CE 530 Molecular Simulation

Lecture 25

Efficiencies and Parallel Methods

David A. Kofke
Department of Chemical Engineering
SUNY Buffalo
kofke@eng.buffalo.edu

Look-Up Tables

- Evaluation of interatomic potential can be time-consuming
  - *For example, consider the exp-6 potential*

$$u(r) = -\frac{A}{r^6} + Be^{-Cr}$$

  - *Requires a square root and an exponential*

- Simple idea:
  - *Precompute a table of values at the beginning of the simulation and use it to evaluate the potential via interpolation*

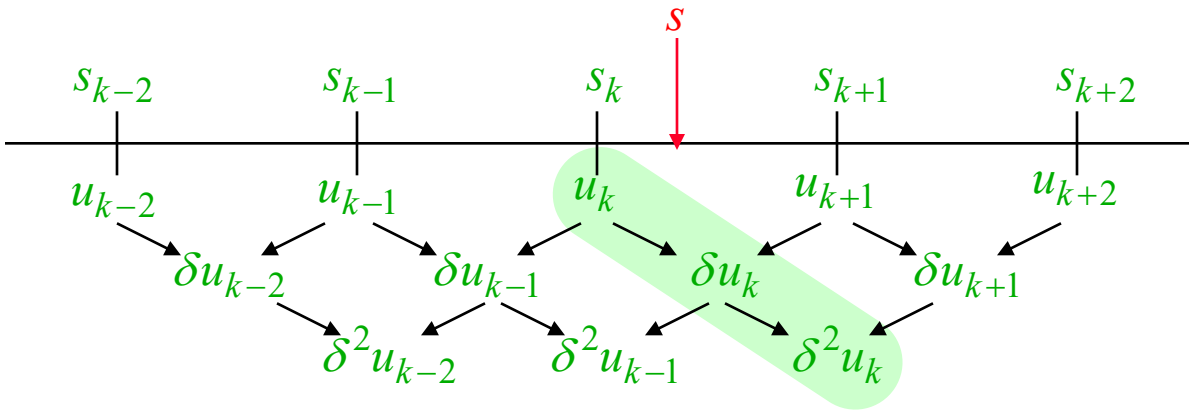
Interpolation

- Many interpolation schemes could be used
- e.g., Newton-Gregory forward difference method
  - *equally spaced values* δs *of* $s = r^2$
  - *given* $u_1 = u(s_1)$, $u_2 = u(s_2)$, *etc.*
  - *define first difference and second difference*

$$\delta u_k = u_{k+1} - u_k \qquad \delta^2 u_k = \delta u_{k+1} - \delta u_k$$

  - *to get* $u(s)$ *for* $s_k < s < s_{k+1}$, *interpolate*

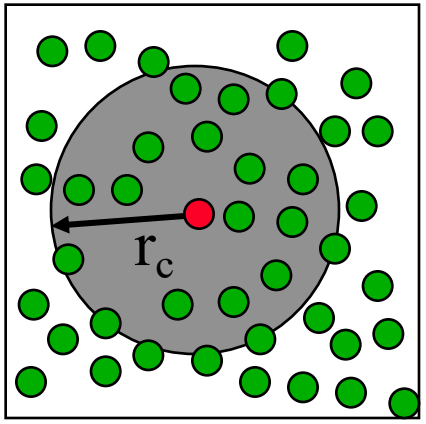
$$u(s) \approx u_k + \xi \delta u_k + \tfrac{1}{2}\xi(\xi - 1)\delta^2 u_k \qquad \xi = (s - s_k)/\delta s$$



  - *forces, virial can be obtained using finite differences*

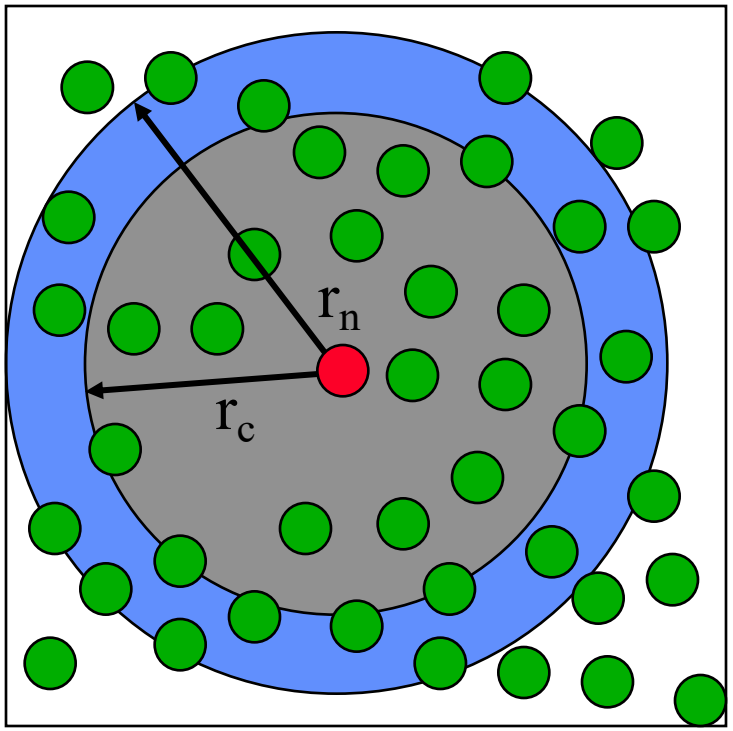
Finding Neighbors Efficiently

- Evaluation of all pair interactions is an $O(N^2)$ calculation
- Very expensive for large systems
- Not all interactions are relevant
  - *potential attenuated or even truncated beyond some distance*
- Worthwhile to have efficient methods to locate neighbors of any molecule
- Two common approaches
  - *Verlet neighbor list*
  - *Cell list*



