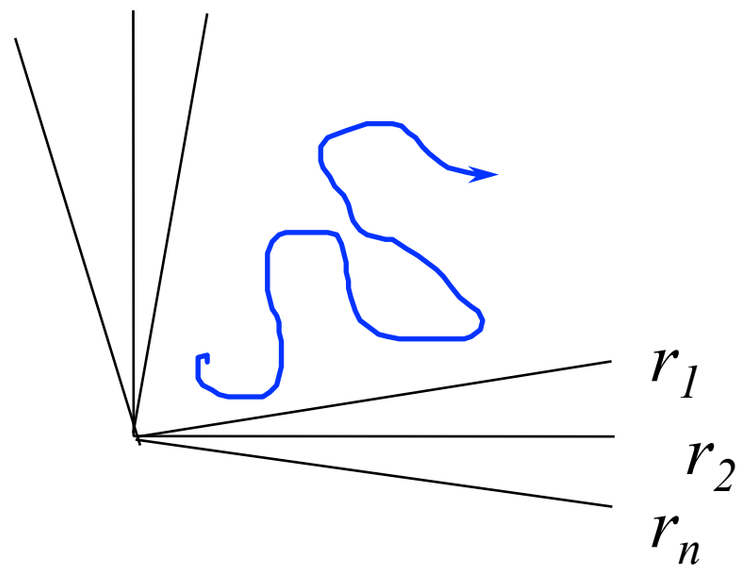
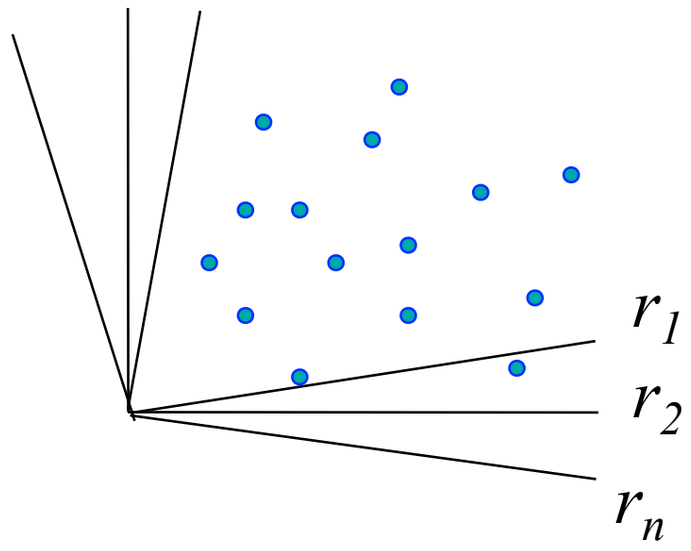
3. Monte Carlo Simulations

Molecular Simulations

➡ **Molecular dynamics**:
solve equations of motion



➡ **Monte Carlo**: importance sampling

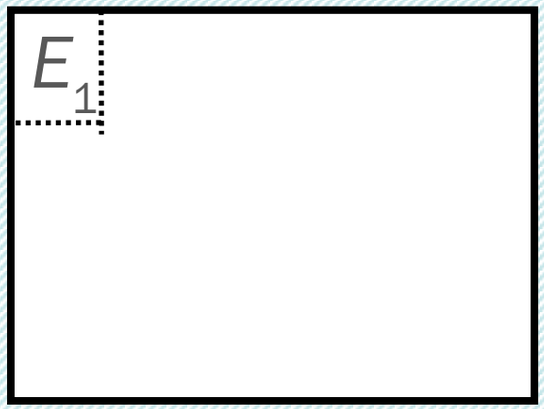


Monte Carlo Simulations

3. Monte Carlo Simulations

3.2 Statistical Thermodynamics

Canonical ensemble: statistical mechanics



Consider a small system that can exchange energy with a big reservoir

$$\ln\Omega\left(E_1, E-E_1\right) = \ln\Omega\left(E\right) - \left(\frac{\partial \ln\Omega}{\partial E}\right) E_1 + \cdots$$

$=1/k_BT$

If the reservoir is very big we can ignore the higher order terms:

$$\frac{\ln\Omega\left(E_1, E-E_1\right)}{\ln\Omega\left(E\right)} = -\frac{E_1}{k_B T}$$

Hence, the probability to find E_1:

$$P\left(E_1\right) = \frac{\Omega\left(E_1, E-E_1\right)}{\sum_i \Omega\left(E_i, E-E_i\right)} = \frac{\Omega\left(E_1, E-E_1\right)/\Omega\left(E\right)}{\sum_i \Omega\left(E_i, E-E_i\right)/\Omega\left(E\right)} = C\frac{\Omega\left(E_1, E-E_1\right)}{\Omega\left(E\right)}$$

$$P\left(E_1\right) \propto \exp\left[-\frac{E_1}{k_B T}\right] \propto \exp\left[-\beta E_1\right]$$

$\beta=1/k_BT$

Summary: Canonical ensemble (N,V,T)

Partition function:

$$Q_{N,V,T} = \frac{1}{\Lambda^{3N} N!} \int e^{-\frac{U(r)}{k_B T}} dr^{3N}$$

Probability to find a particular configuration:

$$P\left(r^{3N}\right) \propto e^{-\frac{U\left(r^{3N}\right)}{k_B T}}$$

Ensemble average:

$$\left\langle A \right\rangle_{N,V,T} = \frac{\frac{1}{\Lambda^{3N} N!} \int A\left(r\right) e^{-\beta U\left(r\right)} dr^{3N}}{Q_{N,V,T}} = \frac{\int A\left(r\right) e^{-\beta U\left(r\right)} dr^{3N}}{\int e^{-\beta U\left(r\right)} dr^{3N}}$$

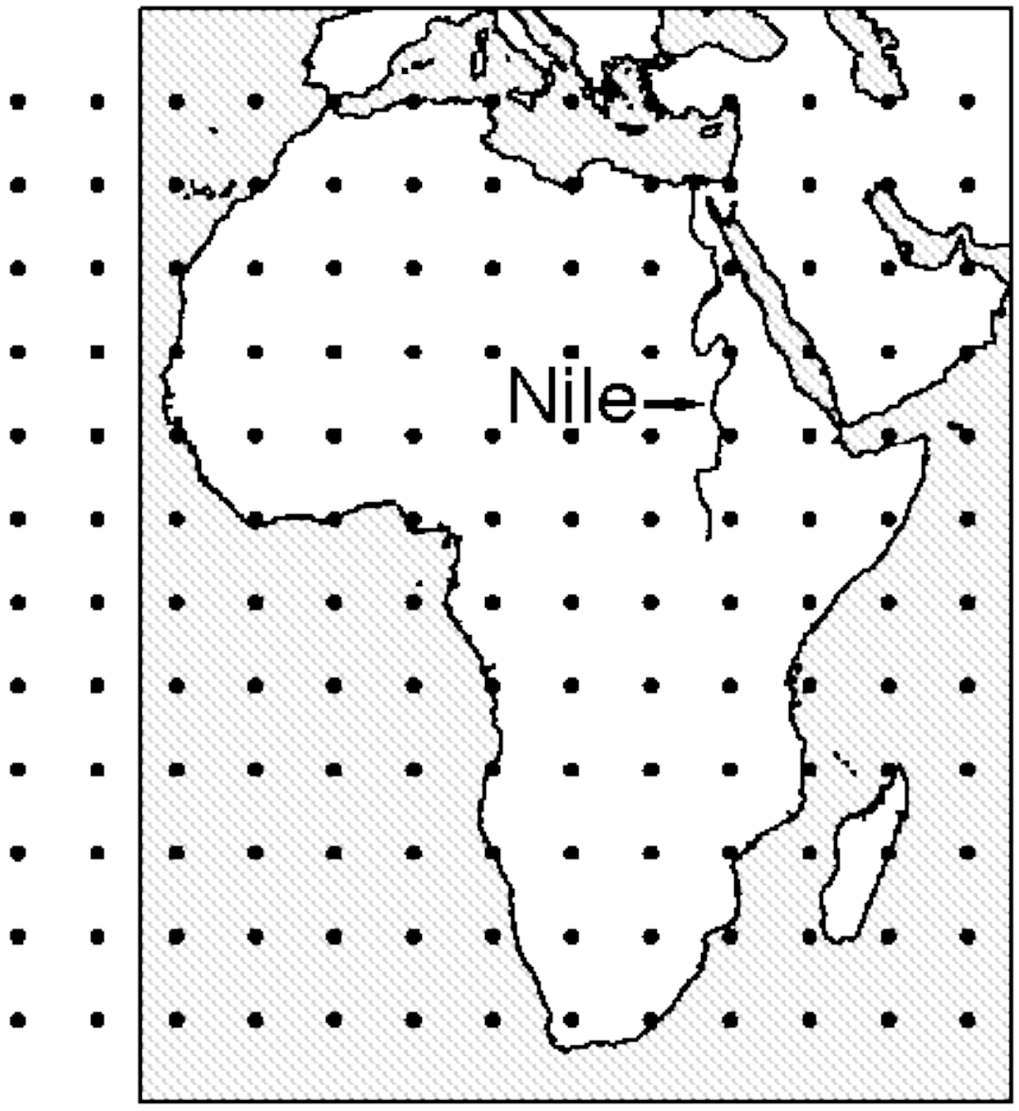
Free energy:

