Metropolis MC in various ensembles

Generally one follows the basic Metropolis sampling algorithm

One needs sample via random particle displacements, volume changes, as well as removal and insertion of particles.

One must use appropriate weight function and acceptance rules.

In the NVT ensemble, the natural choice for Metropolis MC, the PDF and weight functions are:

$$p_\nu^{NVT} = \frac{\exp(-\beta E_\nu)}{\sum_\nu \exp(-\beta E_\nu)} \qquad \frac{\rho_n}{\rho_m} \equiv \frac{p_n}{p_m} = \exp\left(-\frac{E_n - E_m}{k_B T}\right) = \exp\left(-\frac{E_{nm}}{k_B T}\right)$$

Isothermal – Isobaric (NPT) ensemble

One needs to allow for random particle displacements as well as volume changes. Scaled coordinates $s_i = L^{-1} r_i$ (r_i are atomic coordinates) used for volume changes.

The PDF in NPT is: $$p_\nu^{NPT} = \frac{\exp[-\beta(E_\nu + PV_\nu)]}{\sum_\nu \exp[-\beta(E_\nu + PV_\nu)]}$$

Markov chains are generated with a limiting distribution proportional to:

$$\exp[-\beta(PV + \boldsymbol{V}(s)) + N \ln V]$$

New states obtained by random particle displacements and/or volume changes:

$$s_i^n = s_i^m + \delta s_{max}(2\xi - 1) \qquad V_n = V_m + \delta V_{max}(2\xi - 1)$$

In the new state n, a quantity closely related to enthalpy is calculated:

$$\delta H_{nm} = \delta V_{nm} + P(V_n - V_m) - N\beta^{-1} \ln(V_n / V_m)$$

and move accepted with probability equal to min[1, $\exp(-\beta \delta H_{nm})$].

Algorithm 8: MC in constant (NPT) ensemble

```
program mc_npt                               basic Metropolis NPT simulation

do icycl = 1, ncycl                          perform ncycl MC cycles
    ran = ranf()*(npart +1) + 1

    if (ran .le. npart) then
        call mcmove                          attempt particle displacement
    else
        call mcvol                           attempt volume change
    endif

    if (mod (icycl, nsamp) .eq. 0)
        call sample                          sample averages

enddo
end
```

Obs: Each cycle, one performs on *average* **npart** attempts to displace particles and **one** attempt to change the volume.

Algorithm 9: Attempt to change volume

```
subroutine mcvol                                          attempt to change volume

call toterg(box, eno)                                     total energy old configuration
   vo = box**3                                            determine old volume
   lnvn = log(vo) + (ranf() – 0.5)*vmax                   perform random walk in lnV
   vn = exp(lnvn)
   boxn = vn**(1/3)                                       new box length
do i = 1, npart
       x(i) = x(i)*boxn/box                               rescale centre of mass
enddo
call toterg(boxn, enn)                                    total energy new configuration
arg = -beta*((enn – eno) + p*(vn – vo) – (npart + 1)*log(vn/vo)/beta)
                                                          appropriate weight function!

if (ranf() .gt. exp(arg)) then                            check acceptance rule
   do i = 1, npart                                        for REJECTED moves
       x(i) = x(i)*box/boxn                               restore old positions
   enddo
endif

return
end
```

Grand – canonical (μVT) ensemble

In this case one needs to allow for random addition/removal of particles from the system in addition to random particle displacements. Scaled coordinates, defined as for NPT can be used, and an activity term:

$$z = \exp(\beta\mu)\big/\Lambda^3; \qquad \Lambda = \left(h^2\big/2\pi m k_B T\right)^{1/2} \text{ – thermal de Broglie wavelength}$$

Markov chains are generated with a limiting distribution proportional to:

$$\exp[-\beta(\boldsymbol{V}(s) - N\mu) - \ln N! - 3N\ln\Lambda + N\ln V]$$

Random particle displacements yield states accepted with the same probability as in the NVT ensemble: $\min[1,\exp(-\beta\delta E_{nm})]$.