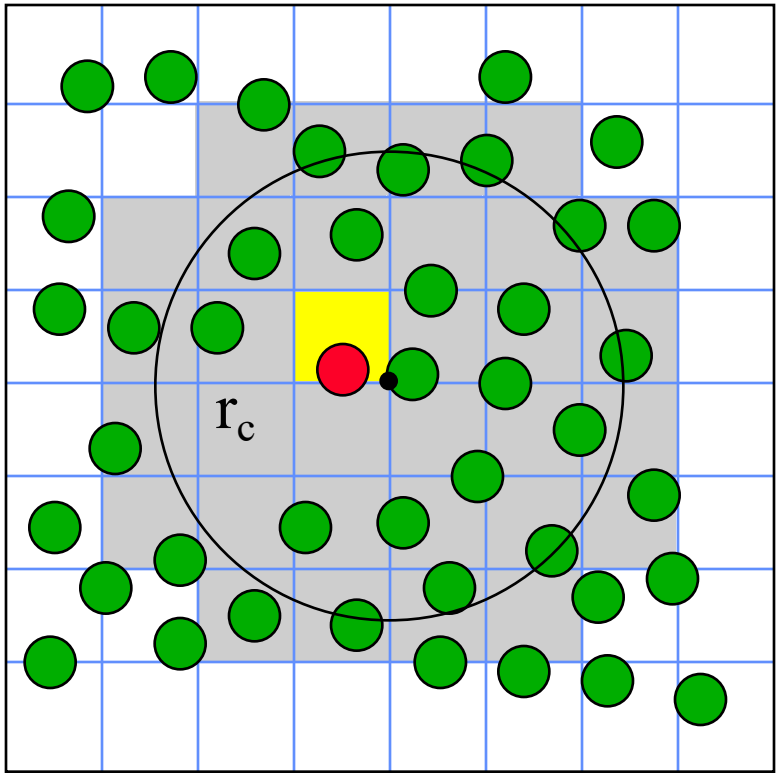
Verlet List

- Maintain a list of neighbors
  - *Set neighbor cutoff radius as potential cutoff plus a "skin"*
- Update list whenever a molecule travels a distance greater than the skin thickness
- Energy calculation is O(N)
- Neighbor list update is $O(N^2)$
  - *but done less frequently*



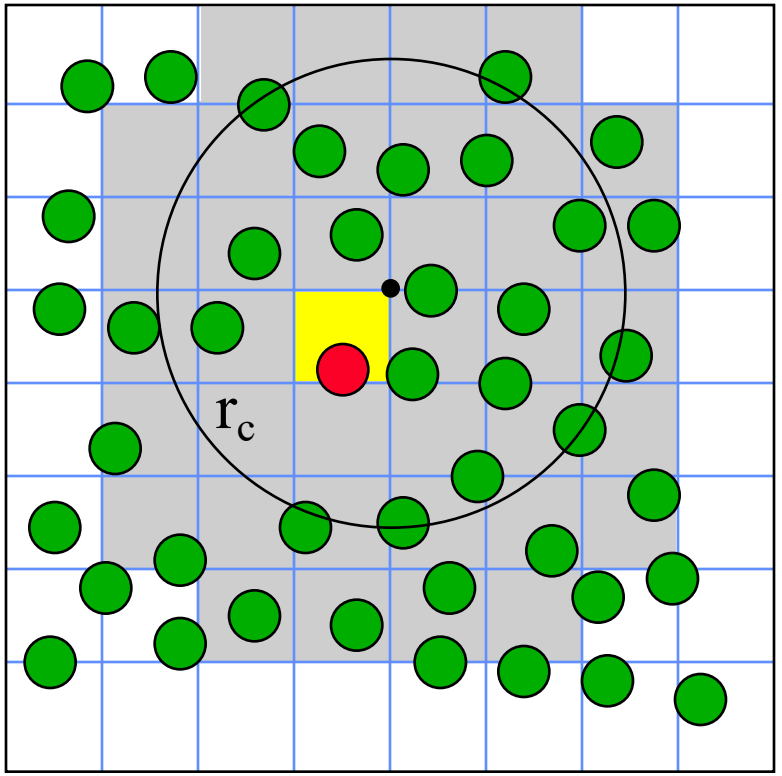
Cell List

- Partition volume into a set of cells
- Each cell keeps a list of the atoms inside it
- At beginning of simulation set up neighbor list for each cell
  - *list never needs updating*