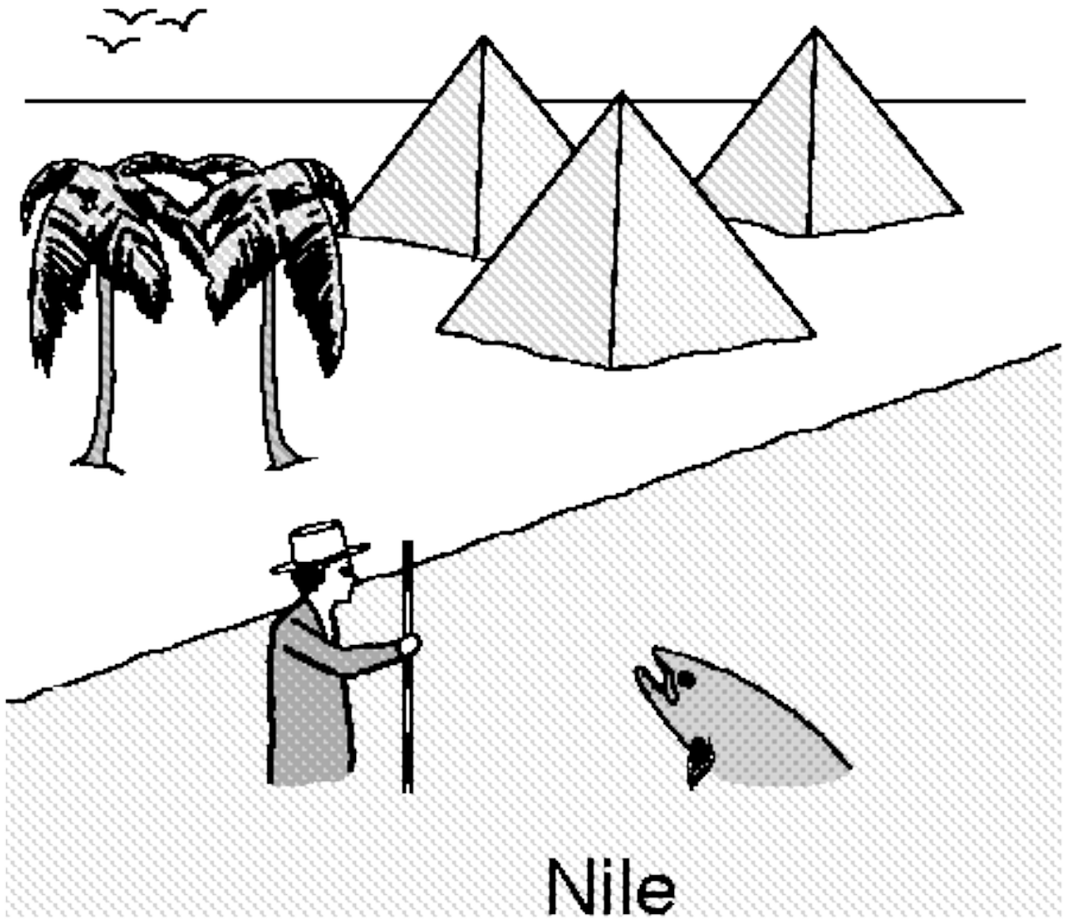
$$\beta F = -\ln Q_{NVT}$$

3.Monte Carlo Simulations

3.3 Importance Sampling

Numerical Integration





Monte Carlo simulations

Generate M configurations using Monte Carlo moves:

$$\left\{r_1^{3N}, r_2^{3N}, r_3^{3N}, r_4^{3N}, \cdots, r_M^{3N}\right\}$$

We can compute the average:

$$\overline{A} = \sum_{i=1}^{M} A\left(r_i^{3N}\right)$$

The probability to generate a configuration in our MC scheme: P^{MC}

$$\overline{A} = \frac{\int A\left(r^{3N}\right) P^{MC}\left(r^{3N}\right) dr^{3N}}{\int P^{MC}\left(r^{3N}\right) dr^{3N}}$$

Question: how to chose P^{MC} such that we sample the canonical ensemble?

Ensemble Average

$$\langle A \rangle_{NVT} = \frac{1}{Q_{NVT}} \frac{1}{N! \Lambda^{3N}} \int A\left(r^{3N}\right) e^{-\beta U\left(r^{3N}\right)} dr^{3N}$$

We can rewrite this using the probability to find a particular configuration

$$\langle A \rangle_{NVT} = \int A\left(r^{3N}\right) P\left(r^{3N}\right) dr^{3N}$$

with

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

$$\langle A \rangle_{NVT} = \int A\left(r^{3N}\right) P\left(r^{3N}\right) dr^{3N} = \frac{\int A\left(r^{3N}\right) e^{-\beta U\left(r^{3N}\right)} dr^{3N}}{\int e^{-\beta U\left(r^{3N}\right)} dr^{3N}}$$

Monte Carlo - canonical ensemble

Canonical ensemble:

$$\left\langle A \right\rangle_{NVT} = \int A\left(r^{3N}\right) P\left(r^{3N}\right) dr^{3N} = \frac{\int A\left(r^{3N}\right) e^{-\beta U\left(r^{3N}\right)} dr^{3N}}{\int e^{-\beta U\left(r^{3N}\right)} dr^{3N}}$$

with

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

2. No need to know the partition function

Monte Carlo:

$$\bar{A} = \sum_{i=1}^{M} A\left(r_i^{3N}\right)$$

$$\bar{A} = \frac{\int A\left(r^{3N}\right) P^{MC}\left(r^{3N}\right) dr^{3N}}{\int P^{MC}\left(r^{3N}\right) dr^{3N}}$$

