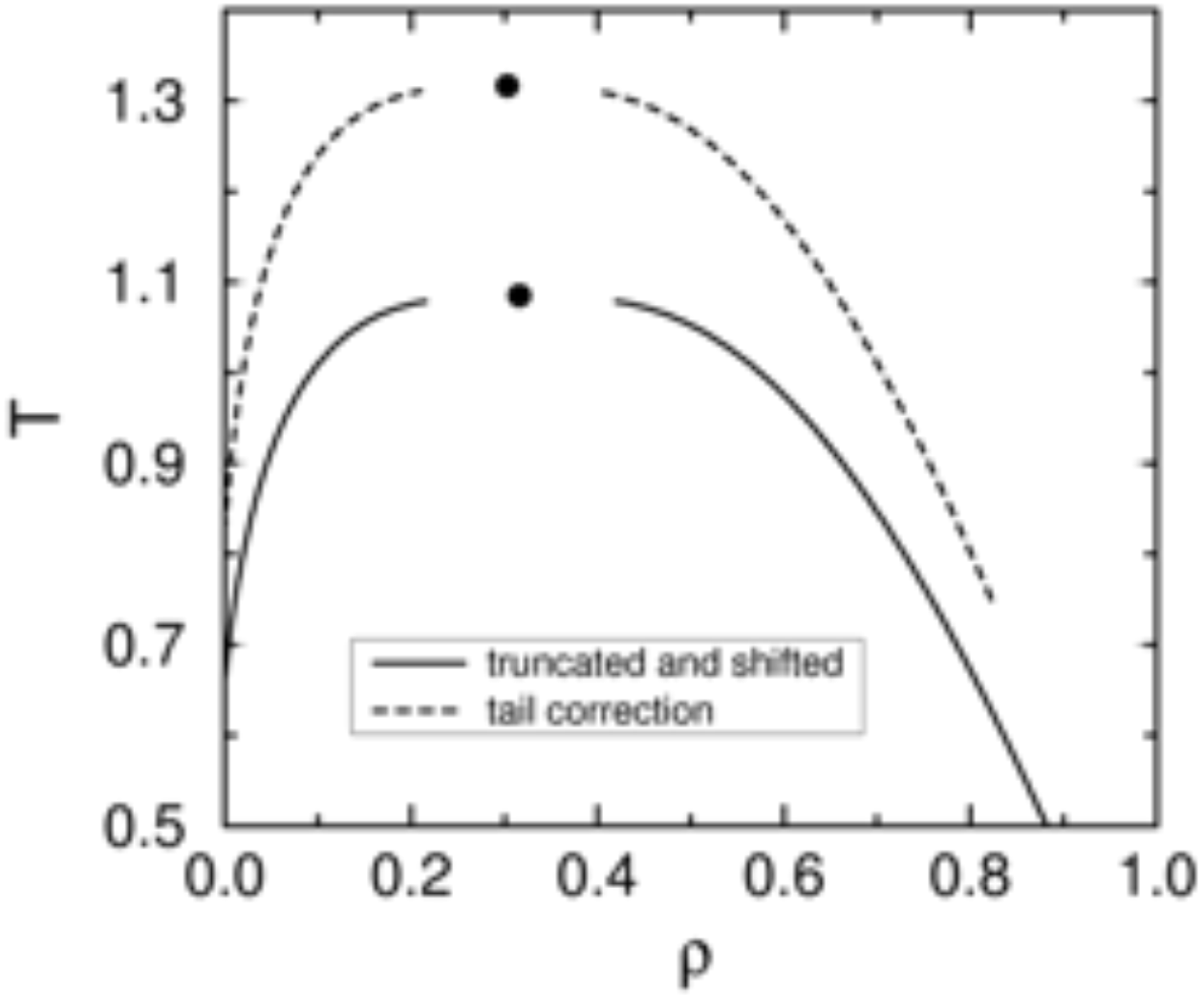
$$v(r) = \epsilon f(r/\sigma)$$

Unit of length: σ

Unit of energy: ε

Unit of time: $\sigma\sqrt{m/\epsilon}$

Phase diagrams of Lennard Jones fluids



Long ranged interactions

- Long-ranged forces require special techniques.
  - Coulomb interaction (1/r in 3D)
  - Dipolar interaction (1/r3 in 3D)

- ...and, in a different context:
  - Interactions through elastic stresses (1/r in 3D)
  - Hydrodynamic interactions (1/r in 3D)
  -

Beyond standard MC

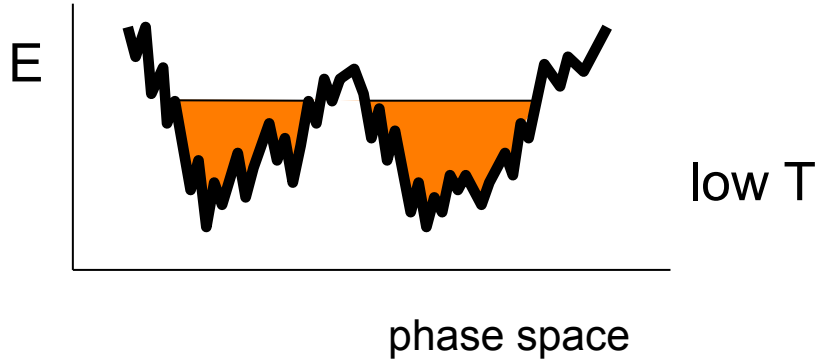
- **Parallel tempering**
- Parallel MC
- Cluster moves

Parallel tempering/Replica Exchange

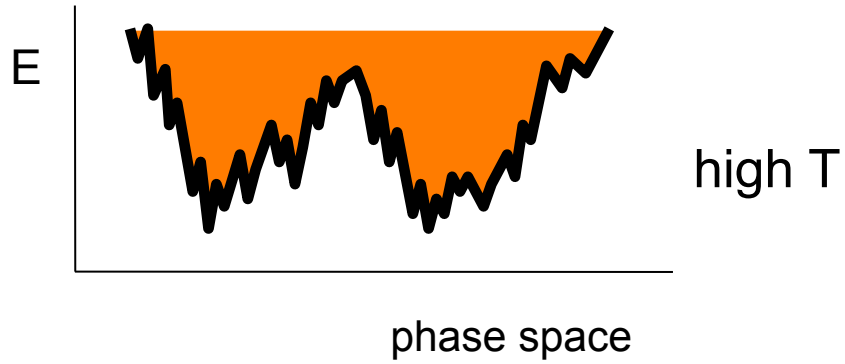
Ergodicity problems can occur, especially in glassy systems: biomolecules, molecular glasses, gels, etc.

