

The background of the slide is a 3D molecular simulation. It features a complex network of orange and yellow rods representing atoms or molecules, with some spheres visible. The overall color scheme is dark with warm, glowing highlights from the simulation.

MOLECULAR SIMULATION

From Algorithms to Applications

second edition

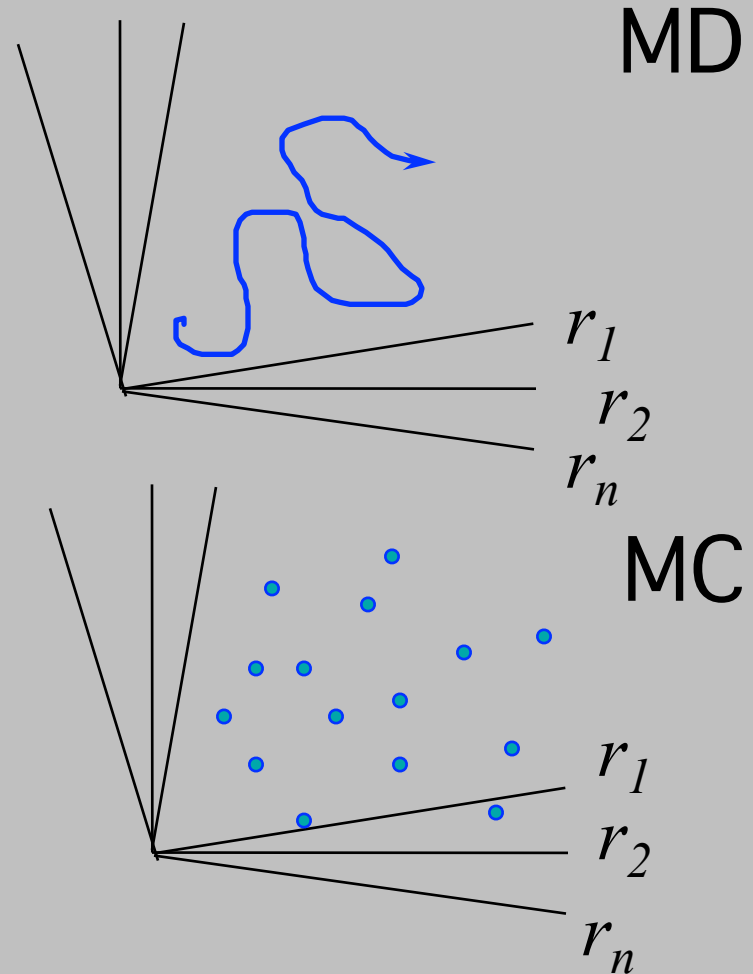
Introduction - Monte Carlo Simulations

Daan **Frenkel** & Berend **Smit**

Molecular Simulations

- ◆ Molecular dynamics:
solve equations of
motion

- ◆ Monte Carlo:
importance sampling



The background of the slide is a molecular simulation visualization. It features a complex network of orange and yellow rods and spheres, representing a molecular structure. The rods are connected at various points, forming a dense, interconnected network. The spheres are smaller and are attached to the rods, representing atoms or functional groups. The overall color scheme is warm, with shades of orange, yellow, and brown. The text is overlaid on this background.

MOLECULAR SIMULATION

From Algorithms to Applications

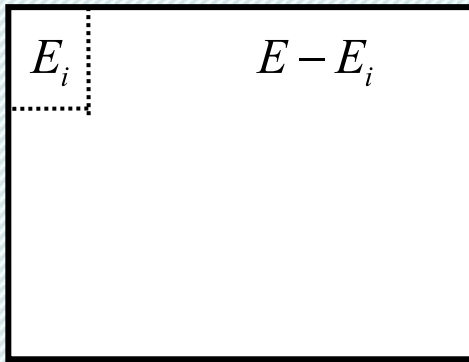
second edition

Statistical Thermodynamics

Daan **Frenkel** & Berend **Smit**

Canonical ensemble

$$1/k_B T$$



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

$$\ln \Omega(E - E_i) = \ln \Omega(E) - \frac{\partial \ln \Omega}{\partial E} E_i + \dots$$

$$\ln \frac{\Omega(E - E_i)}{\Omega(E)} = -\frac{E_i}{k_B T}$$

Hence, the probability to find E_i :

$$P(E_i) = \frac{\Omega(E - E_i)}{\sum_j \Omega(E - E_j)} = \frac{\exp(-E_i/k_B T)}{\sum_j \exp(-E_j/k_B T)}$$

$$P(E_i) \propto \exp(-E_i/k_B T)$$

Boltzmann distribution

Summary: Canonical ensemble (N, V, T)

Partition function:

$$Q(N, V, T) = \frac{1}{\Lambda^{3N} N!} \int d\mathbf{r}^N \exp[-\beta U(\mathbf{r}^N)]$$

Probability to find a particular configuration:

$$P(\Gamma) \propto \exp[-\beta U(\Gamma)]$$

Ensemble average:

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

Free energy:

$$\beta F = -\ln Q_{N,V,T}$$

The background of the slide is a 3D molecular simulation. It features a complex network of orange and yellow rods representing atoms or molecules, with some spheres visible. The overall color scheme is dark with warm, glowing highlights from the molecular structure.