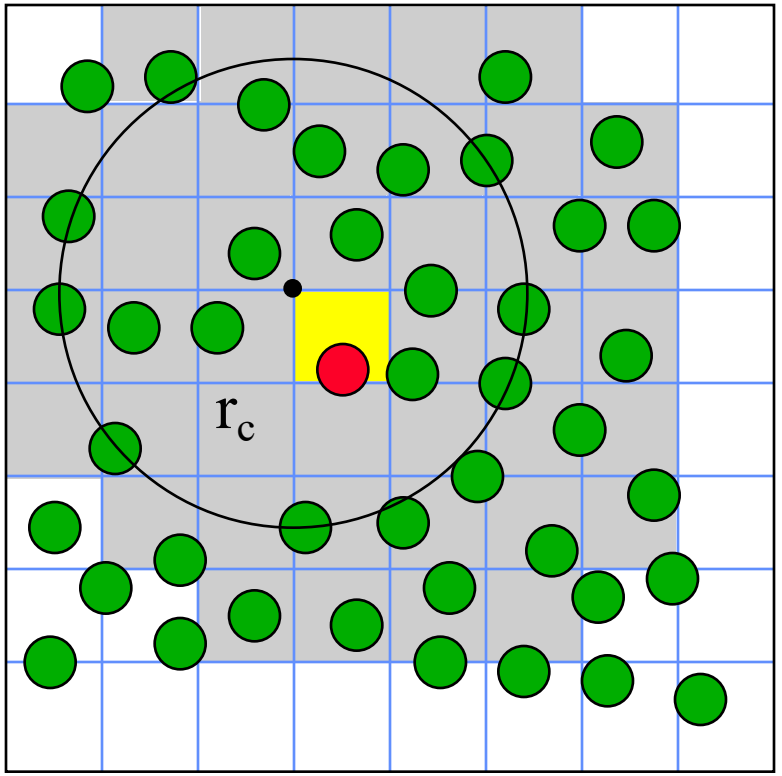
Cell List

- Partition volume into a set of cells
- Each cell keeps a list of the atoms inside it
- At beginning of simulation set up neighbor list for each cell
  - *list never needs updating*



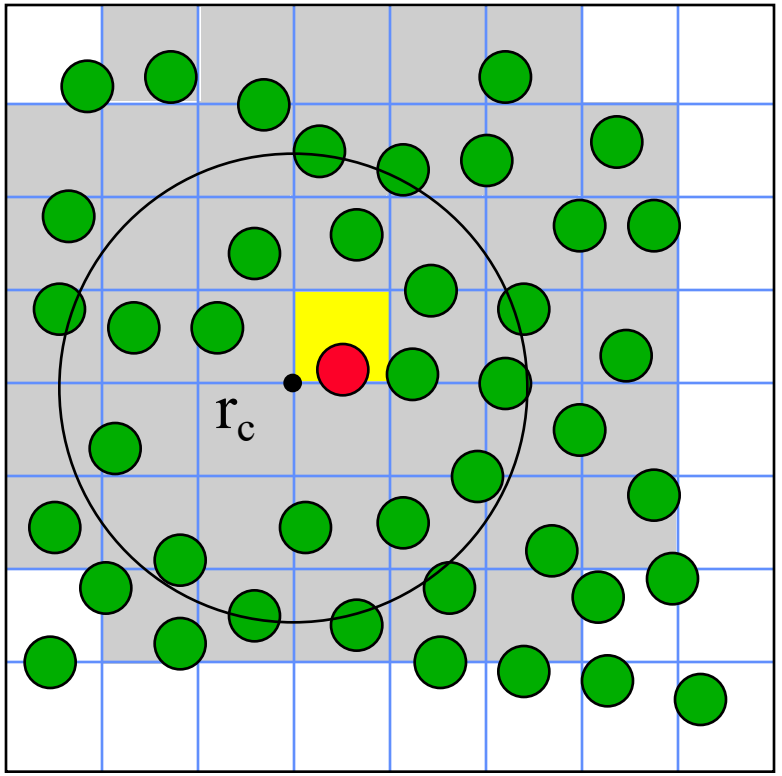
Cell List

- Partition volume into a set of cells
- Each cell keeps a list of the atoms inside it
- At beginning of simulation set up neighbor list for each cell
  - *list never needs updating*



Cell List

- Partition volume into a set of cells
- Each cell keeps a list of the atoms inside it
- At beginning of simulation set up neighbor list for each cell
  - *list never needs updating*
- Fewer unneeded pair interactions for smaller cells