The solution: go to high temperature

High barriers in energy landscape: difficult to sample



Barriers effectively low: easy to sample



Non-Boltzmann sampling

$$\langle A \rangle_{NVT_1} = \frac{1}{Q_{NVT_1}} \frac{1}{\Lambda^{3N} N!} \int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N) \exp\left[-\beta_1 U(\mathbf{r}^N)\right]$$

$$= \frac{\int \mathrm{d}\mathbf{r}^N A(\mathbf{r}^N) \exp\left[-\beta_1 U(\mathbf{r}^N)\right] \exp\left[\beta_2 U(\mathbf{r}^N) - \beta_2 U(\mathbf{r}^N)\right]}{\int \mathrm{d}\mathbf{r}^N \exp\left[-\beta_1 U(\mathbf{r}^N)\right] \exp\left[\beta_2 U(\mathbf{r}^N) - \beta_2 U(\mathbf{r}^N)\right]}$$

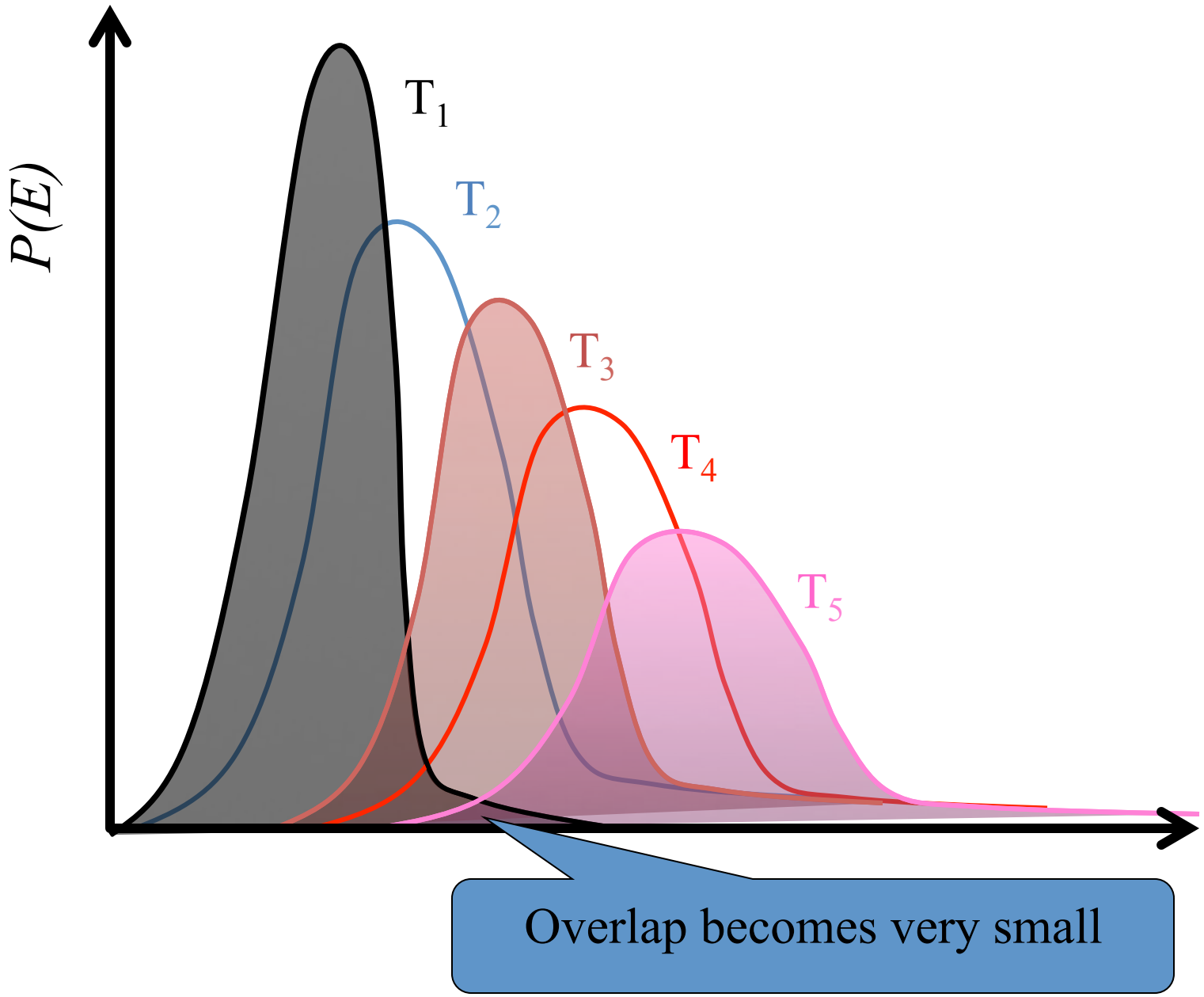
$$= \frac{\left\langle A \exp\left[(\beta_2 - \beta_1) U\right]\right\rangle_{NVT_2}}{\left\langle \exp\left[(\beta_2 - \beta_1) U\right]\right\rangle_{NVT_2}}$$

T_1 is arbitrary!

Why are we not using this?

We only need a *single* simulation!