The insertion, respectively removal of a particle, yields states according to:

$$N \to N+1 = \min\left[1, \frac{V}{\Lambda^3(N+1)}\exp\{\beta[\mu - E(N+1) + E(N)]\}\right]$$

$$N \to N-1 = \min\left[1, \frac{\Lambda^3 N}{V}\exp\{-\beta[\mu + E(N-1) - E(N)]\}\right]$$

Algorithm 10: MC in constant (μVT) ensemble

```
program mc_gc                                   basic Metropolis  μVT simulation

do icycl = 1, ncycl                             perform ncycl MC cycles
   ran = int(ranf()*(npart + nexc)) + 1

   if (ran .le. npart) then
      call mcmove                               perform particle displacement
   else
      call mcexc                                exchange a particle with reservoir
   endif

   if (mod(icycl, nsamp) .eq. 0)
      call sample                               sample averages
enddo

return
end
```

Obs: Each cycle, one performs on *average* **npart** attempts to displace particles and **nexc** attempts to exchange particles with the reservoir.

Algorithm 11: Attempt to exchange particle with reservoir

```
subroutine mcexc                              attempt to exchange particles with reservoir

if (ranf() .lt. 0.5) then                     decide to remove or add a particle
  if (npart. eq. 0) return                    test whether there is a particle
    o = int(npart*ranf()) + 1                   select a particle to be removed
    call ener(x(o), eno)                        energy particle o
    arg = npart*exp(beta*eno) / (zz*vol)        acceptance rule
    if (ranf() .lt. arg) then                   check acceptance rule
      x(o) = x(npart)                           if accepted, remove particle o
      npart = npart – 1
    endif
else
  xn = ranf()*box                             new particle at a random position
  call ener(xn, enn)                          energy new particle
  arg = zz*vol*exp(-beta*enn) / (npart+1)     acceptance rule
  if (ranf() .lt. arg) then                   check acceptance rule
    x(npart+1) = xn                           if accepted, add new particle
    npart = npart +1
  endif

return
end
```

MC simulations in the Gibbs ensemble

Gibbs ensemble – originally introduced as a combination of NVT, NPT and μVT ensembles

Well suited for simulations of "coexistence without interfaces".
- – eg. First order phase transitions, phase equilibria in general.
- – standard technique for studies in vapour-liquid and liquid-liquid equilibria.

Can be implemented as either NVT or NPT ensembles
- – NVT used in one-component simulations
- – NPT used in simulations of systems with two or more components.

Focus here is on the NVT "Gibbs ensemble".

Definition: the ensemble in which two systems can exchange both volume and particles in such a way that the total volume V and total number of particles N are fixed.

MC simulations in the Gibbs ensemble

MC schemes for this ensemble must sample all possible configurations of two systems that can exchange particles and volume.

One needs to consider the following trial moves:

– Displacement of a randomly selected particle.
– Change of the volume such that total volume remains constant.
– Transfer of a randomly selected particle from one box to the other.

Particle displacement: $\rho_n / \rho_0 \approx \min \left\{ 1,\ \exp\ \{-\beta\{-\beta({}^{n_1}_{n})\text{- } U(s_0^{n_1})]\} \right\}$

Volume change: $\rho_n / \rho_0 \approx \min \left\{ 1, \left(\frac{V_1^n}{V_1^0} \right)^{n_1+1} \left(\frac{V - V_1^n}{V - V_1^0} \right)^{N-n_1+1} \exp\{-\beta[U(s_n^N) - U(s_0^N)]\}\} \right\}$

Particle exchange: $\rho_n / \rho_0 \approx \min \left\{ 1, \frac{n_1\,(V\text{-}V1)}{(N\text{-}\ n_1 + 1)\ V_1} \exp\{-\beta[U(s_n^N) - U(s_0^N)]\} \right\}$