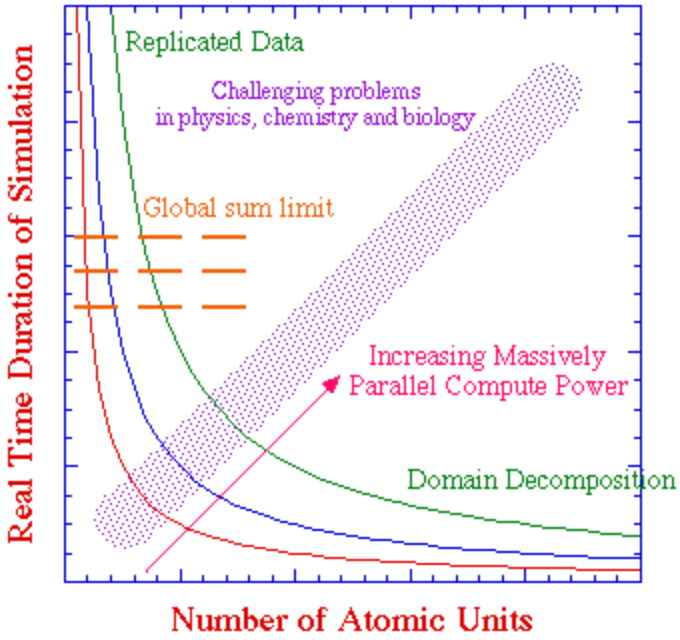


Parallelizing Simulation Codes

- Two parallelization strategies
  - *Domain decomposition*
    - Each processor focuses on fixed region of simulation space (cell)
    - Communication needed only with adjacent-cell processors
    - Enables simulation of very large systems for short times
  - *Replicated data*
    - Each processor takes some part in advancing all molecules
    - Communication among all processors required
    - Enables simulation of small systems for longer times

Limitations on Parallel Algorithms

- Straightforward application of raw parallel power insufficient to probe most interesting phenomena



- Advances in theory and technique needed to enable simulation of large systems over long times

Figure from P.T. Cummings

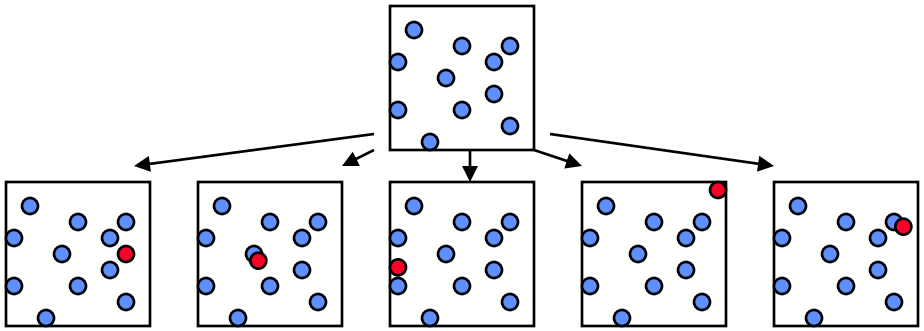
Parallelizing Monte Carlo

- Parallel moves in independent regions
  - *moves and range of interactions cannot span large distances*
- Hybrid Monte Carlo
  - *apply MC as bad MD, and apply MD parallel methods*

    time information lost while introducing limitations of MD
- Farming of independent tasks or simulations
  - *equilibration phase is sequential*
  - *often a not-too-bad approach*
- Parallel trials with coupled acceptance
  - *"Esselink" method*

Esselink Method

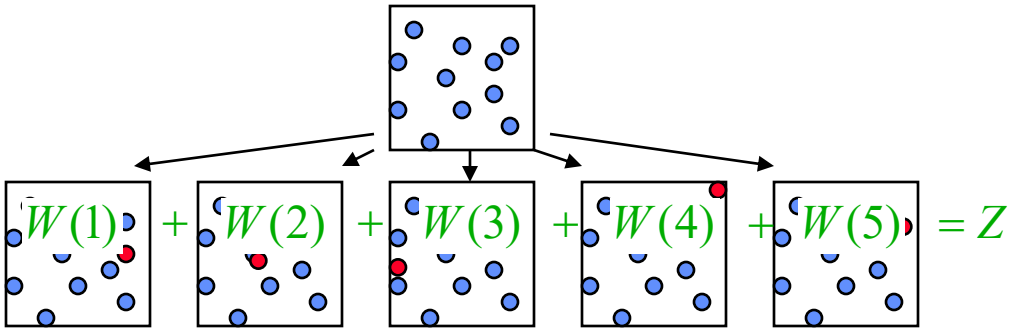
- 1. Generate k trials from the present configuration
  - *each trial handled by a different processor*
  - *useful if trials difficult to generate (e.g., chain configurational bias)*



Esselink Method

- 1. Generate k trials from the present configuration
  - *each trial handled by a different processor*
  - *useful if trials difficult to generate (e.g., chain configurational bias)*
- 2. Compute an appropriate weight W(i) for each new trial
  - *e.g., Rosenbluth weight if CCB*
  - *more simply, W(i) = exp[-U(i)/kT]*
- 3. Define a normalization factor