We perform a simulation at $T=T_2$ and we determine A at $T=T_1$


P(E)
T1
T2
T3
T4
T5
Overlap becomes very small

Parallel tempering/Replica Exchange

Simulate two systems simultaneously

system 1
temperature T_1

system 2
temperature T_2

$$e^{-\beta_1 U_1(r^N)}$$

$$e^{-\beta_2 U_2(r^N)}$$

total Boltzmann weight:

$$e^{-\beta_1 U_1(r^N)} e^{-\beta_2 U_2(r^N)}$$

Swap move

- Allow two systems to swap

system 2
temperature T_1

system 1
temperature T_2

$$e^{-\beta_1 U_2(r^N)}$$

$$e^{-\beta_2 U_1(r^N)}$$

total Boltzmann weight:

$$e^{-\beta_1 U_2(r^N)} e^{-\beta_2 U_1(r^N)}$$

$$\mathrm{acc}(1 \leftrightarrow 2) = \min\left(1, e^{(\beta_2 - \beta_1)[U_2(r^N) - U_1(r^N)]}\right)$$

Acceptance rule

The ratio of the new boltzmann factor over the old one is

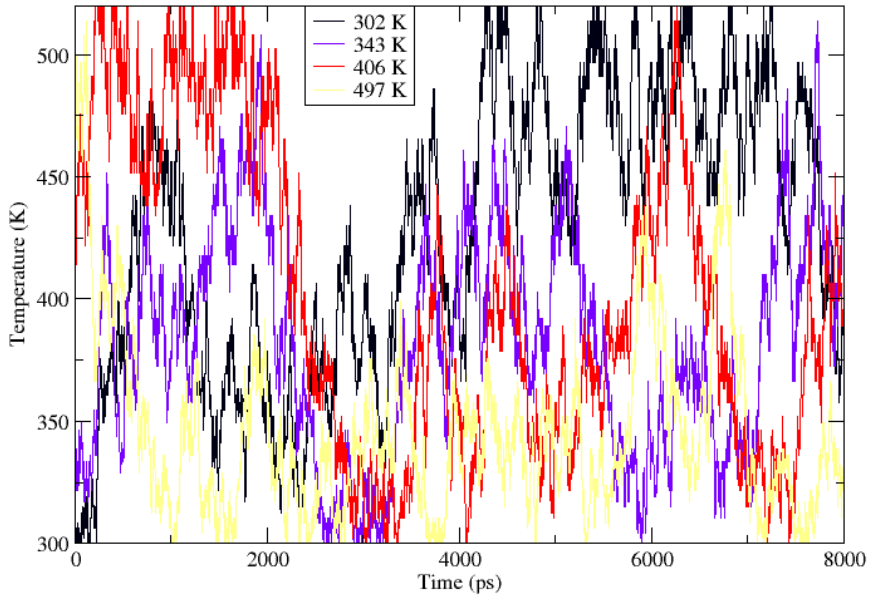
$$\frac{\mathcal{N}(n)}{\mathcal{N}(o)} = e^{(\beta_2 - \beta_1)[U_2(r^N) - U_2(r^N)]}$$

the swap acceptance ratio is

$$\mathrm{acc}(1 \leftrightarrow 2) = \min\left(1, e^{(\beta_2 - \beta_1)[U_2(r^N) - U_1(r^N)]}\right)$$

More replicas

Consider M replica's in the NVT ensemble at a different temperature.



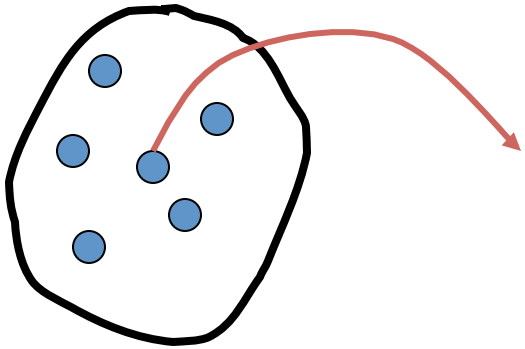
A swap between two systems of different temperatures (T_i,T_j) is accepted if their potential energies are near.

other parameters can be used: Hamiltonian exchange

Beyond standard MC

- Parallel tempering
- **Parallel MC**
- Cluster moves

How to do *parallel* Monte Carlo



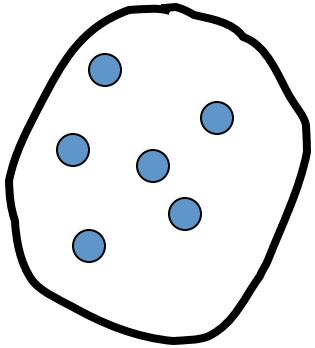
- Is it possible to do Monte Carlo in parallel
  - Monte Carlo is sequential!
  - We first have to know the fate of the current move before we can continue!

Parallel Monte Carlo

Algorithm (*WRONG*):

1. Generate *k* trial configurations in parallel
2. Select out of these the one with the lowest energy

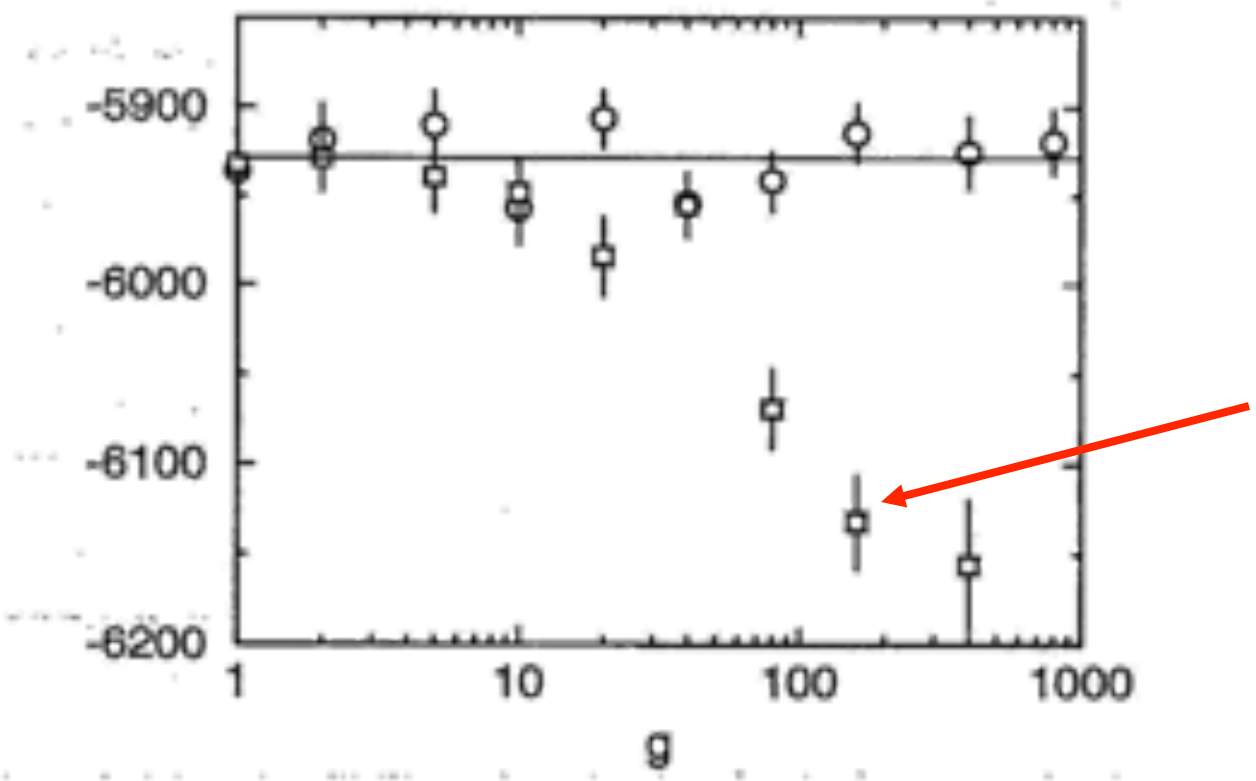
$$P(n) = \frac{\exp\left[-\beta\left(U_n\right)\right]}{\sum_{j=1}^{g} \exp\left[-\beta\left(U_j\right)\right]}$$