Algorithm 12: MC in the Gibbs ensemble

```
program mc_Gibbs                                  Gibbs ensemble simulation

do icycl = 1, ncycl                               perform ncycl MC cycles
  ran = ranf()*(npart + nvol + nswap)             decide what to do
     if (ran .le. npart) then
           call mcmove                            attempt to displace particle
     else if (ran .le. (npart + nvol))
           call mcvol                             attempt to change the volume
     else
           call mcswap                            attempt to swap a particle
     endif
  call sample                                     sample averages
enddo

return
end
```

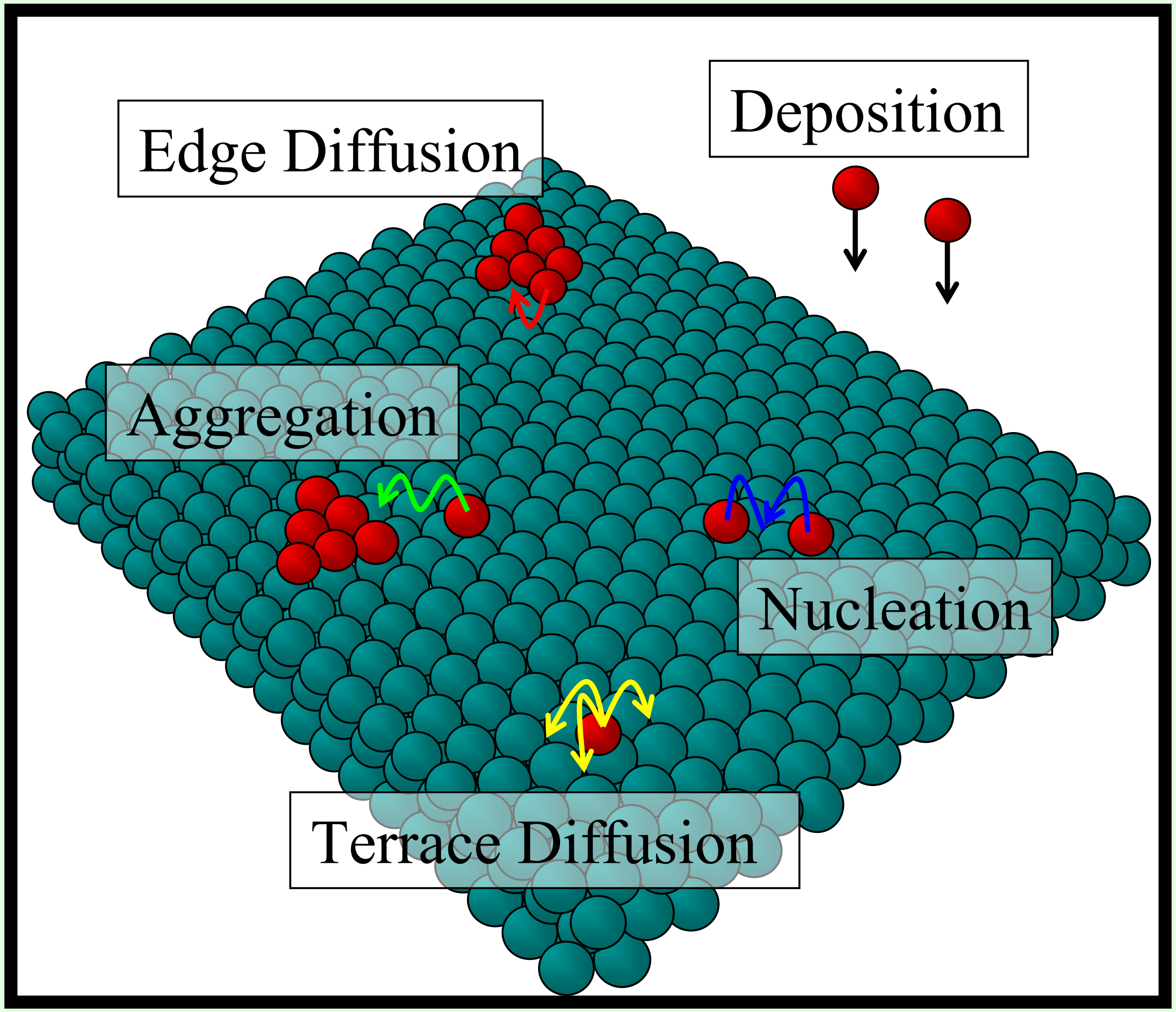
Algorithm 13: Attempt to change volume in Gibbs ensemble

```
      subroutine mcvol                                      attempt to change volume
call toterg(box1, en1o)                                          energy old conf. box 1
call toterg(box2, en2o)                                          and 2 (box1: box length)
vo1 = box1**3                                                    old volume box 1
vo2 = v – vo1                                                    and box 2
   lnvn = log(vo1/vo2) + (ranf() – 0.5)*vmax)                  random walk in ln(V_1/V_2)
   v1n = v*exp(lnvn) / (1 + exp(lnvn))                         new volume box 1
   v2n = v – v1n                                               and box 2
   box1n = v1n**(1/3)                                            new box length box 1
   box2n = v2n**(1/3)                                            new box length box 2
      do i = 1, npart
         if (ibox(i) .eq. 1) then                           determine which box
                fact = box1n/box1o
         else
                fact = bo2n/box2o
         endif
         x(i) = x(i)*fact                                   rescale positions
      enddo
call toterg(box1n, en1n)                                         total energy new box 1
call toterg(box2n, en2n)                                         total energy new box 2

arg1 = -beta*((en1n - en1o) + (npbox(1) + 1)*log(v1n/v1o) /beta)     appropriate weight function
arg2 = -beta*((en2n - en2o) + (npbox(2) + 1)*log(v2n/v2o) /beta)     appropriate weight function

if (ranf() .gt. exp(arg1 + arg2))  then                     check acceptance rule
      do i = 1, npart                                       for REJECTED moves
         if (ibox(i) .eq. 1)  then                          determine which box
                fact = box1o/box1n
         else
                fact = box2o/box2n
         endif
         x(i) = x(i)*fact                                   restore positions
         enddo
   endif

return
end
```