MOLECULAR SIMULATION

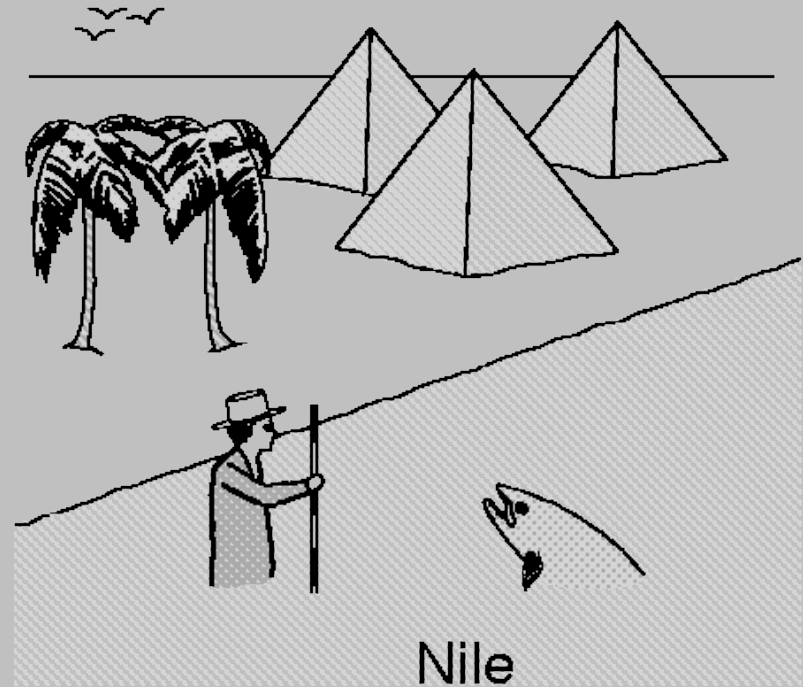
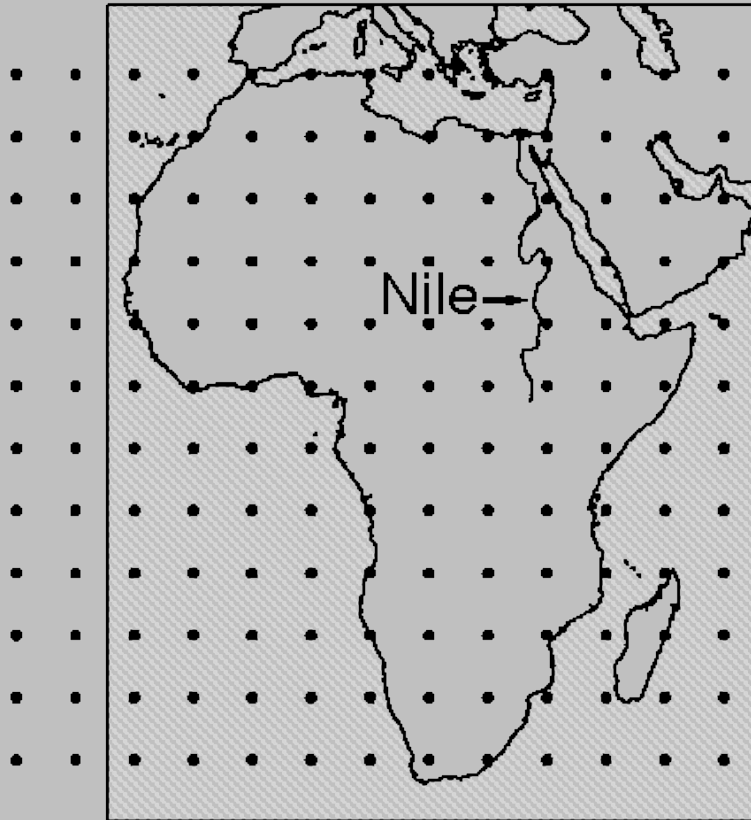
From Algorithms to Applications

second edition

Importance Sampling

Daan **Frenkel** & Berend **Smit**

Numerical Integration



Ensemble Average

$$\begin{aligned}\langle A \rangle_{NVT} &= \frac{1}{Q_{NVT}} \frac{1}{\Lambda^{3N} N!} \int d\mathbf{r}^N A(\mathbf{r}^N) \exp[-\beta U(\mathbf{r}^N)] \\ &= \frac{\int d\mathbf{r}^N A(\mathbf{r}^N) P(\mathbf{r}^N)}{\int d\mathbf{r}^N P(\mathbf{r}^N)} = \int d\mathbf{r}^N A(\mathbf{r}^N) P(\mathbf{r}^N)\end{aligned}$$

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

$$\begin{aligned}&= \frac{\int d\mathbf{r}^N A(\mathbf{r}^N) C \exp[-\beta U(\mathbf{r}^N)]}{\int d\mathbf{r}^N C \exp[-\beta U(\mathbf{r}^N)]} = \frac{\int d\mathbf{r}^N A(\mathbf{r}^N) \exp[-\beta U(\mathbf{r}^N)]}{\int d\mathbf{r}^N \exp[-\beta U(\mathbf{r}^N)]}\end{aligned}$$