3. Accept and reject using normal Monte Carlo rule:

$$\mathrm{acc}\left(o \rightarrow n\right) = \exp\left[-\beta\left(U_n - U_o\right)\right]$$

Conventional acceptance rule



Conventional acceptance rules leads to a ***bias***

Detailed balance!

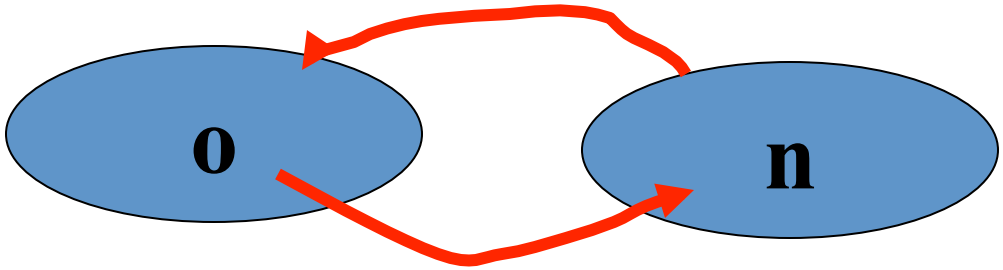
$$K(o \to n) = K(n \to o)$$

$$K(o \to n) = N(o) \times \alpha(o \to n) \times \mathrm{acc}(o \to n)$$

$$K(n \to o) = N(n) \times \alpha(n \to o) \times \mathrm{acc}(n \to o)$$

$$\frac{\mathrm{acc}(o \to n)}{\mathrm{acc}(n \to o)} = \frac{N(n) \times \alpha(n \to o)}{N(o) \times \alpha(o \to n)} = \frac{N(n)}{N(o)}$$

Detailed balance



$$K(o \to n) = K(n \to o)$$

$$K(o \to n) = N(o) \times \alpha(o \to n) \times \mathrm{acc}(o \to n)$$

$$K(n \to o) = N(n) \times \alpha(n \to o) \times \mathrm{acc}(n \to o)$$

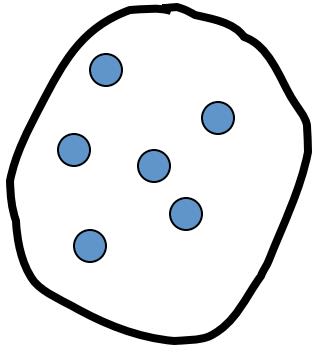
$$\frac{\mathrm{acc}(o \to n)}{\mathrm{acc}(n \to o)} = \frac{N(n) \times \alpha(n \to o)}{N(o) \times \alpha(o \to n)} = \frac{N(n)}{N(o)}$$

?

$$K(o \rightarrow n) = N(o) \times \alpha(o \rightarrow n) \times \mathrm{acc}(o \rightarrow n)$$

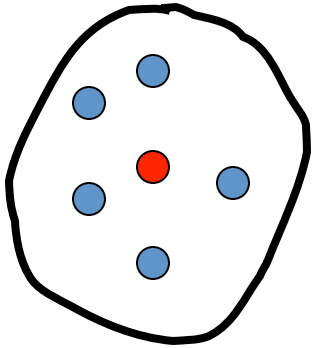
$$\alpha(o \rightarrow n) = \frac{\exp\left[-\beta\left(U_n\right)\right]}{\sum_{j=1}^{g} \exp\left[-\beta\left(U_j\right)\right]}$$

$$\alpha(o \rightarrow n) = \frac{\exp\left[-\beta\left(U_n\right)\right]}{W\left(n\right)}$$



$$\alpha(n \rightarrow o) = \frac{\exp\left[-\beta\left(U_o\right)\right]}{\sum_{j=1}^{g} \exp\left[-\beta\left(U_j\right)\right]}$$

$$\alpha(n \rightarrow o) = \frac{\exp\left[-\beta\left(U_o\right)\right]}{W\left(o\right)}$$