$$Z = \sum_{i=1}^{k} W(i)$$



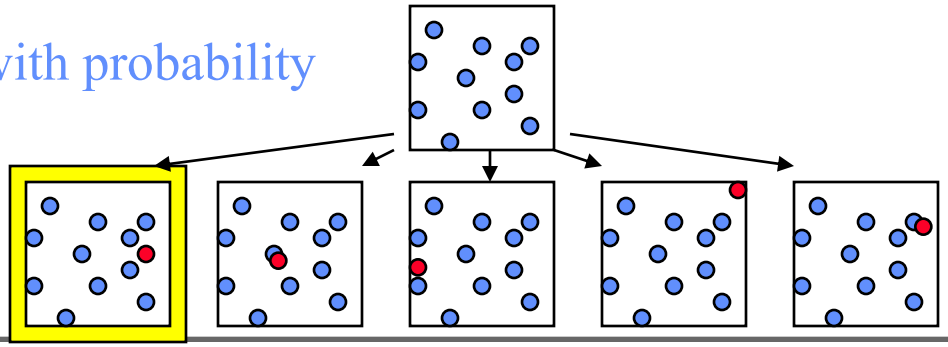
Esselink Method

- 1. Generate k trials from the present configuration
  - *each trial handled by a different processor*
  - *useful if trials difficult to generate (e.g., chain configurational bias)*
- 2. Compute an appropriate weight W(i) for each new trial
  - *e.g., Rosenbluth weight if CCB*
  - *more simply, W(i) = exp[-U(i)/kT]*
- 3. Define a normalization factor

$$Z = \sum_{i=1}^{k} W(i)$$

- 4. Select one trial with probability

$$p(n) = W(n) / Z$$



Esselink Method

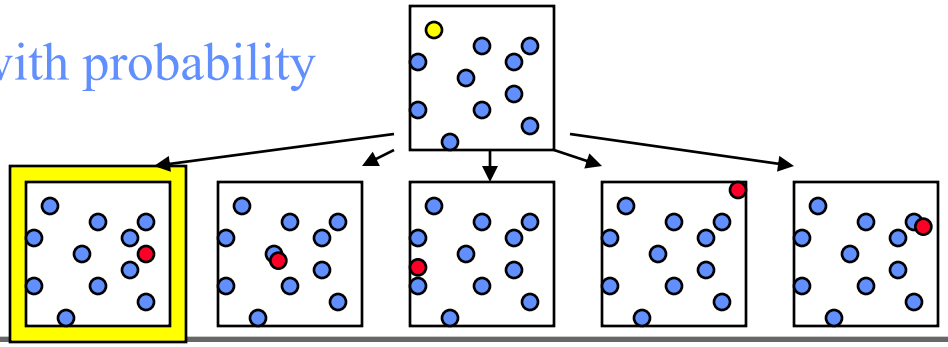
- 1. Generate k trials from the present configuration
  - *each trial handled by a different processor*
  - *useful if trials difficult to generate (e.g., chain configurational bias)*
- 2. Compute an appropriate weight W(i) for each new trial
  - *e.g., Rosenbluth weight if CCB*
  - *more simply, W(i) = exp[-U(i)/kT]*
- 3. Define a normalization factor

$$Z = \sum_{i=1}^{k} W(i)$$

- 4. Select one trial with probability

$$p(n) = W(n)/Z$$

- Now account for reverse trial:
- 5. Pick a molecule from original configuration
  - *Need to evaluate a probability it would be generated from trial configuration*
  - *Plan: Choose a path that includes the other (ignored) trials*



Esselink Method

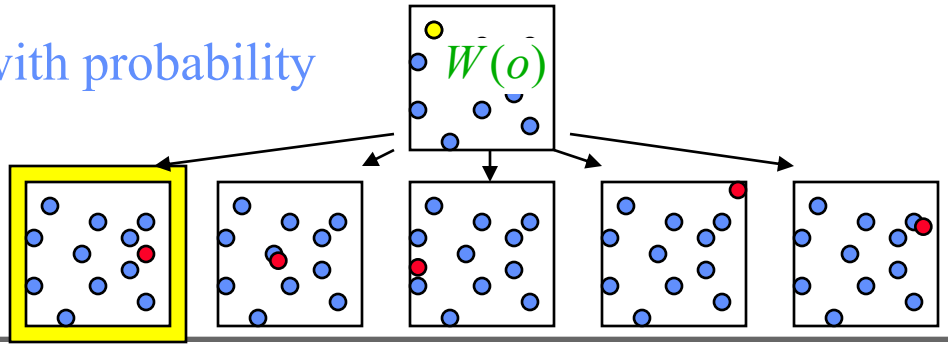
- 1. Generate k trials from the present configuration
  - *each trial handled by a different processor*
  - *useful if trials difficult to generate (e.g., chain configurational bias)*
- 2. Compute an appropriate weight W(i) for each new trial
  - *e.g., Rosenbluth weight if CCB*
  - *more simply, W(i) = exp[-U(i)/kT]*
- 3. Define a normalization factor

$$Z = \sum_{i=1}^{k} W(i)$$

- 4. Select one trial with probability

$$p(n) = W(n)/Z$$

- Now account for reverse trial:
- 5. Pick a molecule from original configuration
  - *Need to evaluate a probability it would be generated from trial configuration*
  - *Plan: Choose a path that includes the other (ignored) trials*
- 6. Compute the reverse-trial W(o)