Algorithm 14: Attempt to swap a particle between two boxes

```
    subroutine mswap                                    attempts to swap a particle between two boxes
if (ranf() .lt. 0.5) then                               which box to add or remove
  in = 1
  out = 2
else
  in = 2
  out = 1
endif

xn = ranf ()*box(in)                                    new particle at random position
call ener(xn, enn, in)                                  energy new particle in box in

w(in) = w(in) + vol(in)*exp(-beta*enn) / (npbox(in) +1)   update chemical potential ***

if (npbox(out) .eq. 0) return                           if box empty return
  ido = 0                                               find a particle to be removed
  do while (ido. ne. out)
      o = int(npart*ranf()) +1
      ido = ibox(o)
  enddo
  call ener(x(o), eno, out)                             energy particle o in box out

  arg = exp(-beta*(enn – eno + log (vol(out)*(npbox(in) +1) / (vol(in)*npbox(out))) / beta))
                                                        appropriate weight function
  if (ranf() .lt. arg)  then                            check acceptance rule
      x(o) = xn                                         add new particle to box in
      ibox(o) = in
      npbox(out) = npbox(out) – 1
      npbox(in) = npbox(in) + 1
  endif

return
end
```

Kinetic Monte Carlo



Consider Diffusion on a triangular lattice

$$D = \Theta \cdot D_J$$

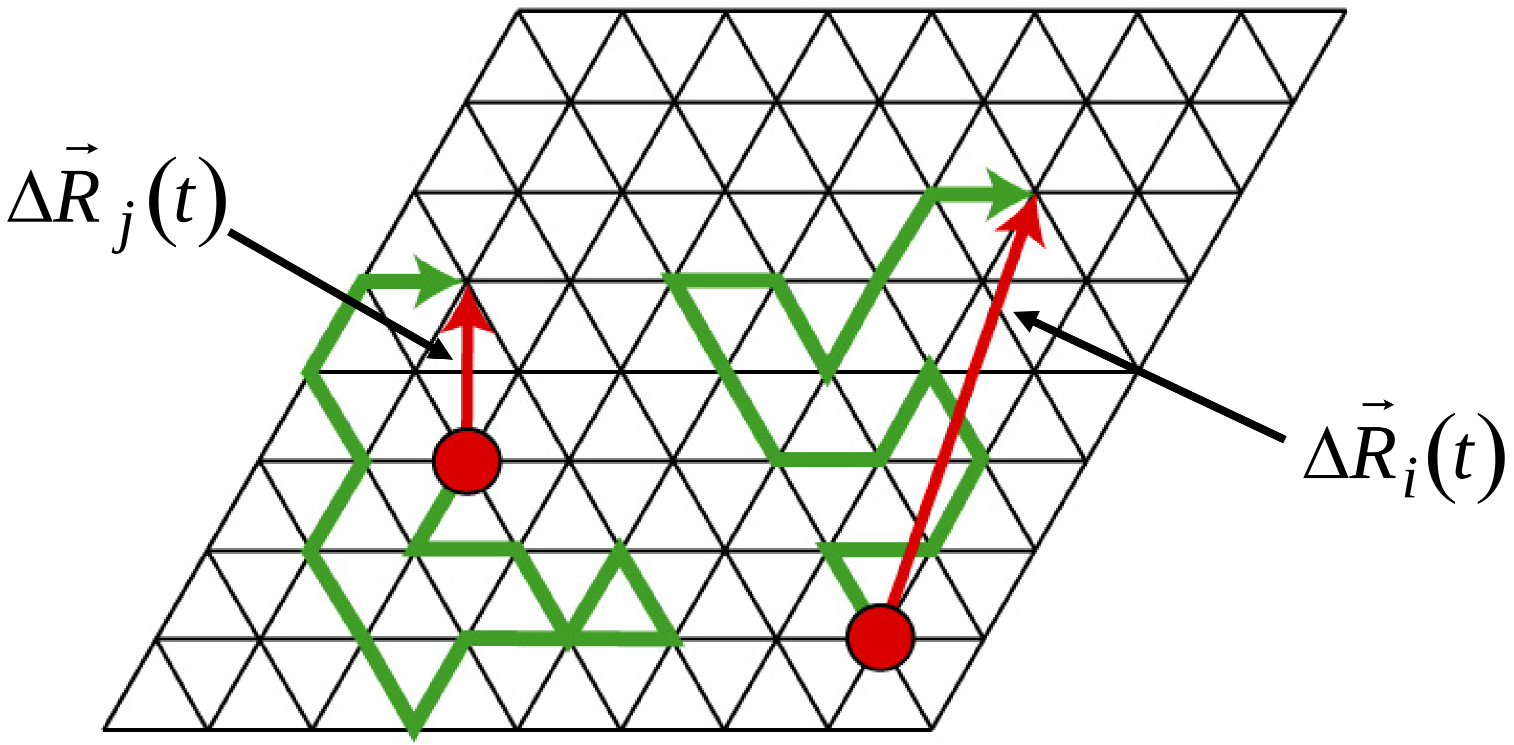
Thermodynamic factor

$$\Theta = \frac{\partial\left(\frac{\mu}{k_B T}\right)}{\partial \ln x} = \frac{\langle N \rangle}{\langle N^2 \rangle - \langle N \rangle^2}$$

Self Diffusion Coefficient

$$D_J = \frac{1}{(2d)t}\left\langle \frac{1}{N}\left(\sum_{i=1}^{N} \Delta\vec{R}_i(t)\right)^2 \right\rangle$$