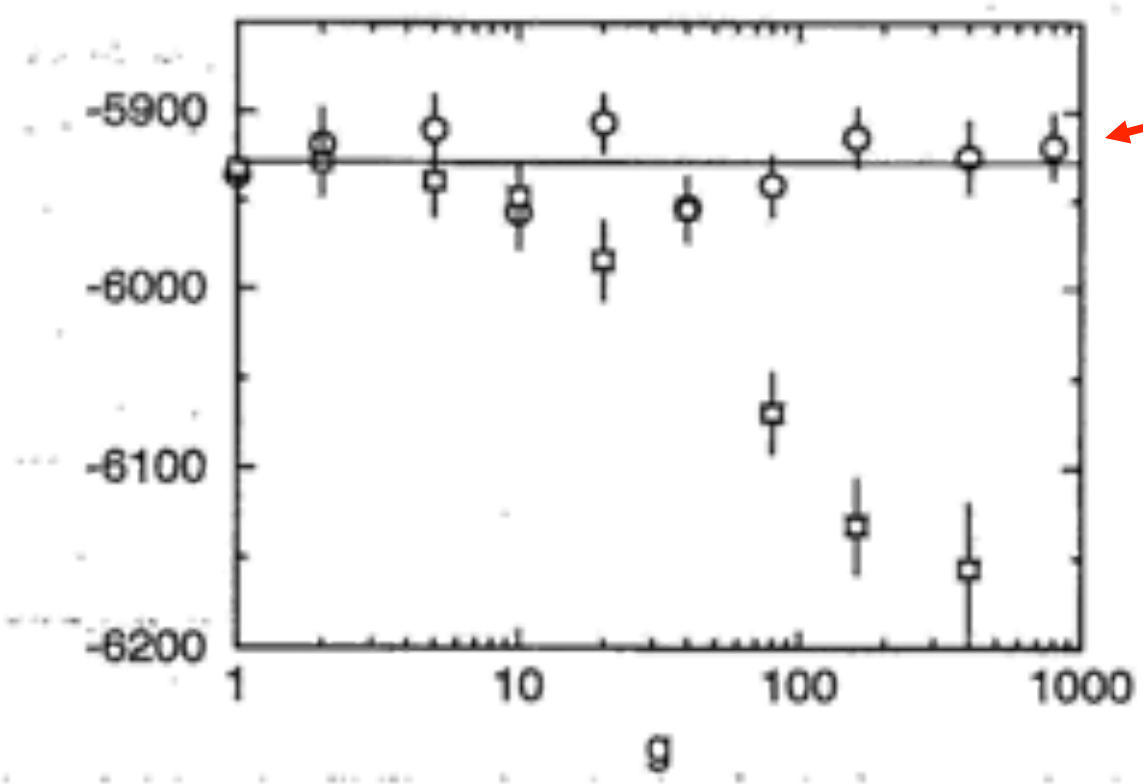


$$\frac{\mathrm{acc}(o \to n)}{\mathrm{acc}(n \to o)} = \frac{N(n) \times \alpha(n \to o)}{N(o) \times \alpha(o \to n)} = \frac{N(n)}{N(o)}$$

$$\alpha(o \to n) = \frac{\exp\left[-\beta\left(U_n\right)\right]}{W\left(n\right)} \quad \alpha(n \to o) = \frac{\exp\left[-\beta\left(U_o\right)\right]}{W\left(o\right)}$$

$$\frac{\mathrm{acc}(o \to n)}{\mathrm{acc}(n \to o)} = \frac{N(n) \times \dfrac{\exp\left[-\beta\left(U_o\right)\right]}{W\left(o\right)}}{N(o) \times \dfrac{\exp\left[-\beta\left(U_n\right)\right]}{W\left(n\right)}} = \frac{W(n)}{W(o)}$$

Modified acceptance rule



Modified acceptance rule remove the ***bias*** exactly!

Beyond standard MC

- Parallel tempering
- Parallel MC
- **Cluster moves**

Metropolis Monte Carlo:

1. generate trial moves
2. Move if accepted
3. Otherwise, stay where you are

Metropolis, Rosenbluth,Rosenbluth, Teller and Teller:

$$\mathrm{acc}(o \to n) = \min\left(1, \exp\{-\beta[\mathcal{U}(\mathbf{r}'^N) - \mathcal{U}(\mathbf{r}^N)]\}\right)$$

Alternative: "symmetric rule"

$$\mathrm{acc}(o \to n) = \frac{\exp\{-\beta\mathcal{U}(\mathbf{r}'^N)\}}{\exp\{-\beta\mathcal{U}(\mathbf{r}'^N)\} + \exp\{-\beta\mathcal{U}(\mathbf{r}^N)\}}$$

Unsatisfactory?

© D. Frenkel

Solution of conflict: if we do **not** impose

$$\alpha(o \to n) = \alpha(n \to o)$$ then

$$\frac{\mathsf{acc}(o \to n)}{\mathsf{acc}(n \to o)} = \frac{\alpha(n \to o)}{\alpha(o \to n)} \exp\{-\beta[\mathcal{U}(n) - \mathcal{U}(o)]\}.$$

In particular, if:

$$\frac{\alpha(n \to o)}{\alpha(o \to n)} = \exp\{-\beta[\mathcal{U}(o) - \mathcal{U}(n)]\}.$$

Then

$$\frac{\mathsf{acc}(o \to n)}{\mathsf{acc}(n \to o)} = 1$$ (100% acceptance)

100% acceptance can be achieved in special cases:
e.g. Swendsen-Wang algorithm

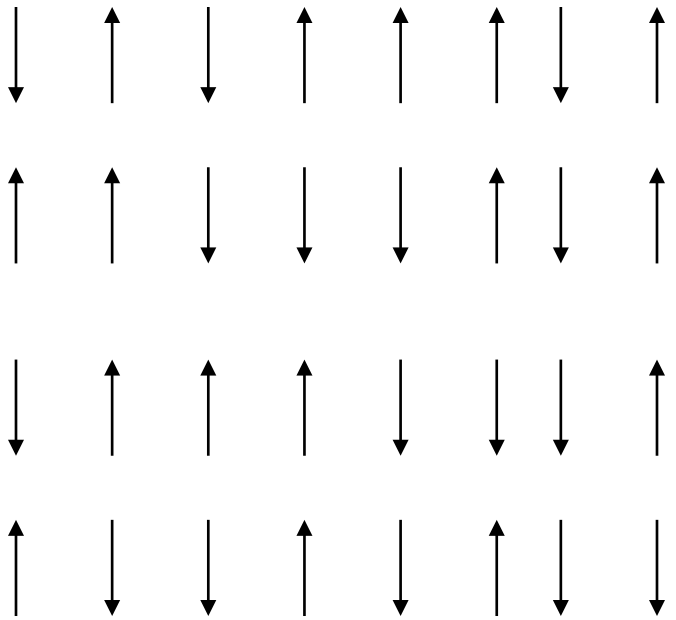
Discrete spin models (Potts, Ising).