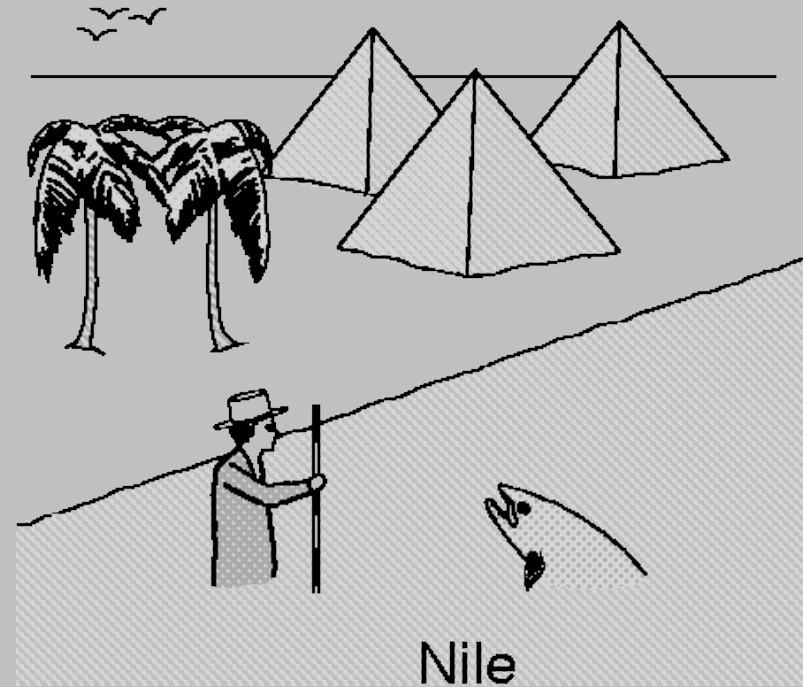
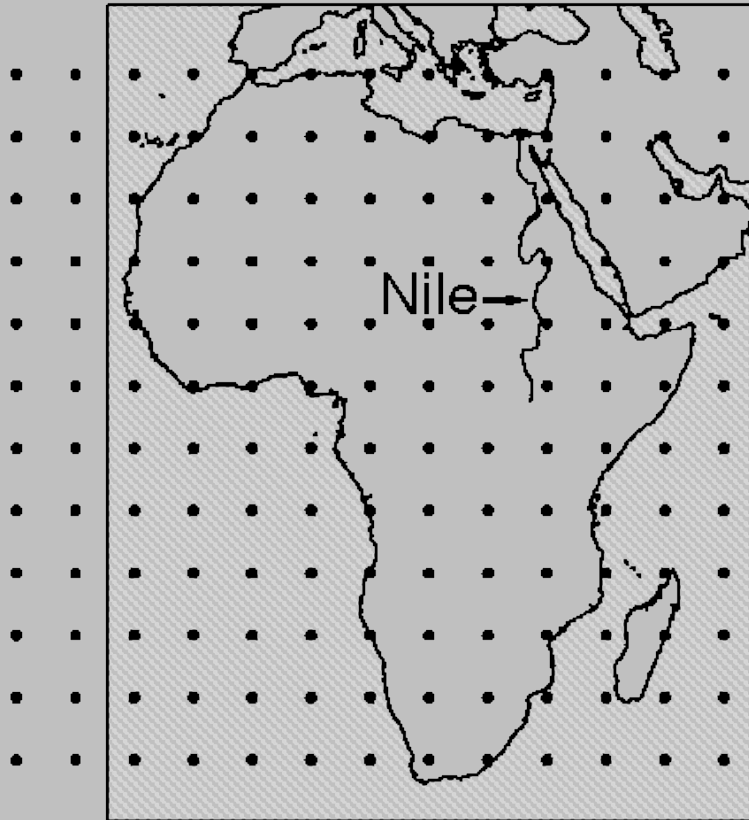
Generate configurations using Monte Carlo moves

$$\{\mathbf{r}_1^N, \mathbf{r}_2^N, \mathbf{r}_3^N, \mathbf{r}_4^N, \dots, \mathbf{r}_M^N\} \quad \bar{A} = \frac{1}{M} \sum_{i=1}^M A(\mathbf{r}_i^N) = \frac{\int d\mathbf{r}^N A(\mathbf{r}^N) P^{MC}(\mathbf{r}^N)}{\int d\mathbf{r}^N P^{MC}(\mathbf{r}^N)}$$

with:

$$\begin{aligned}P^{MC}(\mathbf{r}^N) &= C^{MC} \exp[-\beta U(\mathbf{r}^N)] \\ &= \frac{\int d\mathbf{r}^N A(\mathbf{r}^N) C^{MC} \exp[-\beta U(\mathbf{r}^N)]}{\int d\mathbf{r}^N C^{MC} \exp[-\beta U(\mathbf{r}^N)]} \\ &= \frac{\int d\mathbf{r}^N A(\mathbf{r}^N) \exp[-\beta U(\mathbf{r}^N)]}{\int d\mathbf{r}^N \exp[-\beta U(\mathbf{r}^N)]}\end{aligned}$$

Importance Sampling



Algorithm 1 (Basic Metropolis Algorithm)

<pre>PROGRAM mc</pre>	basic Metropolis algorithm
<pre>do icycl=1,ncycl</pre>	perform <code>ncycl</code> MC cycles
<pre> call mcmove</pre>	displace a particle
<pre> if (mod(icycl,nsamp).eq.0)</pre>	
<pre>+ call sample</pre>	sample averages
<pre>enddo</pre>	
<pre>end</pre>	

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)

<pre>SUBROUTINE mcmove o=int(ranf()*npart)+1 call ener(x(o), eno) xn=x(o)+(ranf()-0.5)*delx call ener(xn, enn) if (ranf().lt.exp(-beta + *(enn-eno)) x(o)=xn return end</pre>	<p>attempts to displace a particle</p> <p>select a particle at random energy old configuration give particle random displacement energy new configuration acceptance rule (3.2.1) accepted: replace $x(o)$ by xn</p>
---	--

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

Desired distribution: NVT ensemble

- 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: Δx ?