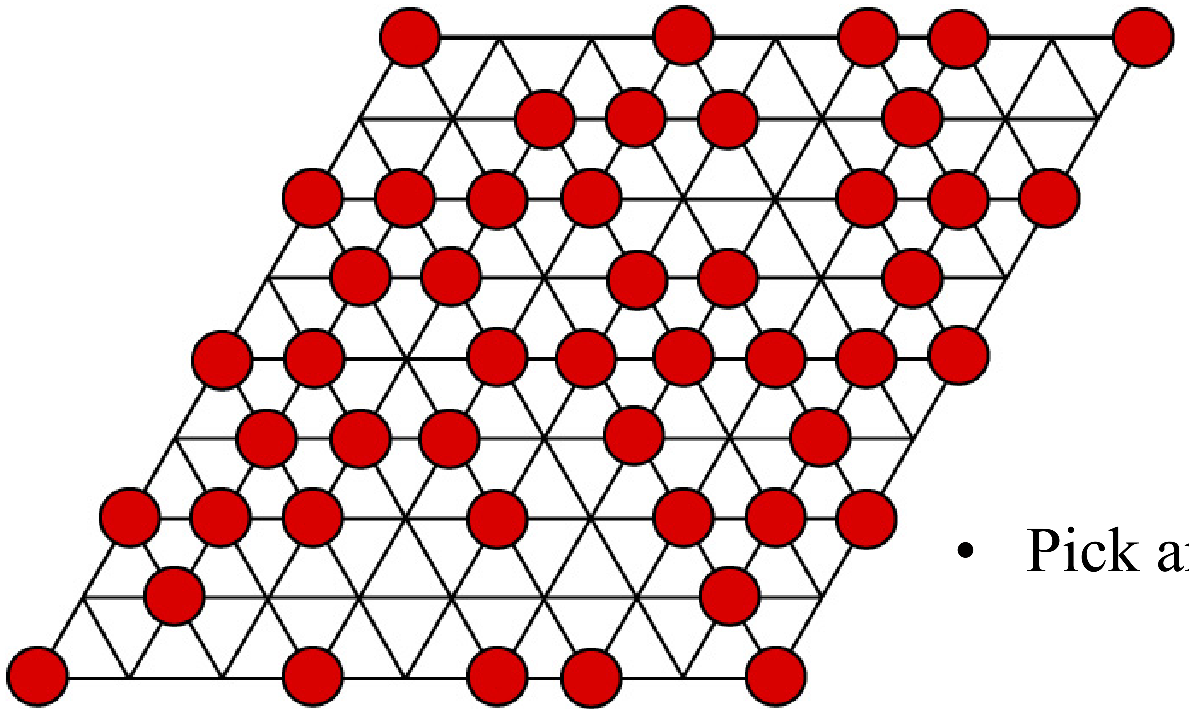
Diffusion



$$D_J = \frac{1}{(2d)t}\left\langle \frac{1}{N}\left(\sum_{i=1}^{N}\Delta\vec{R}_i(t)\right)^2 \right\rangle$$