Esselink Method

- 1. Generate k trials from the present configuration
  - *each trial handled by a different processor*
  - *useful if trials difficult to generate (e.g., chain configurational bias)*
- 2. Compute an appropriate weight W(i) for each new trial
  - *e.g., Rosenbluth weight if CCB*
  - *more simply, W(i) = exp[-U(i)/kT]*
- 3. Define a normalization factor

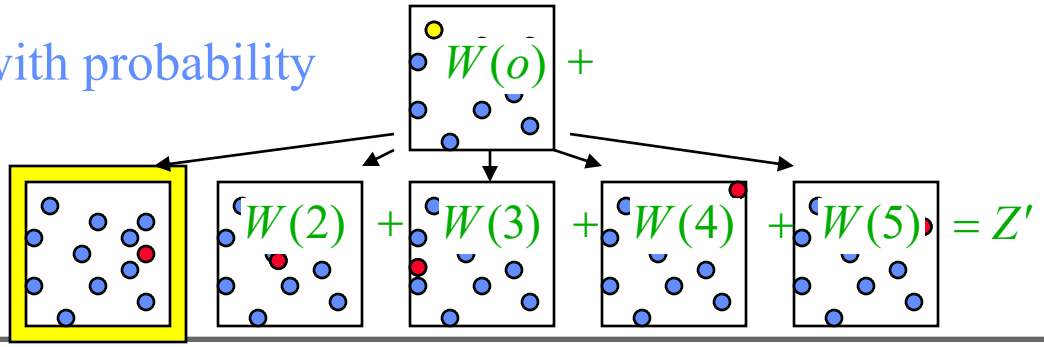
$$Z = \sum_{i=1}^{k} W(i)$$

- 4. Select one trial with probability

$$p(n) = W(n)/Z$$

- Now account for reverse trial:
- 5. Pick a molecule from original configuration
  - *Need to evaluate a probability it would be generated from trial configuration*
  - *Plan: Choose a path that includes the other (ignored) trials*
- 6. Compute the reverse-trial W(o)
- 7. Compute reverse-trial normalizer

$$Z' = Z - W(n) + W(o) \equiv R + W(o)$$



Esselink Method

- 1. Generate k trials from the present configuration
  - *each trial handled by a different processor*
  - *useful if trials difficult to generate (e.g., chain configurational bias)*
- 2. Compute an appropriate weight W(i) for each new trial
  - *e.g., Rosenbluth weight if CCB*
  - *more simply, W(i) = exp[-U(i)/kT]*
- 3. Define a normalization factor

$$Z = \sum_{i=1}^{k} W(i)$$

- 4. Select one trial with probability

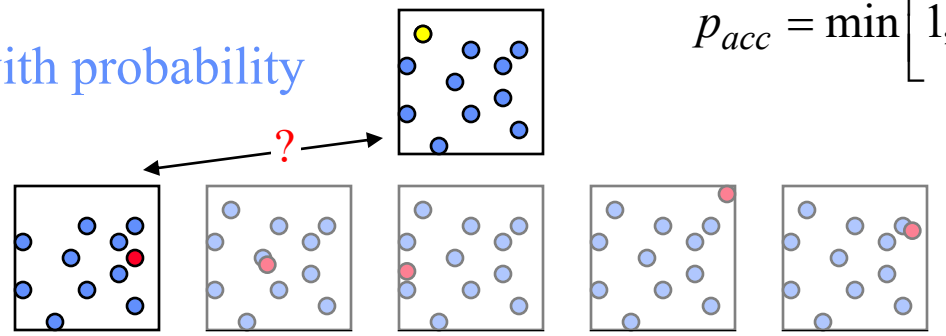
$$p(n) = W(n)/Z$$

- Now account for reverse trial:
- 5. Pick a molecule from original configuration
  - *Need to evaluate a probability it would be generated from trial configuration*
  - *Choose a path that includes the other (ignored) trials*
- 6. Compute the reverse-trial W(o)
- 7. Compute reverse-trial normalizer