Markov Processes

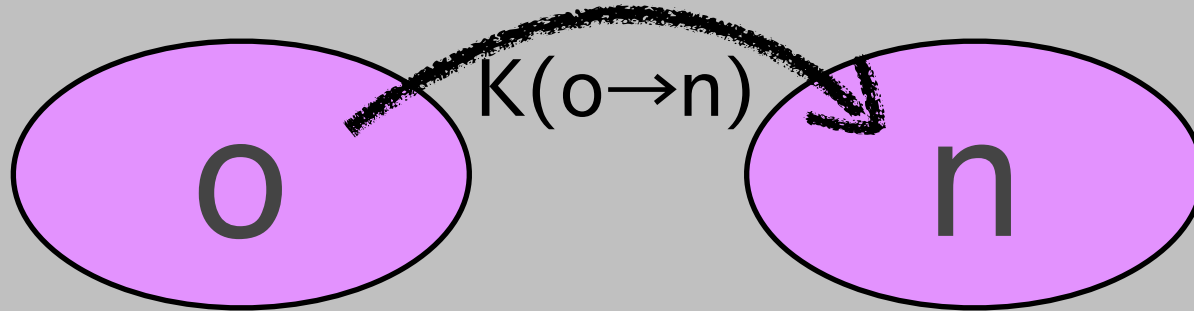
Markov Process

- 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

Ensemble - probability

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

Markov Processes - Detailed Balance

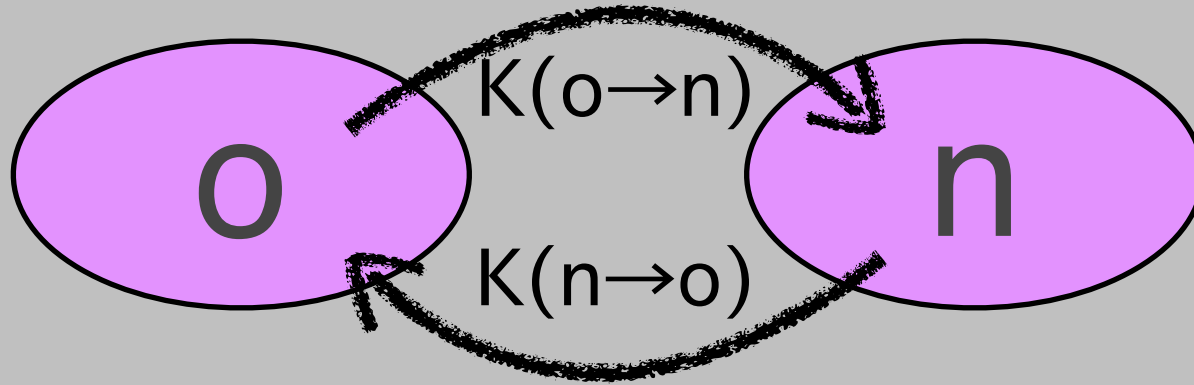


$K(o \rightarrow n)$: total number of systems in our ensemble that move $o \rightarrow n$

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

- $N(o)$: total number of systems in our ensemble in state o
- $\alpha(o \rightarrow n)$: a priori probability to generate a move $o \rightarrow n$
- $\text{acc}(o \rightarrow n)$: probability to accept the move $o \rightarrow n$

Markov Processes - Detailed Balance



Condition of detailed balance:

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

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

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

$$\frac{\text{acc}(o \rightarrow n)}{\text{acc}(n \rightarrow o)} = \frac{N(n) \times \alpha(n \rightarrow o)}{N(o) \times \alpha(o \rightarrow n)} = \frac{N(n)}{N(o)}$$

NVT-ensemble

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

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

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

Which gives as condition for the acceptance rule:

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

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
+ * (enn-eno)) x(o)=xn

acceptance rule (3.2.1)

accepted: replace x(o) by xn

return

end

Comments to this algorithm:

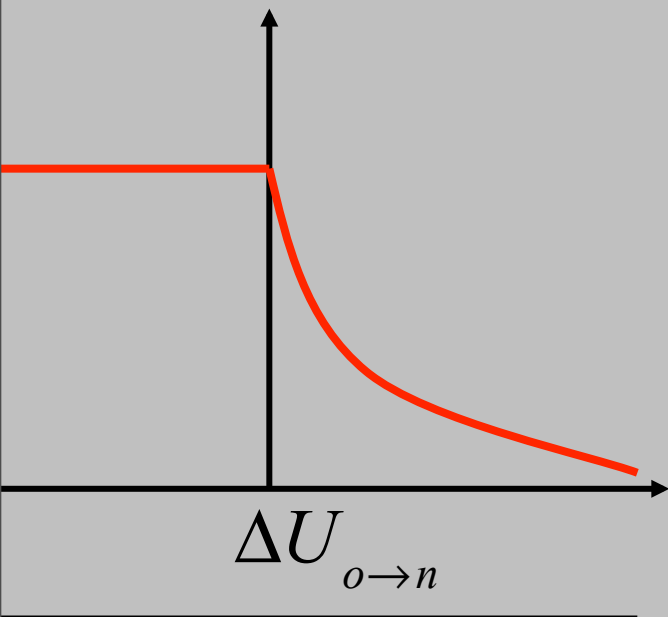
1. Subroutine ener calculates the energy of a particle at the given position.
2. Note that, if a configuration is rejected, the old configuration is retained.
3. The ranf() is a random number uniform in $[0, 1]$.

Metropolis et al.

Many acceptance rules that satisfy:

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

Metropolis *et al.* introduced:



$$\text{acc}(o \rightarrow n) = \min\left(1, \exp\left[-\beta\Delta U_{o \rightarrow n}\right]\right)$$

$$\text{If: } \Delta U_{o \rightarrow n} < 0$$

$$\text{acc}(o \rightarrow n) = 1$$

$$\Delta U_{o \rightarrow n} > 0$$

$$\text{acc}(o \rightarrow n) = \exp\left[-\beta\Delta U_{o \rightarrow n}\right]$$

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

$$\text{ranf} < \exp\left[-\beta\Delta U_{o \rightarrow n}\right]$$

The background of the slide is a 3D molecular simulation. It features a complex network of orange and yellow cylindrical rods, representing atoms or molecules, connected by bonds. The rods are arranged in a dense, interconnected pattern, with some rods extending towards the viewer and others receding into the background. The lighting is soft, creating a sense of depth and highlighting the geometric structure of the simulation.

MOLECULAR SIMULATION

From Algorithms to Applications

second edition

Monte Carlo Simulations

Daan **Frenkel** & Berend **Smit**

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: δx ?

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: δx ?

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 $o \rightarrow n$:

$$\pi(o \rightarrow n) = \alpha(o \rightarrow n) \times \text{acc}(o \rightarrow n)$$

As by definition we make a transition:

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

The probability we do not make a move:

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

This term
is $\neq 0$

Model

Let us take a spin system



With energy $U\uparrow = +1$ and $U\downarrow = -1$

$$P(\uparrow) \propto \exp\left(-\frac{U\uparrow}{k_B T}\right)$$

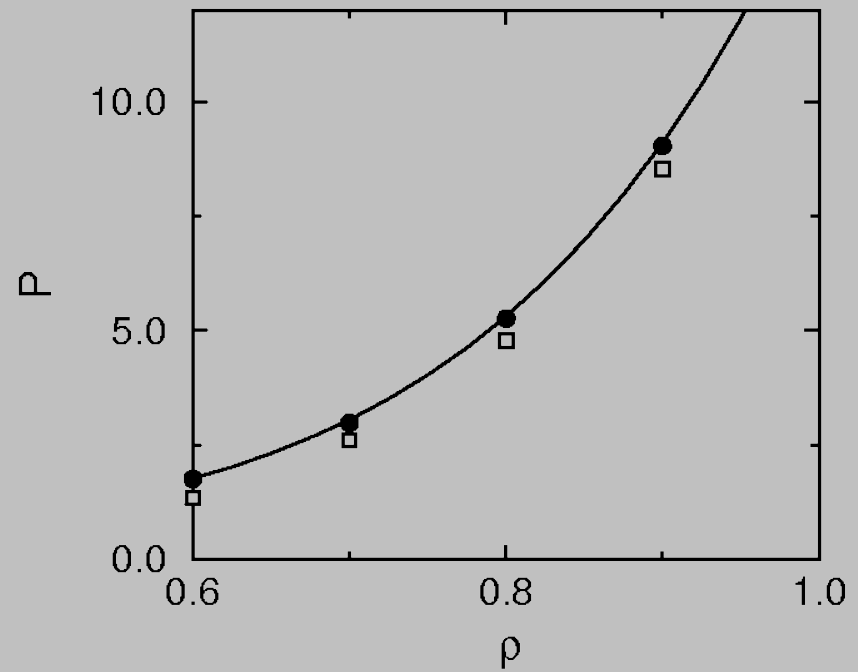
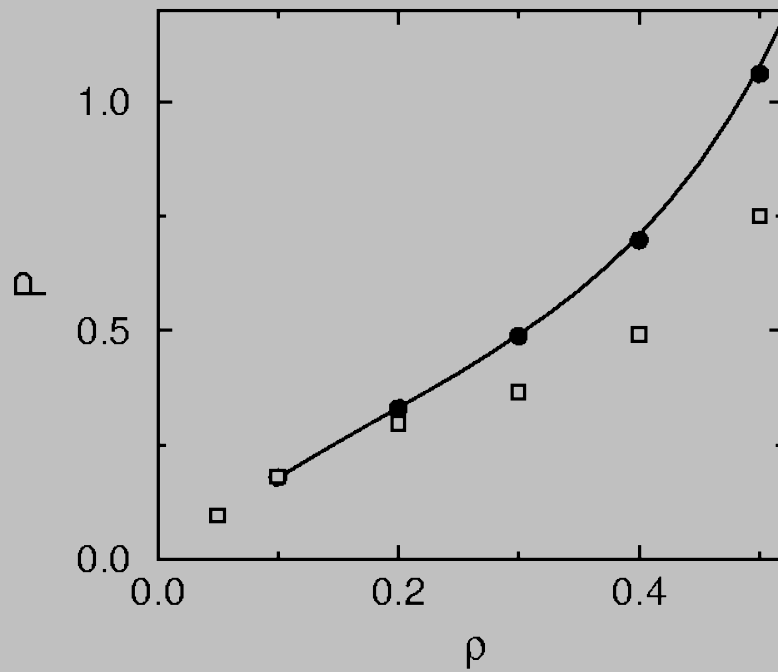


If we do not keep the old configuration:



Independent of the
temperature

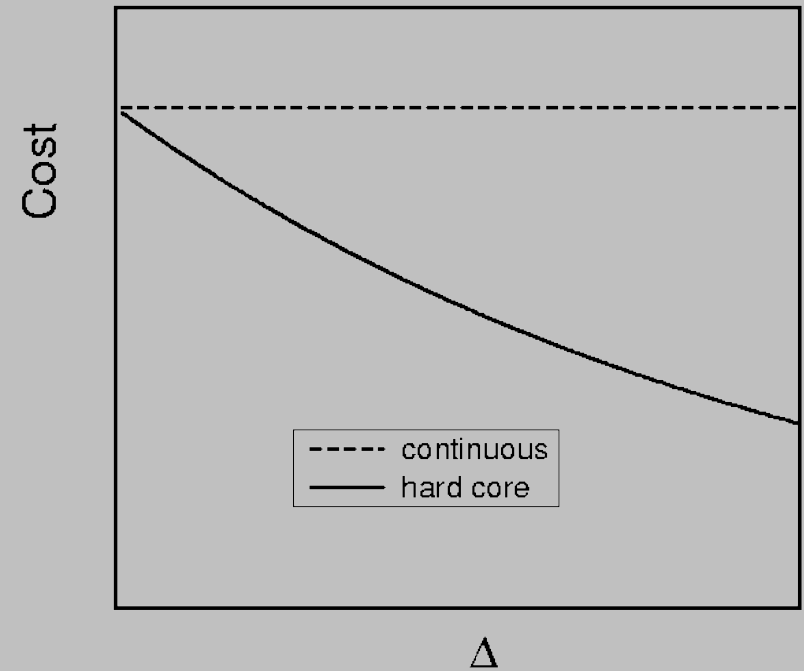
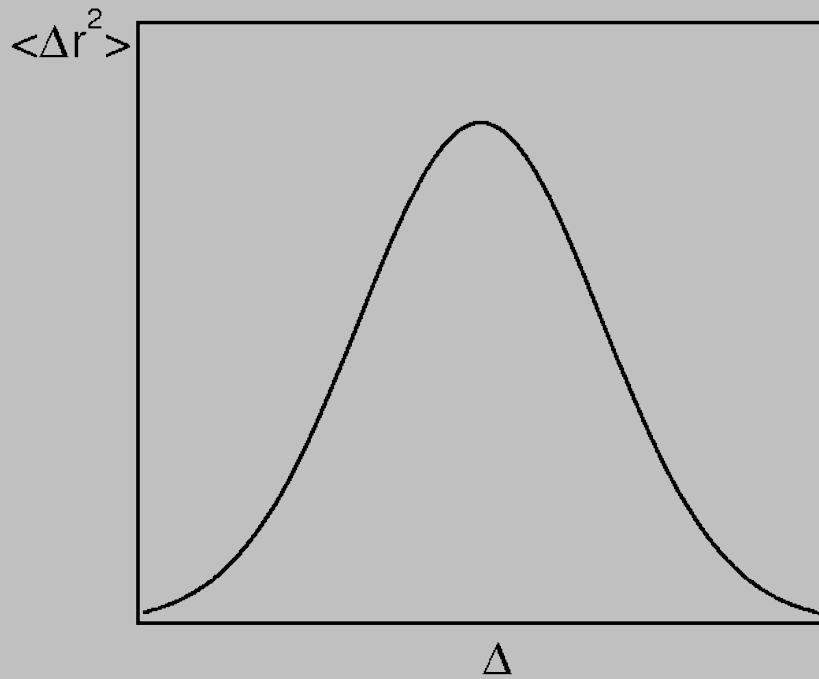
Lennard Jones fluid



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: Δx ?**

Not too big Not too small



The background of the slide is a molecular simulation visualization. It features a complex network of orange and yellow cylindrical rods, representing molecular chains or polymers, set against a dark blue background. The rods are interconnected, forming a dense, branching structure. In the upper left corner, there is a small, semi-transparent sphere with a black outline, possibly representing a specific atom or a simulation artifact.