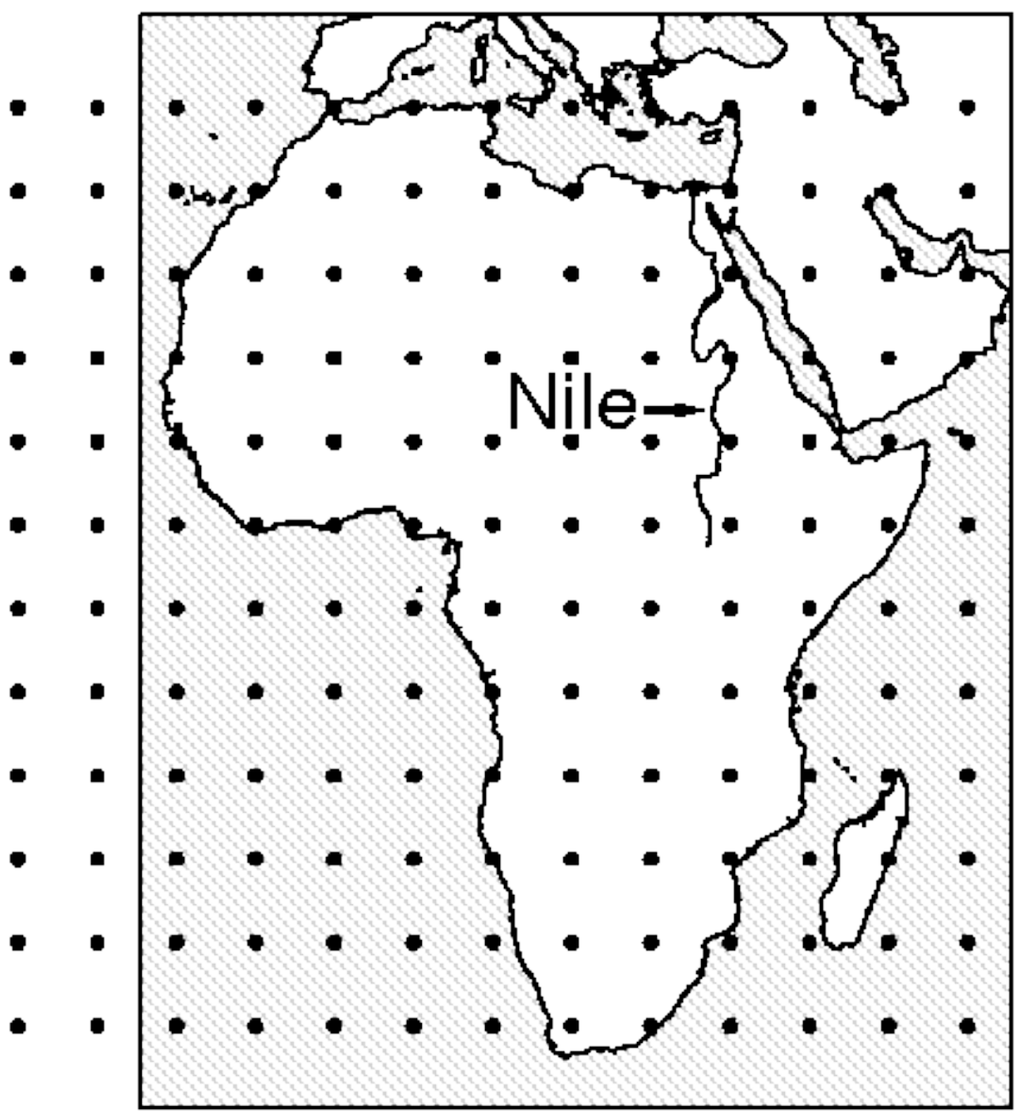
Hence, we need to sample:

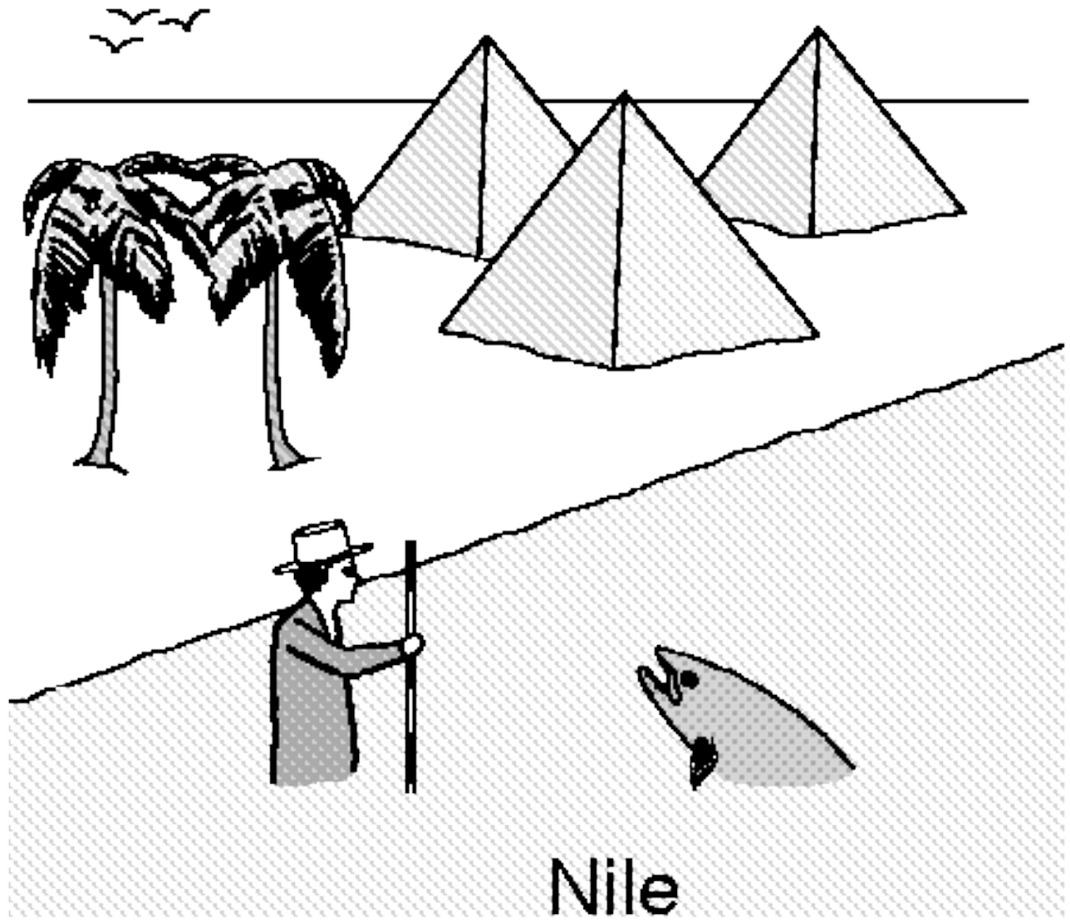
$$P^{MC}\left(r^{3N}\right) = C e^{-\beta U\left(r^{3N}\right)}$$

1. No need to know C

$$\bar{A} = \frac{C\int A\left(r^{3N}\right) e^{-\beta U\left(r^{3N}\right)} dr^{3N}}{C\int e^{-\beta U\left(r^{3N}\right)} dr^{3N}} = \frac{\int A\left(r^{3N}\right) e^{-\beta U\left(r^{3N}\right)} dr^{3N}}{\int e^{-\beta U\left(r^{3N}\right)} dr^{3N}} = \left\langle A \right\rangle_{NVT}$$

Importance Sampling: what got lost?