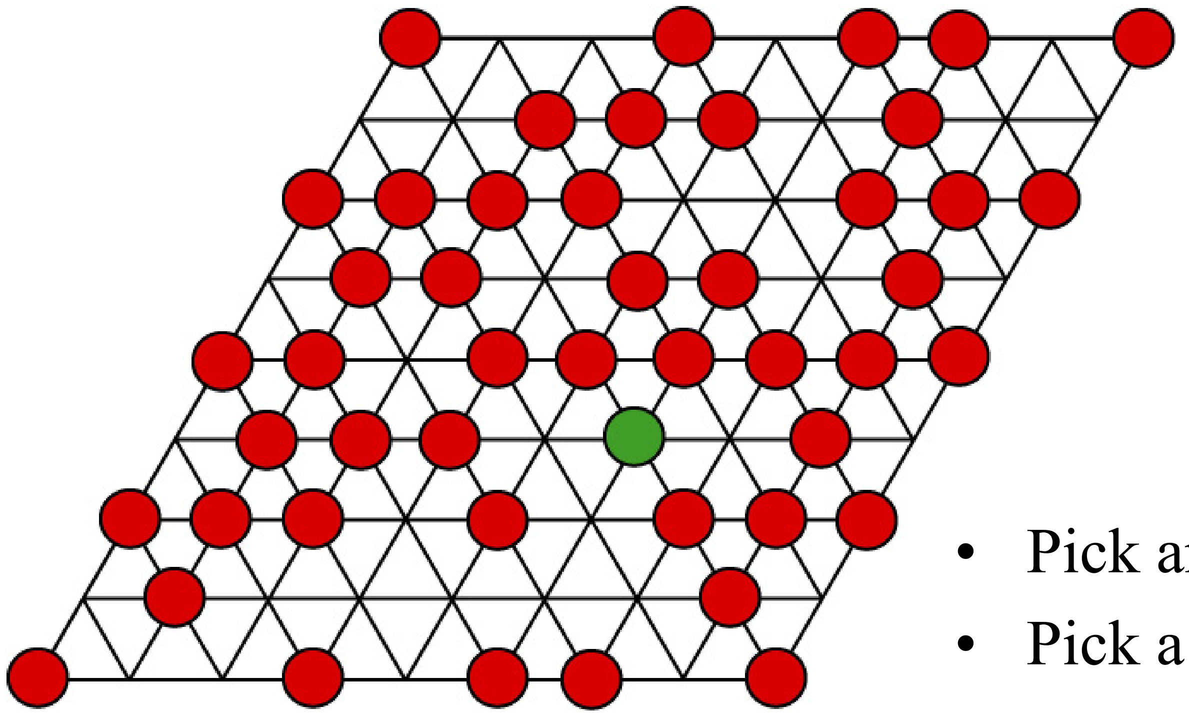
$$D^* = \frac{1}{(2d)t}\left\langle \frac{1}{N}\sum_{i=1}^{N}\Delta\vec{R}_i(t)^2 \right\rangle$$