MOLECULAR SIMULATION

From Algorithms to Applications

second edition

non-Boltzmann Sampling and Bias

Daan **Frenkel** & Berend **Smit**

Non-Boltzmann sampling

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

T_1 is arbitrary

$$= \frac{\int dr^N A(r^N) \exp[-\beta_1 U(r^N)]}{\int dr^N \exp[-\beta_1 U(r^N)]}$$

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

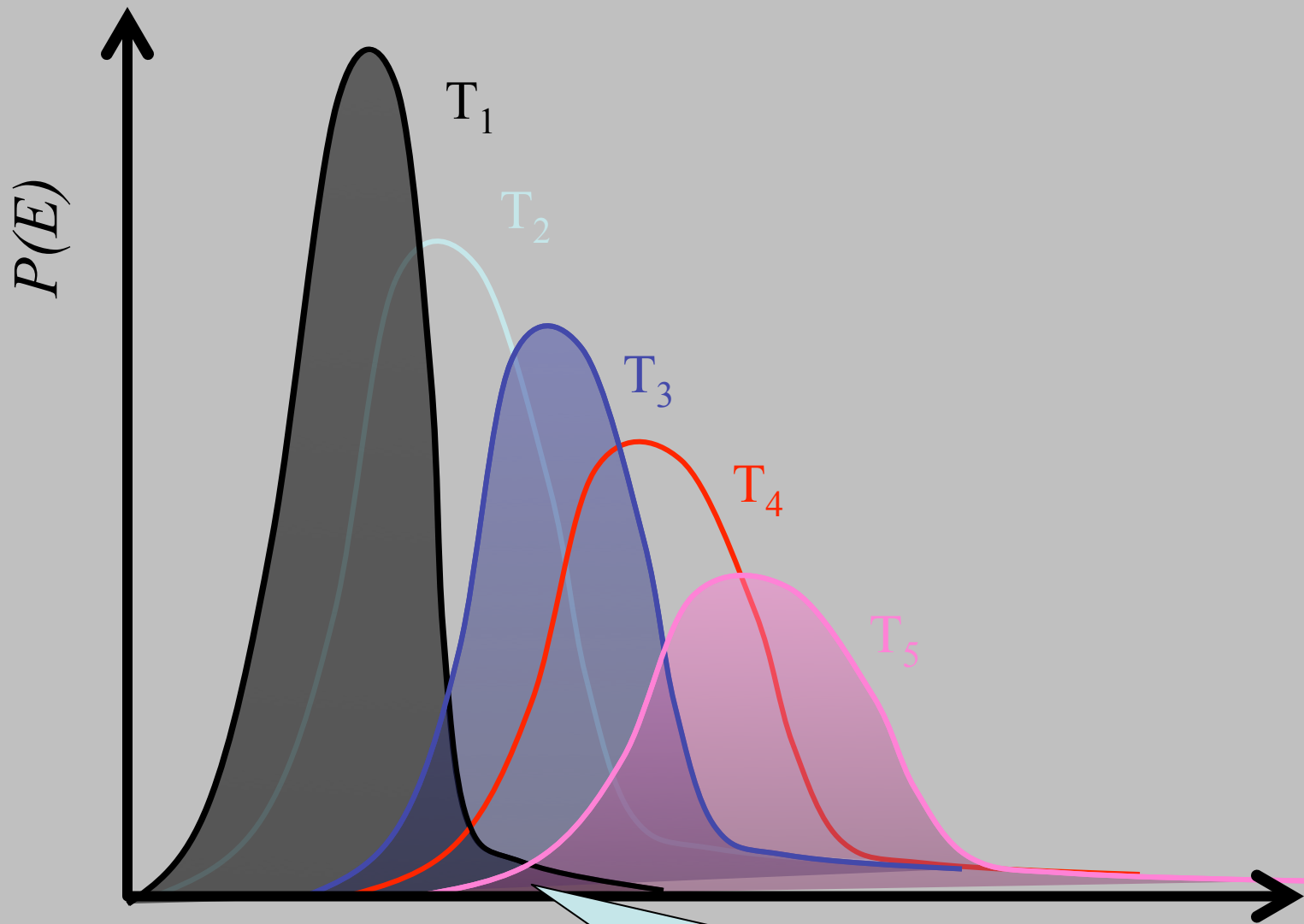
$$= \frac{\int dr^N A(r^N) \exp[-\beta_1 U(r^N)] \exp[\beta_2 U(r^N) - \beta_2 U(r^N)]}{\int dr^N \exp[-\beta_1 U(r^N)] \exp[\beta_2 U(r^N) - \beta_2 U(r^N)]}$$

Why are we not using this?

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

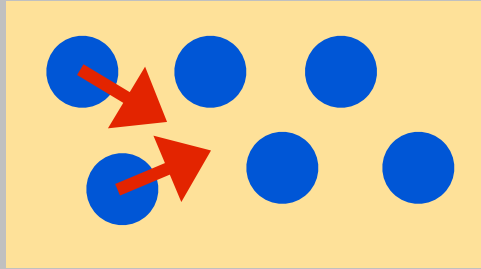
$$= \frac{\langle A \exp[(\beta_2 - \beta_1)U] \rangle_{NVT_2}}{\langle \exp[(\beta_2 - \beta_1)U] \rangle_{NVT_2}}$$

We only need a **single** simulations



Overlap becomes very small

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 fate of the current move before we can continue!