3.Monte Carlo Simulation

3.4 Details of the algorithm

Algorithm 1 (Basic Metropolis Algorithm)

```
 PROGRAM mc                                  basic Metropolis algorithm

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

Comments to this algorithm:

1. *Subroutine* `mcmove` *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 mcmove                    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 (2.2.1)
 +    *(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* $[0, 1]$.

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`?

3.Monte Carlo Simulations

3.4.1 Detailed balance

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`?

canonical ensembles

Markov Processes

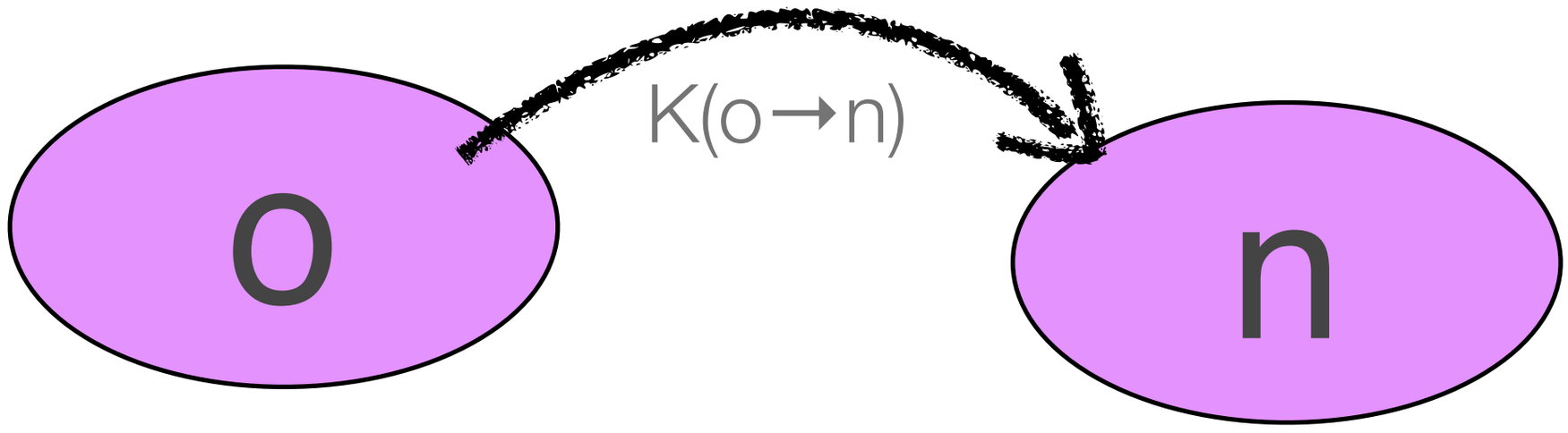
Markov Process

- Next step only depends on the current state
- Ergodic: all possible states can be reached by a set of single steps
- Detailed balance
- * Process will approach a limiting distribution

Ensembles versus probability

- *P(o)*: probability to find the state *o*
- Ensemble: take a very large number (*M*) of identical systems: *N(o) = M x P(o)*; the total number of systems in the state o

Markov Processes - Detailed Balance

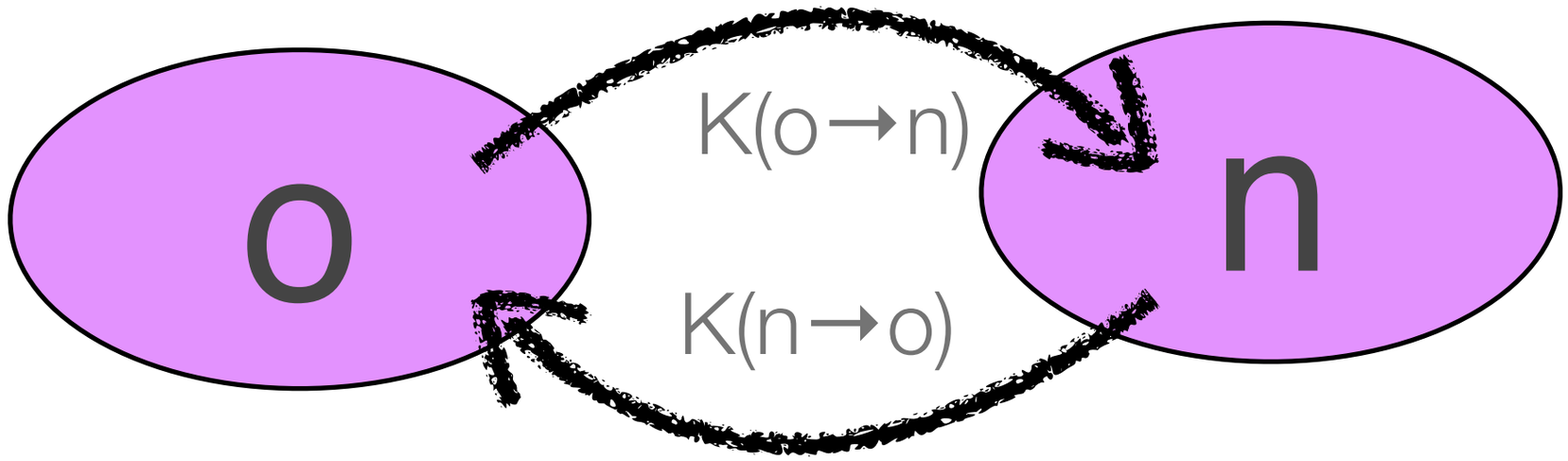


K(o → n): total number of systems in our ensemble that move *o* → *n*

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

- *N(o)* : total number of systems in our ensemble in state o
- *α(o → n)*: a priori probability to generate a move *o* → *n*
- *acc(o → n)*: probability to accept the move *o* → *n*

Markov Processes - Detailed Balance



Condition of detailed balance:

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

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

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

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

NVT-ensemble

In the canonical ensemble the number of configurations in state *n* is given by:

$$N(n) \propto e^{-\beta U(n)}$$

We assume that in our Monte Carlo moves the a priori probability to perform a move is independent of the configuration:

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

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

Which gives as condition for the acceptance rule:

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

Algorithm 2 (Attempt to Displace a Particle)

```
      SUBROUTINE mcmove                    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].

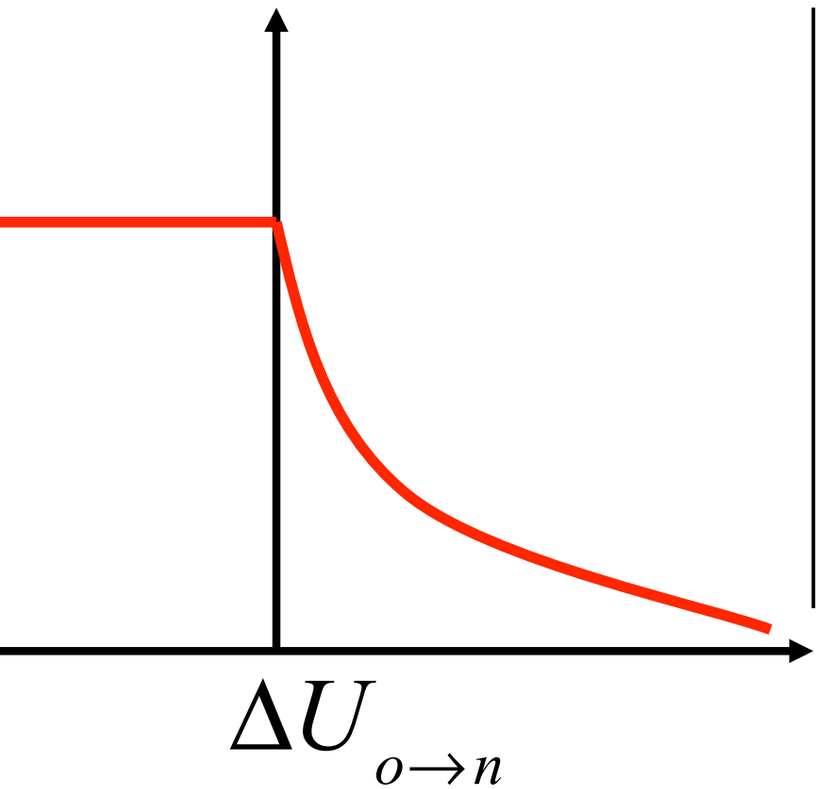
Metropolis et al.

Many acceptance rules that satisfy:

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

Metropolis *et al.* introduced:

$$\mathrm{acc}(o \to n) = \min\left(0, e^{-\beta\left[U(n)-U(o)\right]}\right) = \min\left(0, e^{-\beta \Delta U}\right)$$



If: $\Delta U < 0 \quad \mathrm{acc}(o \to n) = 1$

accept the move

If: $\Delta U > 0 \quad \mathrm{acc}(o \to n) = e^{-\beta \Delta U}$

draw a uniform random number [0;1] and accept the new configuration if:

$$ranf < e^{-\beta \Delta U}$$

3.Monte Carlo Simulation

3.4.2 Particle 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`?

Detailed

Balance

3.Monte Carlo Simulation

3.4.3 Selecting the 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`?

Algorithm 2 (Attempt to Displace a Particle)

```
      SUBROUTINE mcmove                    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].

Mathematical

Transition probability from $o \rightarrow n$:

$$\pi\left(o \rightarrow n\right) = \alpha\left(o \rightarrow n\right) \times \mathrm{acc}\left(o \rightarrow n\right)$$

As by definition we make a transition:

$$\sum_n \pi\left(o \rightarrow n\right) = 1$$

The probability we do not make a move:

$$\pi\left(o \rightarrow o\right) = 1 - \sum_{n \neq 0} \pi\left(o \rightarrow n\right)$$

This term ≠ 0

Model

Let us take a spin system:

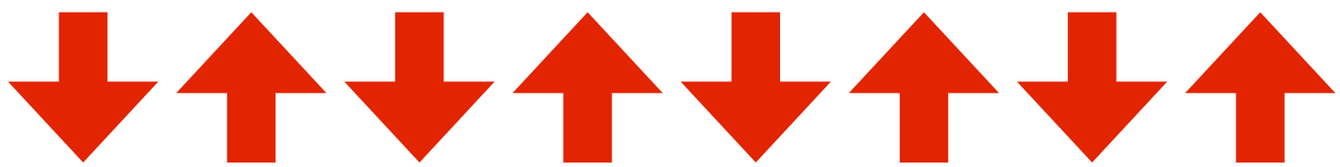
(with energy U↑ = +1 and U↓ = -1)

Probability to find↑:

$$P(\uparrow) = e^{-\beta U(\uparrow)}$$

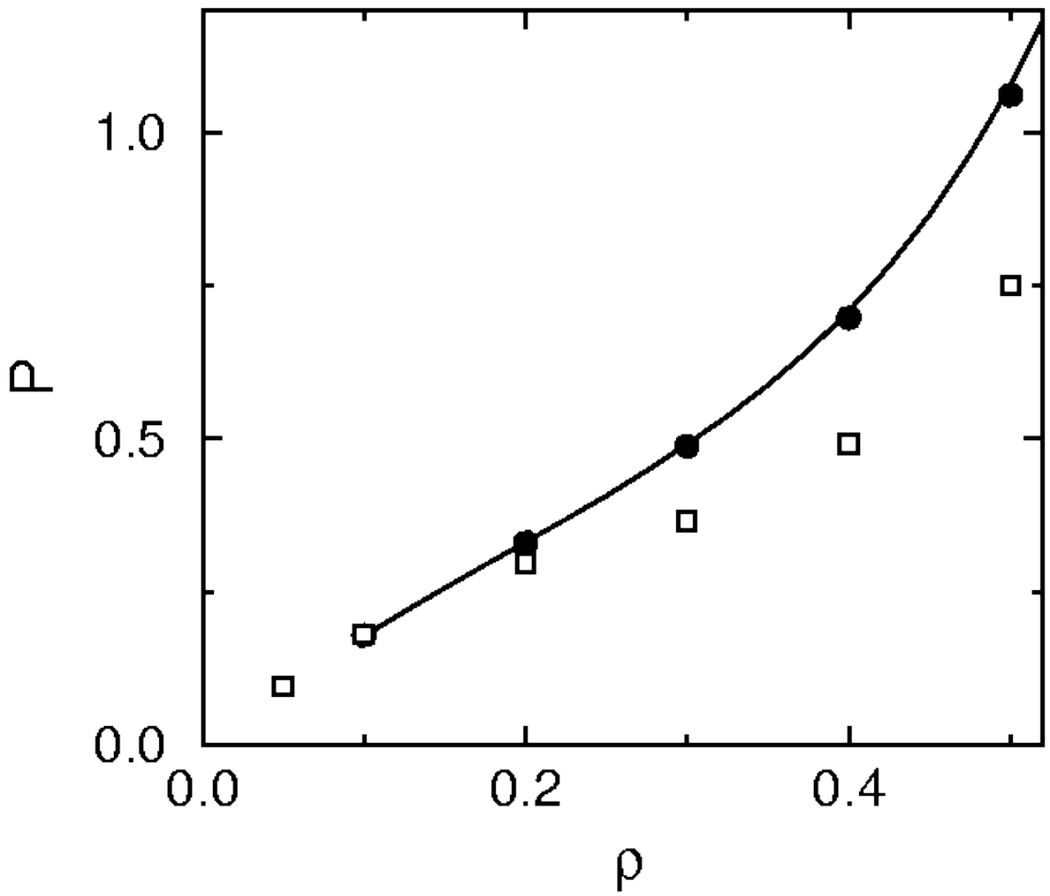
A possible configuration:

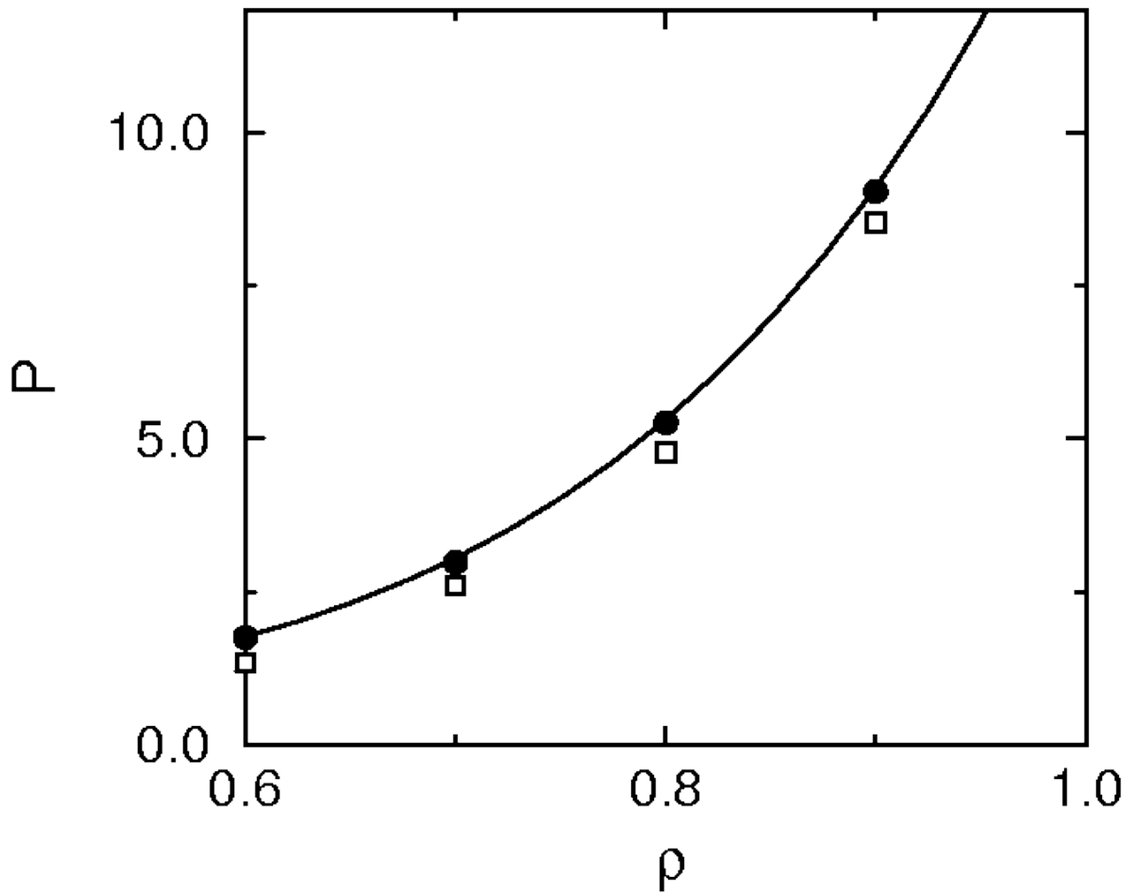
If we do not keep the old configuration:



(independent of the temperature)

Lennard Jones fluid





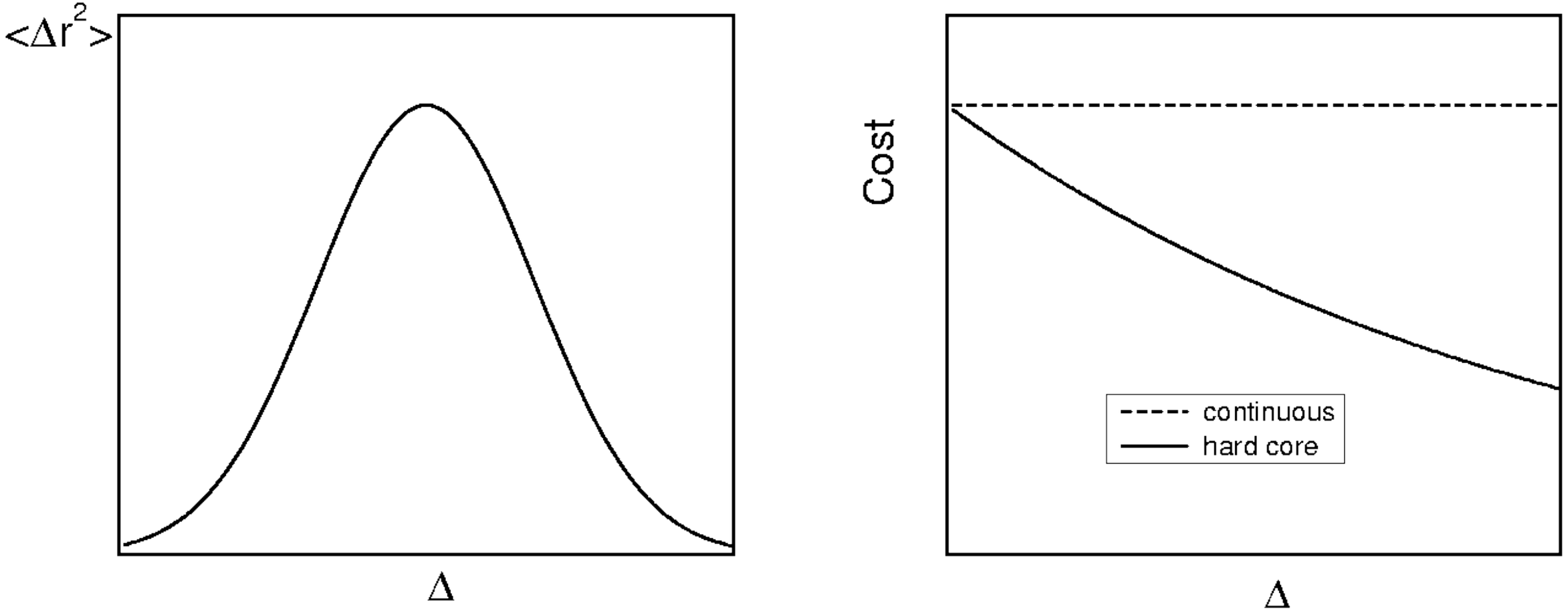
3.Monte Carlo Simulation

3.4.4 Particle displacement

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 big Not too small



3.Monte Carlo Simulation

3.5 Non-Boltzmann sampling

Non-Boltzmann sampling

$\beta_1 = 1/k_B T_1$

Ensemble average of A at temperature T_1:

$$\langle A \rangle_{NVT_1} = \frac{\int A(r) e^{-\beta_1 U(r)}\, dr}{\int e^{-\beta_1 U(r)}\, dr} \times \frac{1}{1}$$