Illustration: 2D Ising model:

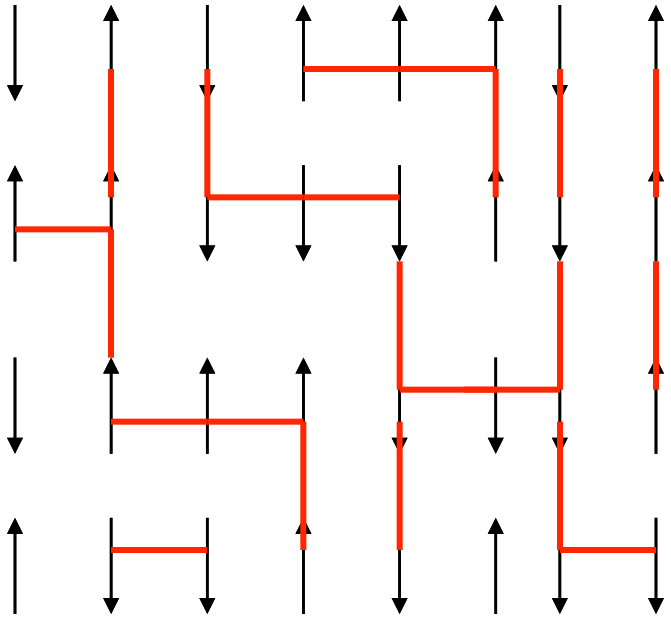
Parallel nearest neighbor spins: energy **–J**

Anti-parallel nearest neighbor spins: energy **+J**

$$U = -J \sum_{i,j} s_i s_j$$



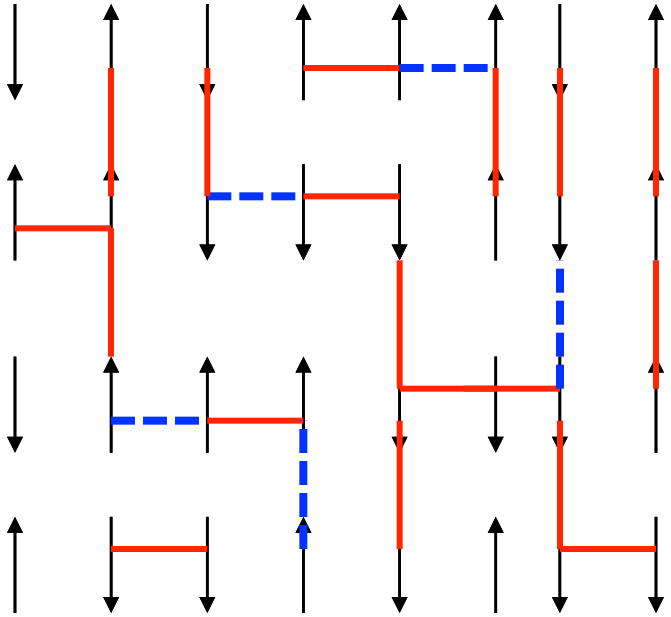
Snapshot: some neighbors are parallel, others anti-parallel



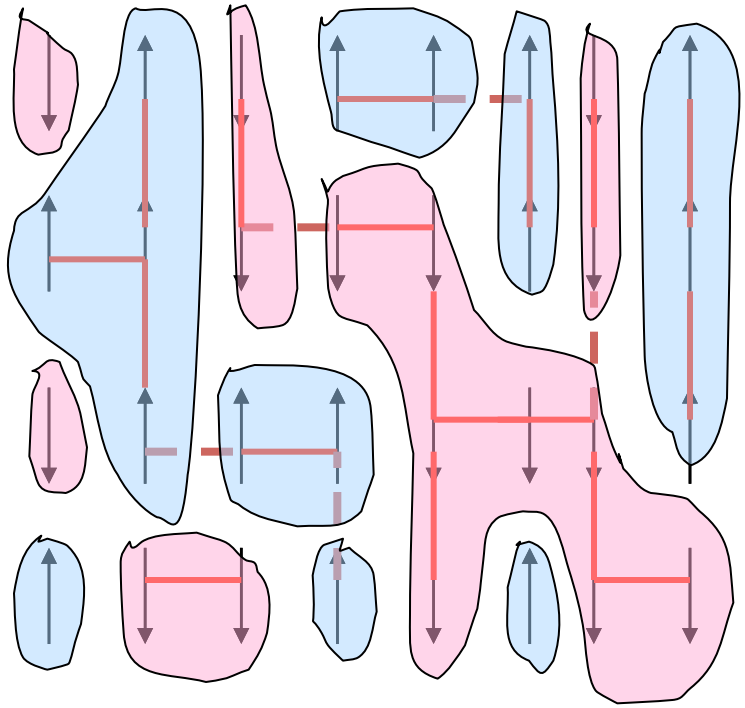
Count number of bonds between parallel neighbors: $\mathbf{N_p}$

Number of bonds between anti-parallel neigbors is: $\mathbf{N_a}$

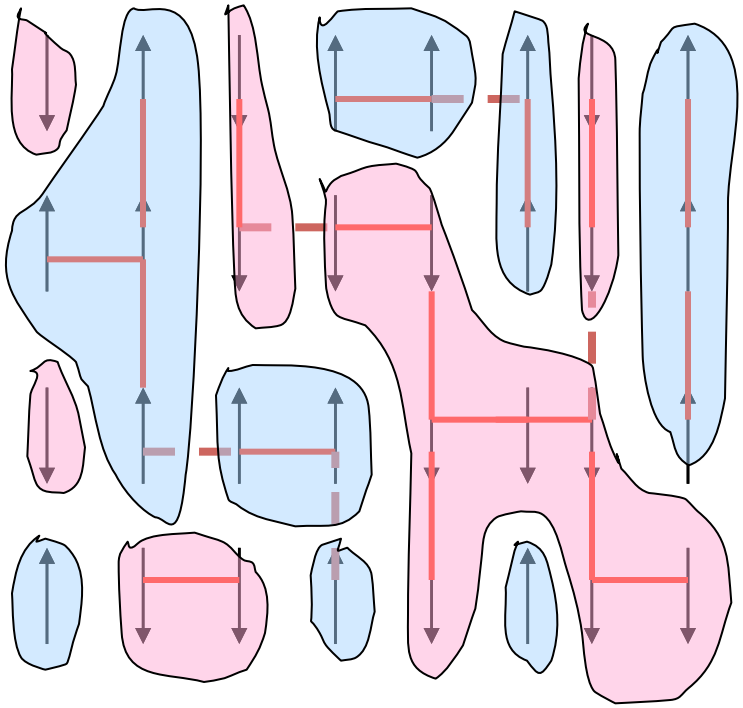
Total energy: $\mathbf{U = (N_a - N_p)\ J}$



Now, make “bonds”. Bonds only form between parallel neighbors. The probability to have a bond (red line) between parallel neighbors is **p** (as yet undetermined). With a probability **1-p**, parallel neighbors are not connected (blue dashed line).