$$Z' = Z - W(n) + W(o) \equiv R + W(o)$$

- 8. Accept new trial with probability

$$p_{acc} = \min\left[1, \frac{Z}{Z'}\right]$$



Some Results

- Use of an incorrect acceptance probability

$$p_{acc} = \min\left[1, \frac{W(n)}{W(o)}\right]$$

$$p_{acc} = \min\left[1, \frac{W(n)+R}{W(o)+R}\right]$$

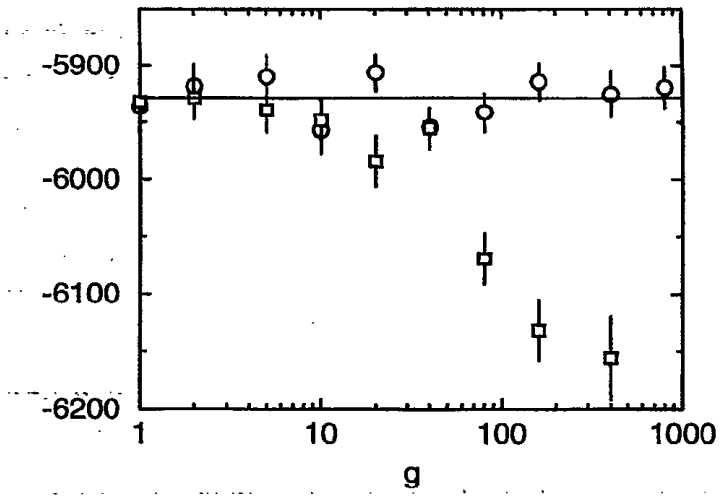
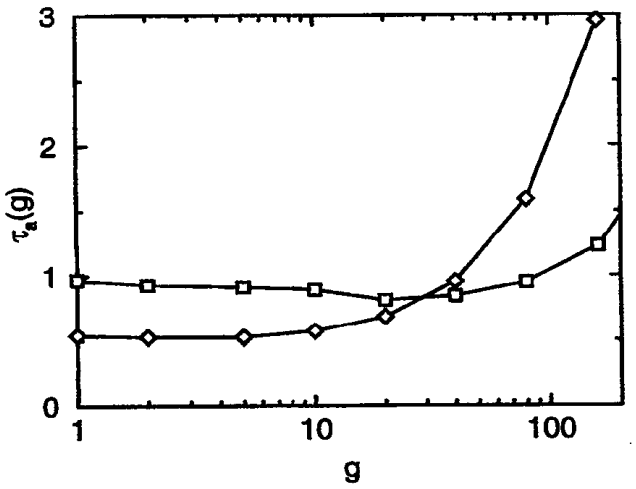


FIG. 1. Comparison of the total energy of pentane in silicalite as calculated from the sampling without the correction for the bias (□) with the correct sampling scheme (○). g is the number of chains grown in parallel and $f = 1$. The horizontal line is the average energy as calculated from the correct results.

- A sequential implemention
- Average cpu time to acceptance of a trial
  - *independent of number of trials* g *up to about* g = *10*
  - *indicates parallel implementation with* g = *10 would have 10 × speedup*
  - *larger* g *is wasteful because acceptable configurations are rejected*

only one can be accepted per move

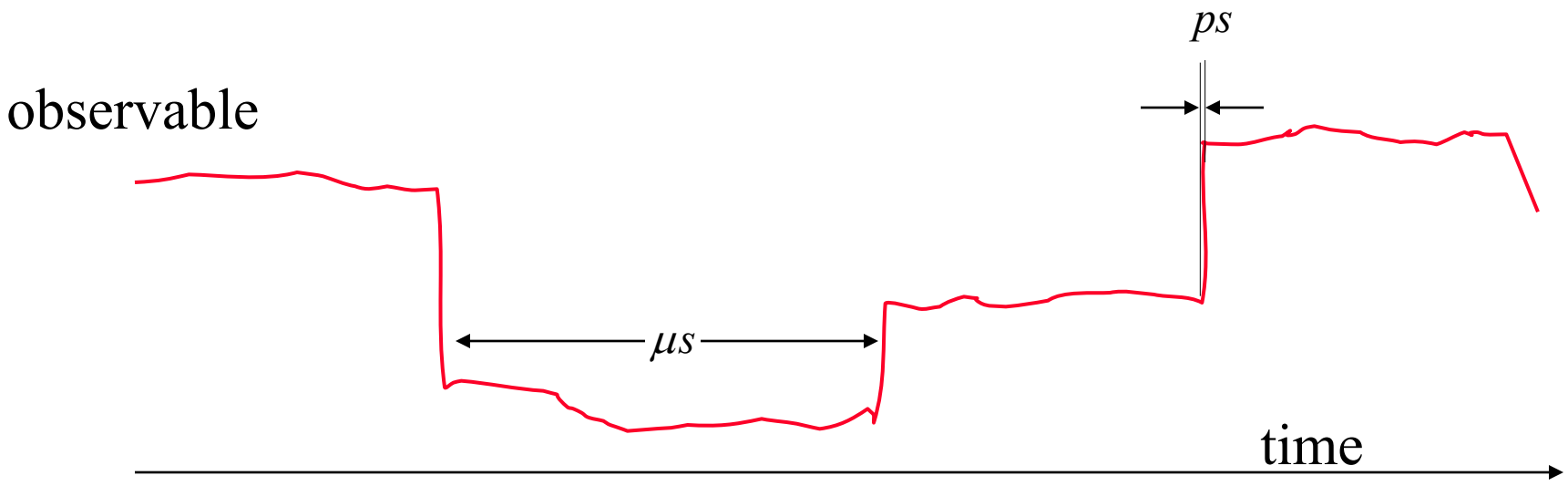


Esselink et al., PRE, 51(2) 1995

FIG. 2. Average time to acceptance $\tau_a(g)$ for methane (◇) and pentane (□) in silicalite for varying number of molecules g placed in parallel. The data are taken from Tables I and II.