with

$$\langle A \rangle_{NVT_1} = \frac{\int A(r) e^{-\beta_1 U(r)}\, dr}{\int e^{-\beta_1 U(r)}\, dr}$$

$$1 = e^{-\beta_2 [U(r) - U(r)]}$$

$$\langle A \rangle_{NVT_1} = \frac{\int A(r) e^{-\beta_1 U(r)} e^{-\beta_2 [U(r) - U(r)]}\, dr}{\int e^{-\beta_1 U(r)} e^{-\beta_2 [U(r) - U(r)]}\, dr}$$

$$\langle A \rangle_{NVT_1} = \frac{\int A(r) e^{-[\beta_1 U(r) - \beta_2 U(r)]} e^{-\beta_2 U(r)}\, dr}{\int e^{-[\beta_1 U(r) - \beta_2 U(r)]} e^{-\beta_2 U(r)}\, dr}$$

again multiply with 1/1:

$$= \frac{\int e^{-\beta_2 U(r)}\, dr \int A(r) e^{-[\beta_1 U(r) - \beta_2 U(r)]} e^{-\beta_2 U(r)}\, dr}{\int e^{-[\beta_1 U(r) - \beta_2 U(r)]} e^{-\beta_2 U(r)}\, dr \int e^{-\beta_2 U(r)}\, dr}$$

This gives us:

$$\langle A \rangle_{NVT_1} = \frac{\left\langle A e^{-(\beta_1 - \beta_2) U} \right\rangle_{NVT_2}}{\left\langle e^{-(\beta_1 - \beta_2) U} \right\rangle_{NVT_2}}$$

Non-Boltzmann sampling

Ensemble average of A at temperature T_1:

$$\langle A \rangle_{NVT_1} = \frac{\int A(r) e^{-\beta_1 U(r)} dr}{\int e^{-\beta_1 U(r)} dr}$$

Why are we not using this?

T_1 is arbitrary, we can use any value

$$\langle A \rangle_{NVT_1} = \frac{\int A(r) e^{-\beta_1 U(r)} e^{-\beta_2 [U(r)-U(r)]} dr}{\int e^{-\beta_1 U(r)} e^{-\beta_2 [U(r)-U(r)]} dr}$$

again multiply with 1/1:

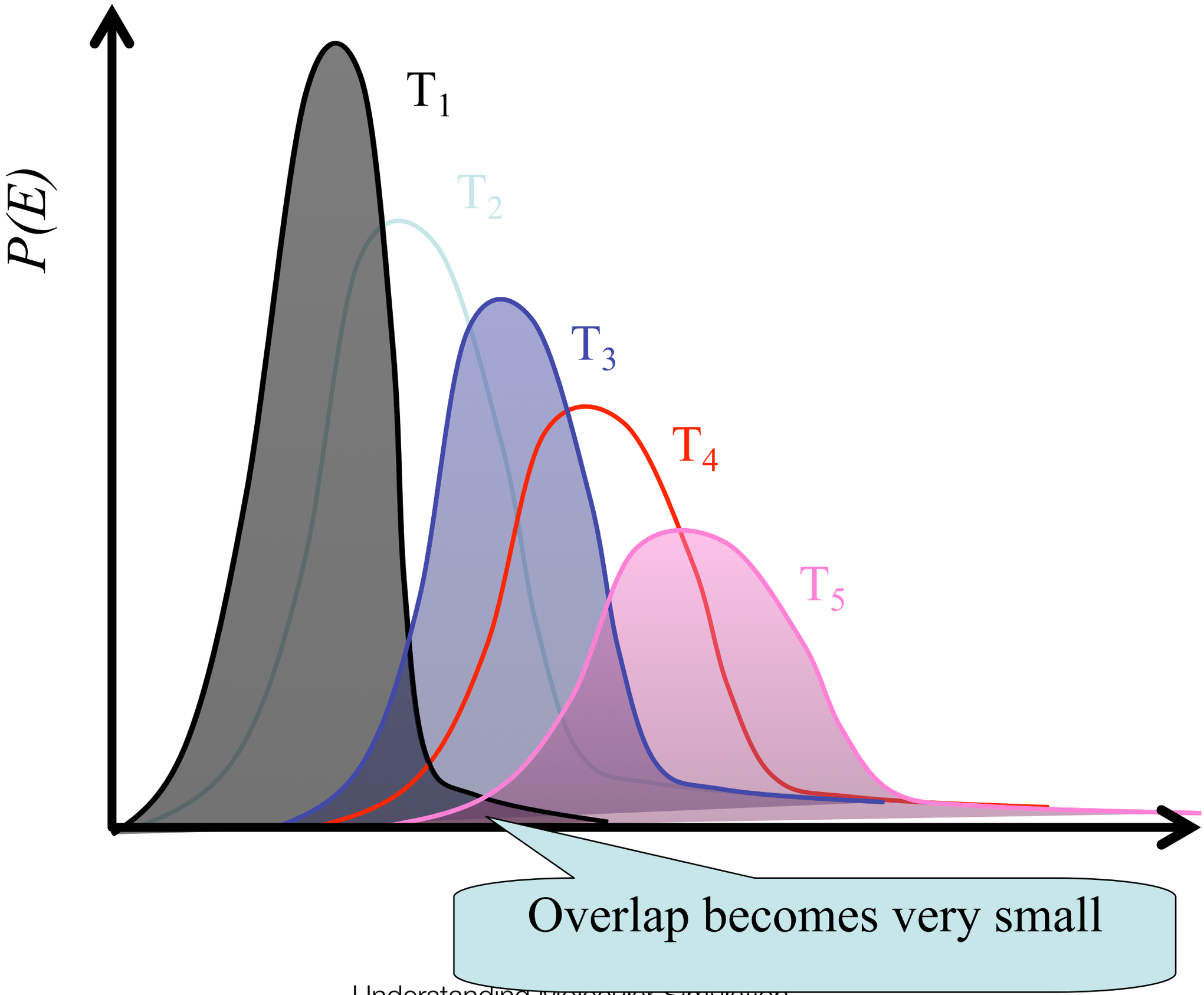
$$\langle A \rangle_{NVT_1} = \frac{\int A(r) e^{-[\beta_1 U(r)-\beta_2 U(r)]} e^{-\beta_2 U(r)} dr}{\int e^{-[\beta_1 U(r)-\beta_2 U(r)]} e^{-\beta_2 U(r)} dr} = \frac{\int e^{-\beta_2 U(r)} dr \int A(r) e^{-[\beta_1 U(r)-\beta_2 U(r)]} e^{-\beta_2 U(r)} dr}{\int e^{-\beta_2 U(r)} dr \int e^{-[\beta_1 U(r)-\beta_2 U(r)]} e^{-\beta_2 U(r)} dr}$$