Parallel Monte Carlo - algorithm

Naive (and wrong)

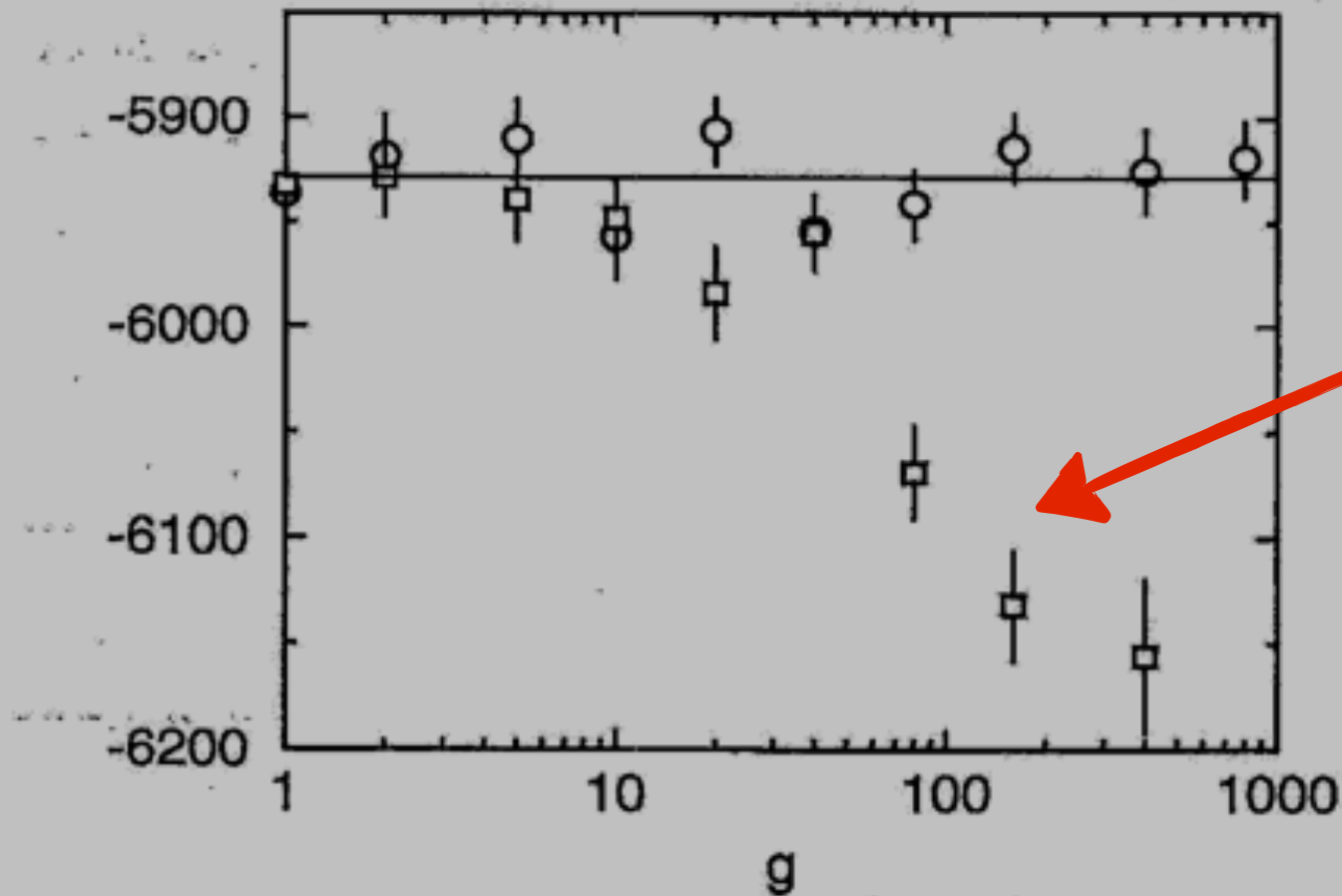
1. Generate k trial configurations in parallel
2. Select out of these the one with the lowest energy

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

3. Accept and reject using normal Monte Carlo rule:

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

Conventional acceptance rules



The conventional acceptance rules give a bias

What went wrong?

Detailed balance!

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

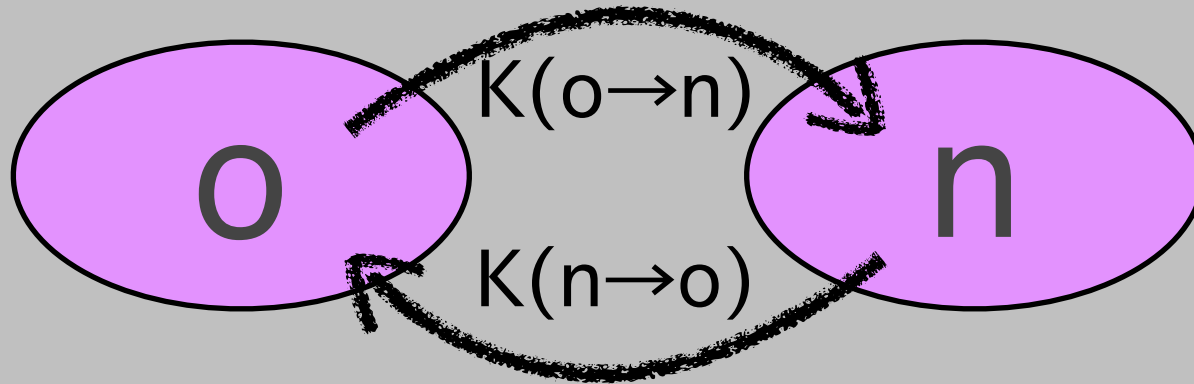
$$K(o \rightarrow n) = N(o) \times \alpha(o \rightarrow n) \times \text{acc}(o \rightarrow n)$$

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

$$\frac{\text{acc}(o \rightarrow n)}{\text{acc}(n \rightarrow o)} = \frac{N(n) \times \alpha(n \rightarrow o)}{N(o) \times \alpha(o \rightarrow n)} = \frac{N(n)}{N(o)}$$



Markov Processes - Detailed Balance



Condition of detailed balance:

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

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

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

$$\frac{\text{acc}(o \rightarrow n)}{\text{acc}(n \rightarrow o)} = \frac{N(n) \times \cancel{\alpha(n \rightarrow o)}}{N(o) \times \cancel{\alpha(o \rightarrow n)}} = \frac{N(n)}{N(o)}$$

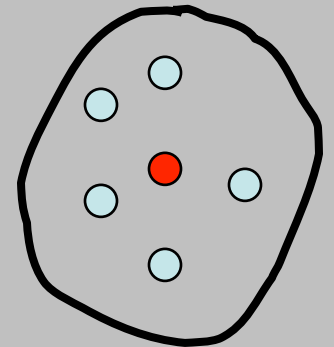
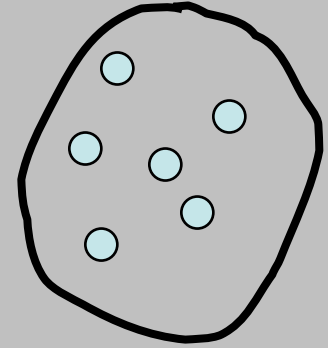
$$K(o \rightarrow n) = N(o) \times \alpha(o \rightarrow n) \times \text{acc}(o \rightarrow n)$$

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

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

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

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

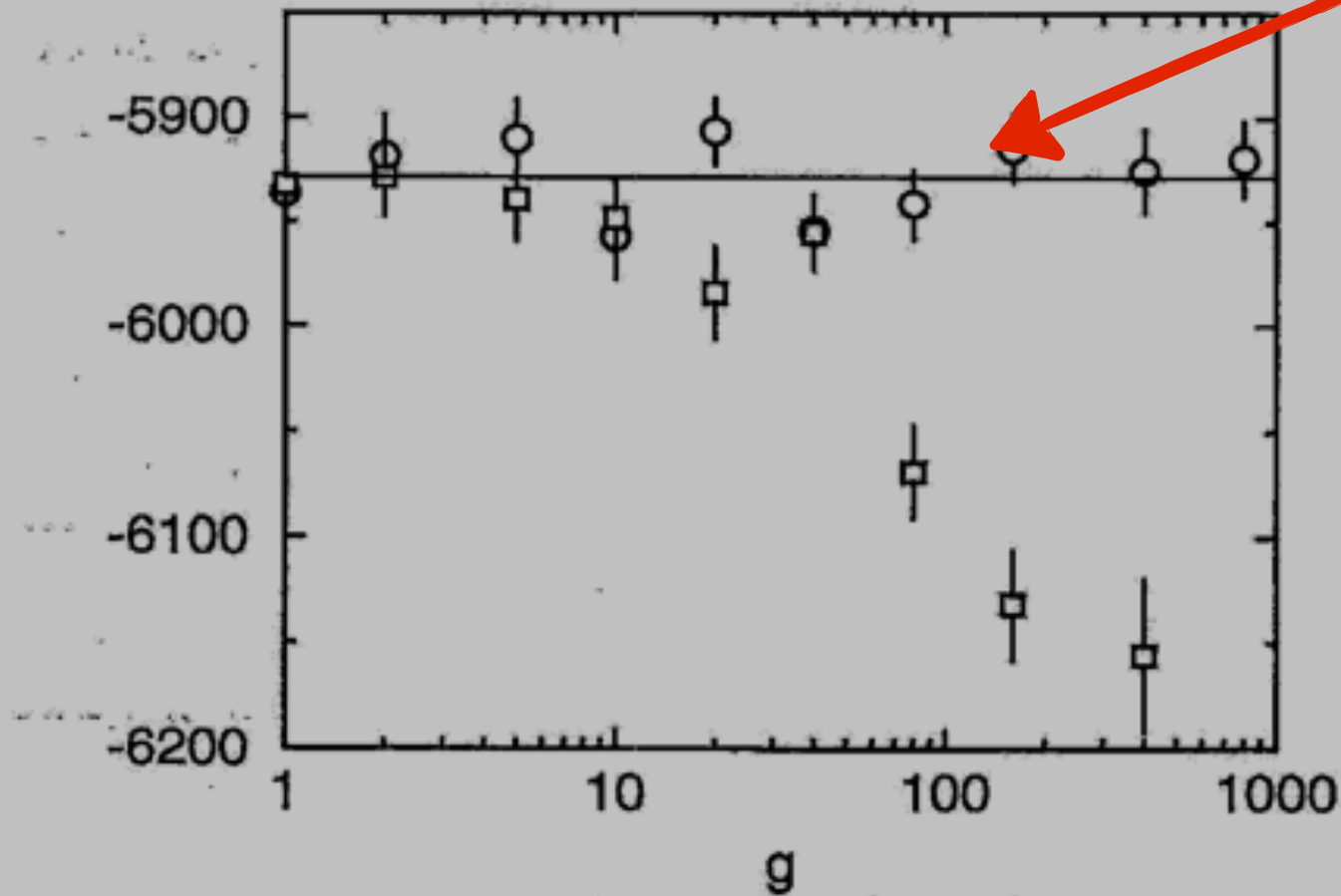


$$\frac{\text{acc}(o \rightarrow n)}{\text{acc}(n \rightarrow o)} = \frac{N(n) \times \alpha(n \rightarrow o)}{N(o) \times \alpha(o \rightarrow n)} = \frac{N(n)}{N(o)}$$

$$\alpha(o \rightarrow n) = \frac{\exp[-\beta(U_n)]}{W(\textcolor{red}{n})} \quad \alpha(n \rightarrow o) = \frac{\exp[-\beta(U_o)]}{W(\textcolor{red}{o})}$$

$$\frac{\text{acc}(o \rightarrow n)}{\text{acc}(n \rightarrow o)} = \frac{N(n) \times \frac{\exp[-\beta(U_o)]}{W(\textcolor{red}{o})}}{N(o) \times \frac{\exp[-\beta(U_n)]}{W(\textcolor{red}{n})}} = \frac{W(n)}{W(o)}$$

Conventional acceptance rules



Modified acceptance rules remove the bias exactly