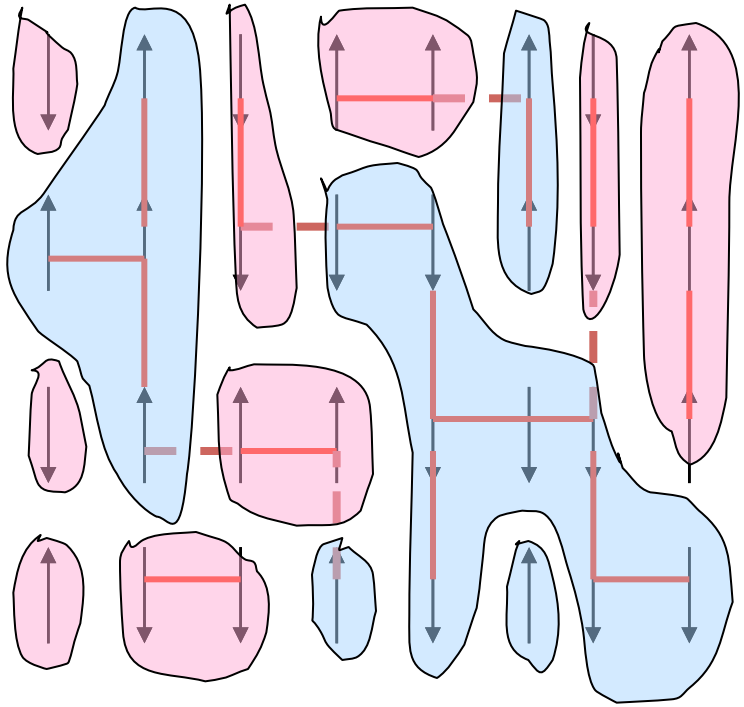


Form clusters of all spins that are connected by bonds. Some clusters are all “spin up” others are all “spin down”. Let us denote the number of clusters by **M**.



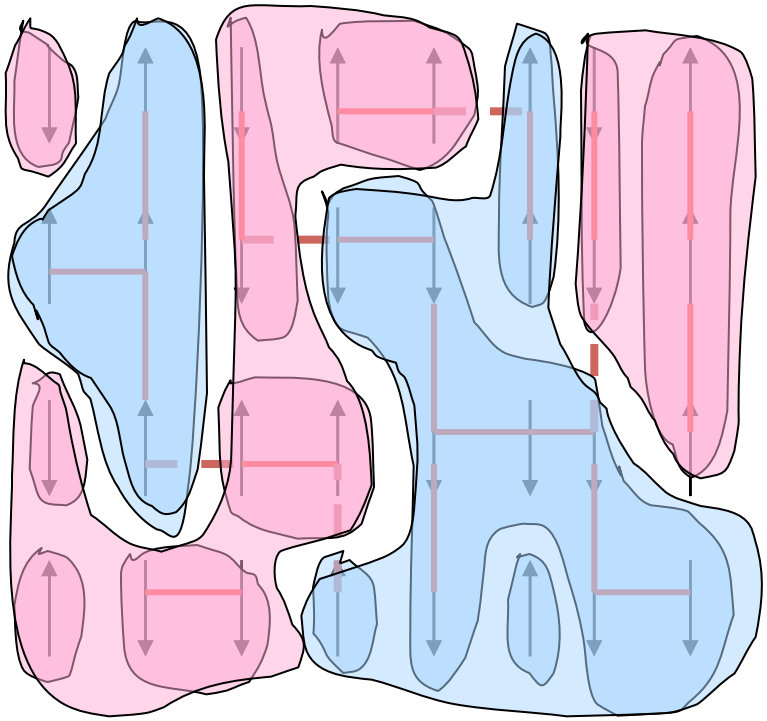
The probability to generate a particular cluster structure where there are $\mathbf{n_c}$ bonds between $\mathbf{N_p}$ pairs of parallel neighbors is:

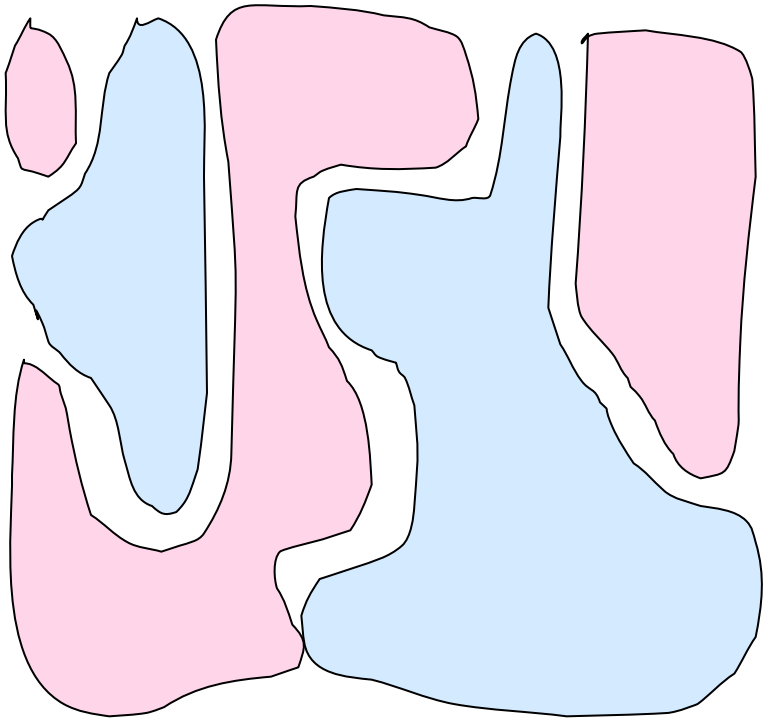
$$P_{gen} = p^{n_c}(1-p)^{N_p - n_c}$$



Now randomly flip clusters. This yields a new cluster configuration with probability $\mathbf{P_{(flip)} = (1/2)^M}$.

Then reconnect parallel spins





New cluster structure!