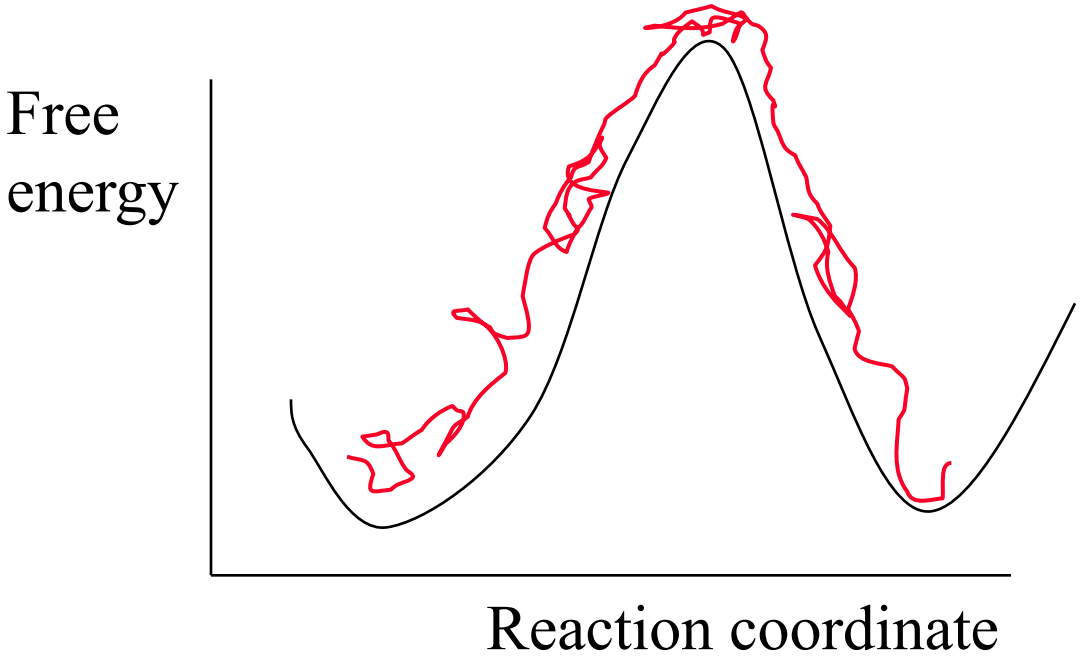
Simulation of Infrequent Events

- Some processes occur quickly but infrequently; e.g.
  - *rotational isomerization*
  - *diffusion in a solid*
  - *chemical reaction*
- Time between events may be microseconds or longer, but event transpires over picoseconds



Transition-State Theory

- Analysis of activation barrier for process



- Transition is modeled as product of two probabilities
  - *probability that reaction coordinate has maximum value*
    given by free-energy calculation (integration to top of barrier)
  - *probability that it proceeds to "product" given that it is at the maximum*
    given by linear-response theory calculation performed at top of barrier
- Requires *a priori* specification of rxn coordinate and barrier value

Parallel Replica Method (Voter's Method)

- Establish several configurations with same coordinates, but different initial momenta
- Specify criterion for departure from current "basin" in phase space
  - *e.g., location of energy minimum*

    evaluate with steepest-descent or conjugate-gradient methods
- Perform simulation dynamics in parallel for different initial systems
- Continue simulations until one of the replicas is observed to depart its local basin
- *Advance simulation clock by sum of simulation times of all replicas*
- Repeat beginning all replicas with coordinates of escaping replica

Theory Behind Voter's Method

- Assumes independent, uncorrelated crossing events
- Probability distribution for a crossing event (sequential calculation)

$$p(t) = ke^{-kt} \qquad \text{k = crossing rate constant}$$

- Probability distribution for crossing event in any of M simulations

$$p_M(t) = \sum_{j=1}^{M} \begin{pmatrix} \text{probability of crossing} \\ \text{in simulation j at time } t_j \end{pmatrix} \times \prod_{m=1}^{M} \begin{pmatrix} \text{probability simulation m} \\ \text{has yet had crossing event} \end{pmatrix}$$

$$= \sum_{j=1}^{M} p(t_j) \times \prod_{m \neq j}^{M} \bar{p}(t_m)$$

- No-crossing cumulative probability

$$\bar{p}(t) = \int_{t}^{\infty} p(\tau) d\tau = e^{-kt}$$

- Thus

$$p_M(t) = \sum_{j=1}^{M} ke^{-kt_j} \times \prod_{m \neq j}^{M} e^{-kt_m}$$

$$= Mke^{-kt_{sum}}$$

Rate constant for independent crossings is same as for individual crossings, if t_{sum} is used

Appealing Features of Voter's Method

- No *a priori* specification of transition path / reaction coordinate required
  - *Works well with multiple (unidentified) transition states*
  - *Does not require specification of "product" states*
- Parallelizes time calculation
  - *"Discarded" simulations are not wasted*
- Works well in a heterogeneous environment of computing platforms
  - *OK to have processors with different computing power*
  - *Parallelization is loosely coupled*

Summary

- Some simple efficiencies to apply to simulations
  - *Table look-up of potentials*
  - *Cell lists for identifying neighbors*
- Parallelization methods
  - *Time parallelization is difficult*
  - *Esselink method for parallelization of MC trials*
  - *Voter's method for parallelization of MD simulation of rare events*