We perform a simulation at T_2

and only 1 simulation …

$$\langle A \rangle_{NVT_1} = \frac{\left\langle A e^{-(\beta_1-\beta_2)U} \right\rangle_{NVT_2}}{\left\langle e^{-(\beta_1-\beta_2)U} \right\rangle_{NVT_2}}$$

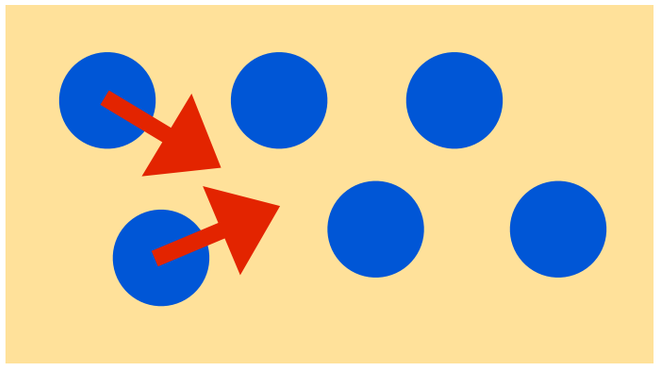
But obtain an ensemble average at T_1


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

3. Monte Carlo Simulation

3.6 Parallel Monte Carlo

Parallel Monte Carlo



How to do a Monte Carlo simulation in parallel?

- (trivial but works best) Use an ensemble of systems with different seeds for the random number generator
- Is it possible to do Monte Carlo in parallel?
  - Monte Carlo is sequential!
  - We first have to know the fait of the current move before we can continue!

Parallel Monte Carlo - algorithm

Naive (and wrong)

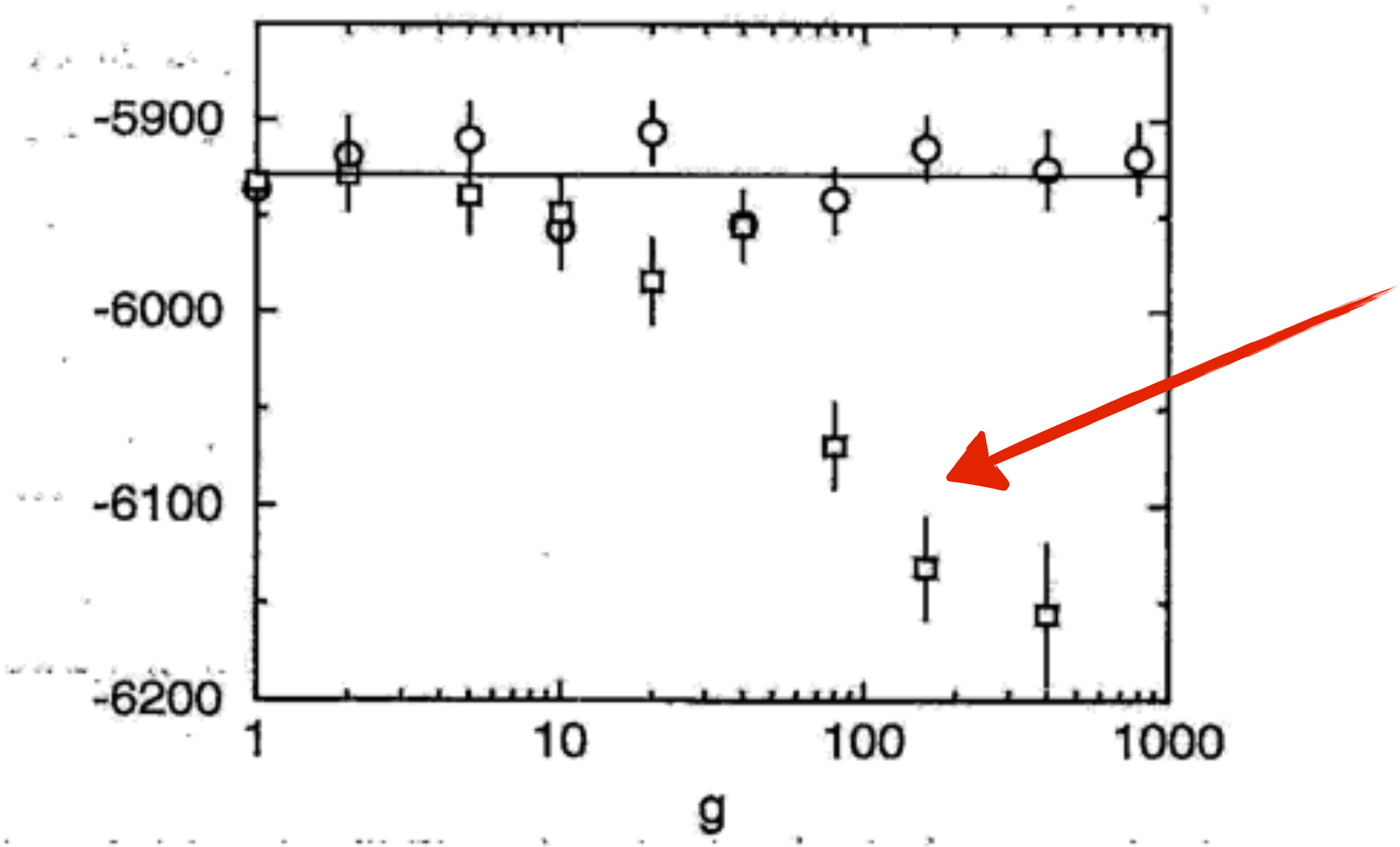
1. Generate k trial configurations in parallel
2. Select out of these the one with the lowest energy

$$P(n) = \frac{e^{-\beta U(n)}}{\sum_{j=1}^{g} e^{-\beta U(j)}}$$

3. Accept and reject using normal Monte Carlo rule:

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

Conventional acceptance rules



The conventional acceptance rules give a bias

What went wrong?

Detailed balance!

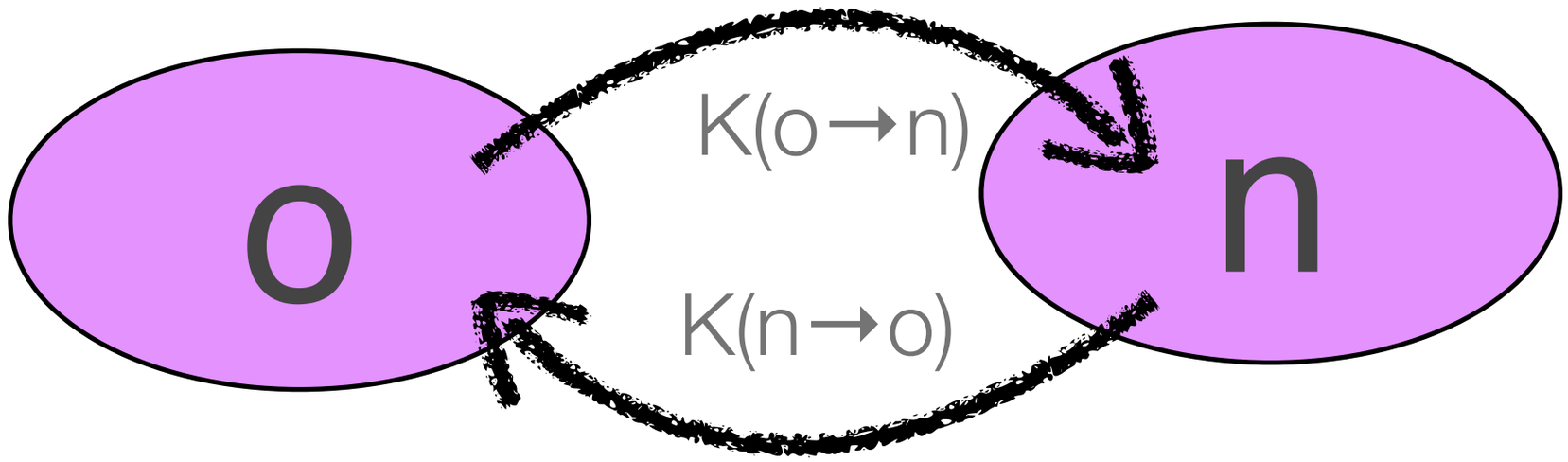
$$K(o \to n) = K(n \to o)$$

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

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

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