Now make it into a Monte Carlo algorithm:

$$P_o P_{clus}(o) P_{flip}(M) P_{acc}(o \to n)$$

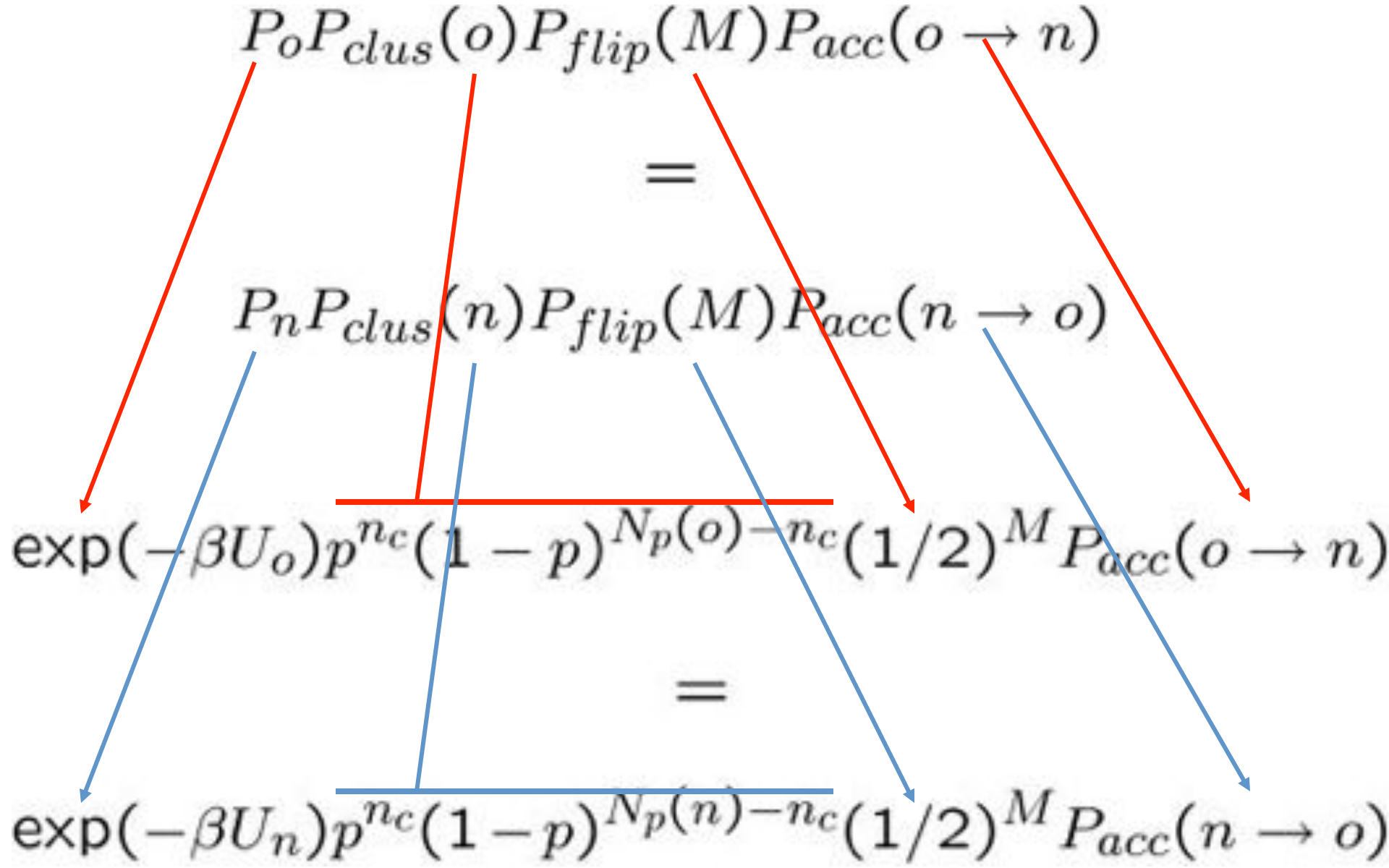
$$=$$

$$P_n P_{clus}(n) P_{flip}(M) P_{acc}(n \to o)$$

$$\exp(-\beta U_o) p^{n_c} (1-p)^{N_p(o)-n_c} (1/2)^M P_{acc}(o \to n)$$

$$=$$

$$\exp(-\beta U_n) p^{n_c} (1-p)^{N_p(n)-n_c} (1/2)^M P_{acc}(n \to o)$$


$P_o P_{clus}(o) P_{flip}(M) P_{acc}(o \to n)$
=
$P_n P_{clus}(n) P_{flip}(M) P_{acc}(n \to o)$
$\exp(-\beta U_o) p^{n_c} (1-p)^{N_p(o)-n_c} (1/2)^M P_{acc}(o \to n)$
=
$\exp(-\beta U_n) p^{n_c} (1-p)^{N_p(n)-n_c} (1/2)^M P_{acc}(n \to o)$

Moreover, we want 100% acceptance, i.e.:

$$P_{acc}(o \to n) = P_{acc}(n \to o) = 1$$

$$\exp(-\beta U_o) p^{n_c} (1-p)^{N_p(o)-n_c} (1/2)^M P_{acc}(o \to n)$$

$$=$$

$$\exp(-\beta U_n) p^{n_c} (1-p)^{N_p(n)-n_c} (1/2)^M P_{acc}(n \to o)$$

Hence:

$$\exp(-\beta U_o)(1-p)^{N_p(o)} = \exp(-\beta U_n)(1-p)^{N_p(n)}$$

$$\exp(\beta(U_n - U_o)) = (1-p)^{N_p(n)-N_p(o)}$$

But remember:

$$U_n - U_o = J(N_a(n) - N_p(n)) - J(N_a(o) - N_p(o))$$

or

$$\Delta U = J(\Delta N_a - \Delta N_p)$$

But: $\Delta N_a = -\Delta N_p$

and therefore

$$\Delta U = -2J\Delta N_p$$

$$\exp(\beta(U_n - U_o)) = \exp(-2\beta J(N_p(n) - N_p(o)))$$

Combining this with:

$$\exp(\beta(U_n - U_o)) = (1-p)^{N_p(n)-N_p(o)}$$

we obtain:

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

100% acceptance!!!