Markov Processes - Detailed Balance



Condition of detailed balance:

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

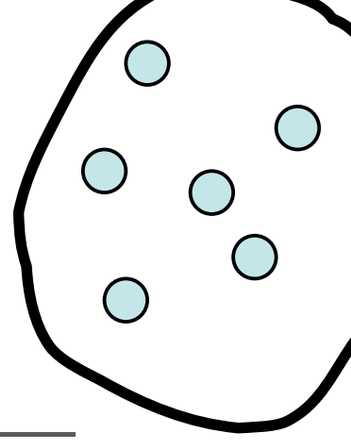
$$K(o \to n) = N(o) \times \alpha(o \to n) \times \mathrm{acc}(o \to n)$$

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

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

?

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



A priori probability to generate configuration n:

$$\alpha(o \to n) = \frac{e^{-\beta U(n)}}{\sum_{j=1}^{g} e^{-\beta U(j)}}$$

Rosenbluth factor configuration n:

$$W(n) = \sum_{j=1}^{g} e^{-\beta U(j)}$$

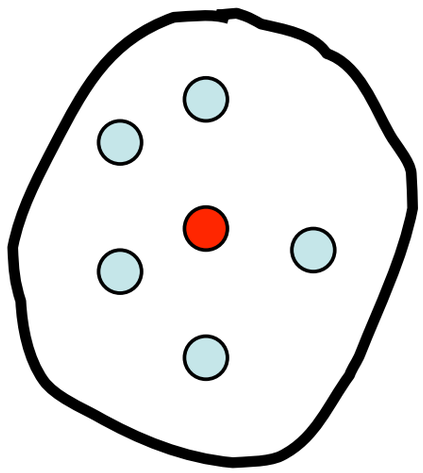
$$\alpha(o \to n) = \frac{e^{-\beta U(n)}}{W(n)}$$

A priori probability to generate configuration o:

$$\alpha(n \to o) = \frac{e^{-\beta U(o)}}{\sum_{j=1}^{g} e^{-\beta U(j)}}$$

Rosenbluth factor configuration o:

$$W(o) = e^{-\beta U(o)} + \sum_{j=1}^{g-1} e^{-\beta U(j)}$$



$$\alpha(n \to o) = \frac{e^{-\beta U(o)}}{W(o)}$$

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

Now with the correct a priori probabilities to generate a configuration:

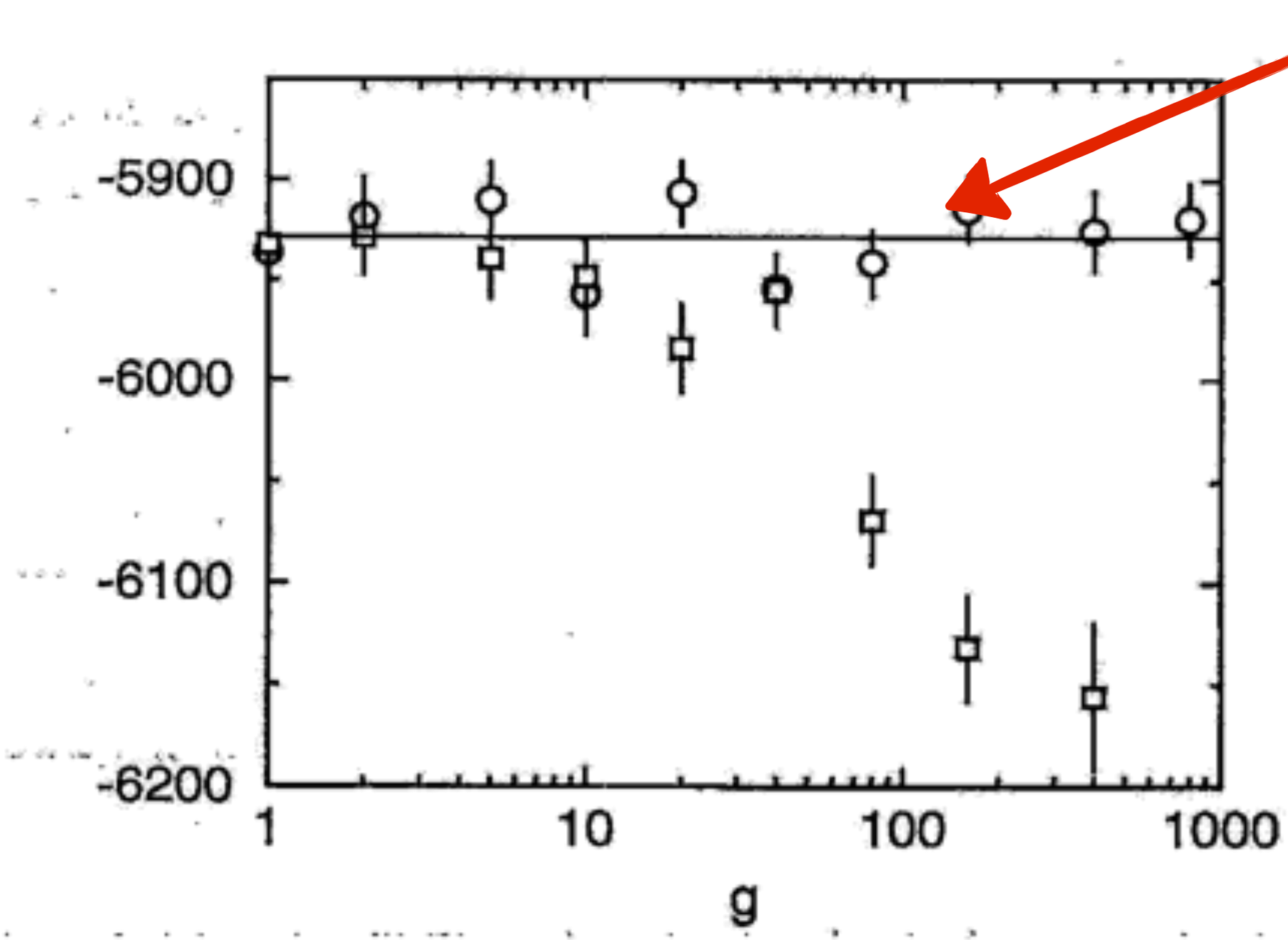
$$\alpha(o \to n) = \frac{e^{-\beta U(n)}}{W(n)}$$

$$\alpha(n \to o) = \frac{e^{-\beta U(o)}}{W(o)}$$

This gives as acceptance rules:

$$\frac{\mathrm{acc}(o \to n)}{\mathrm{acc}(n \to o)} = \frac{e^{-\beta U(n)} \times \dfrac{e^{-\beta U(o)}}{W(o)}}{e^{-\beta U(o)} \times \dfrac{e^{-\beta U(n)}}{W(n)}} = \frac{W(n)}{W(o)}$$

Conventional acceptance rules



Modified acceptance rules remove the bias exactly