Standard Monte Carlo to study diffusion



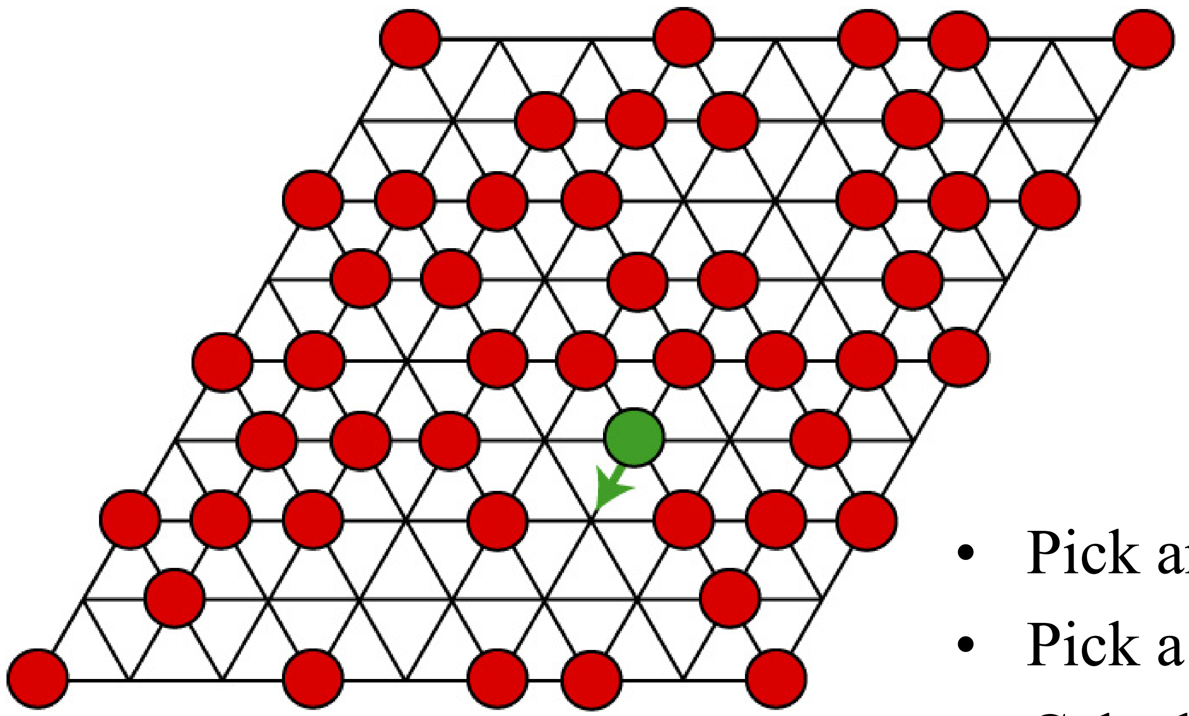
- Pick an atom at random

Standard Monte Carlo to study diffusion



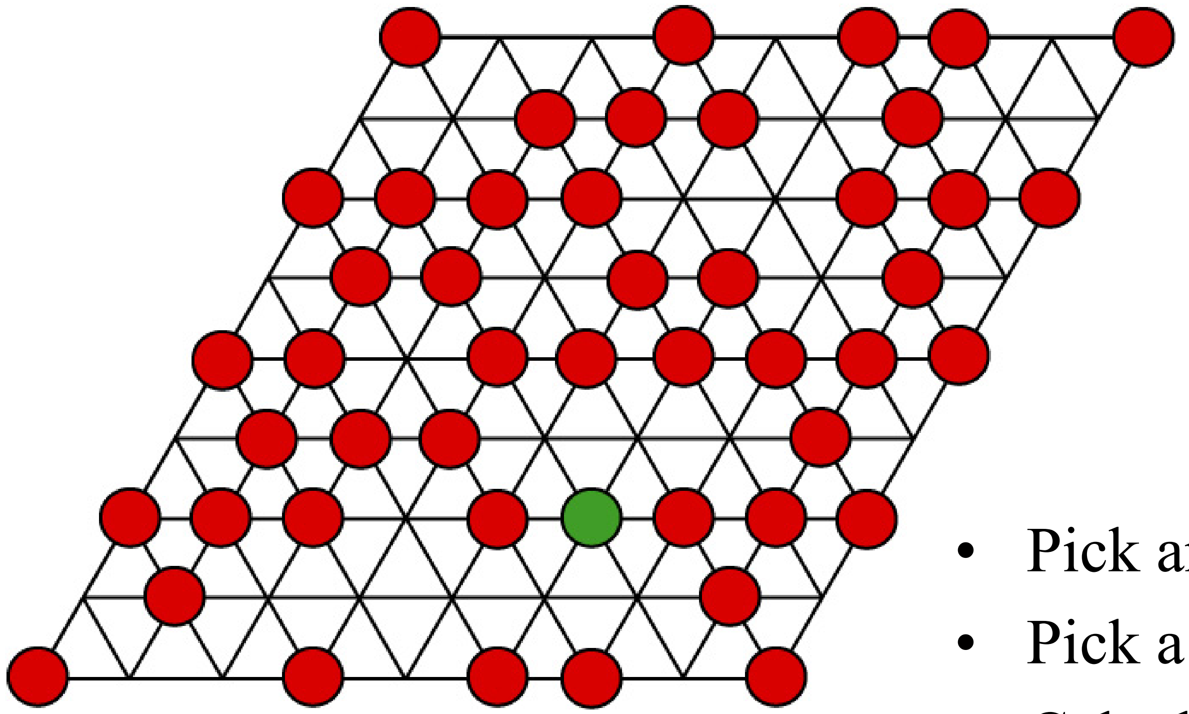
- Pick an atom at random
- Pick a hop direction

Standard Monte Carlo to study diffusion



- Pick an atom at random
- Pick a hop direction
- Calculate $\exp\left(-\Delta E_b / k_B T\right)$

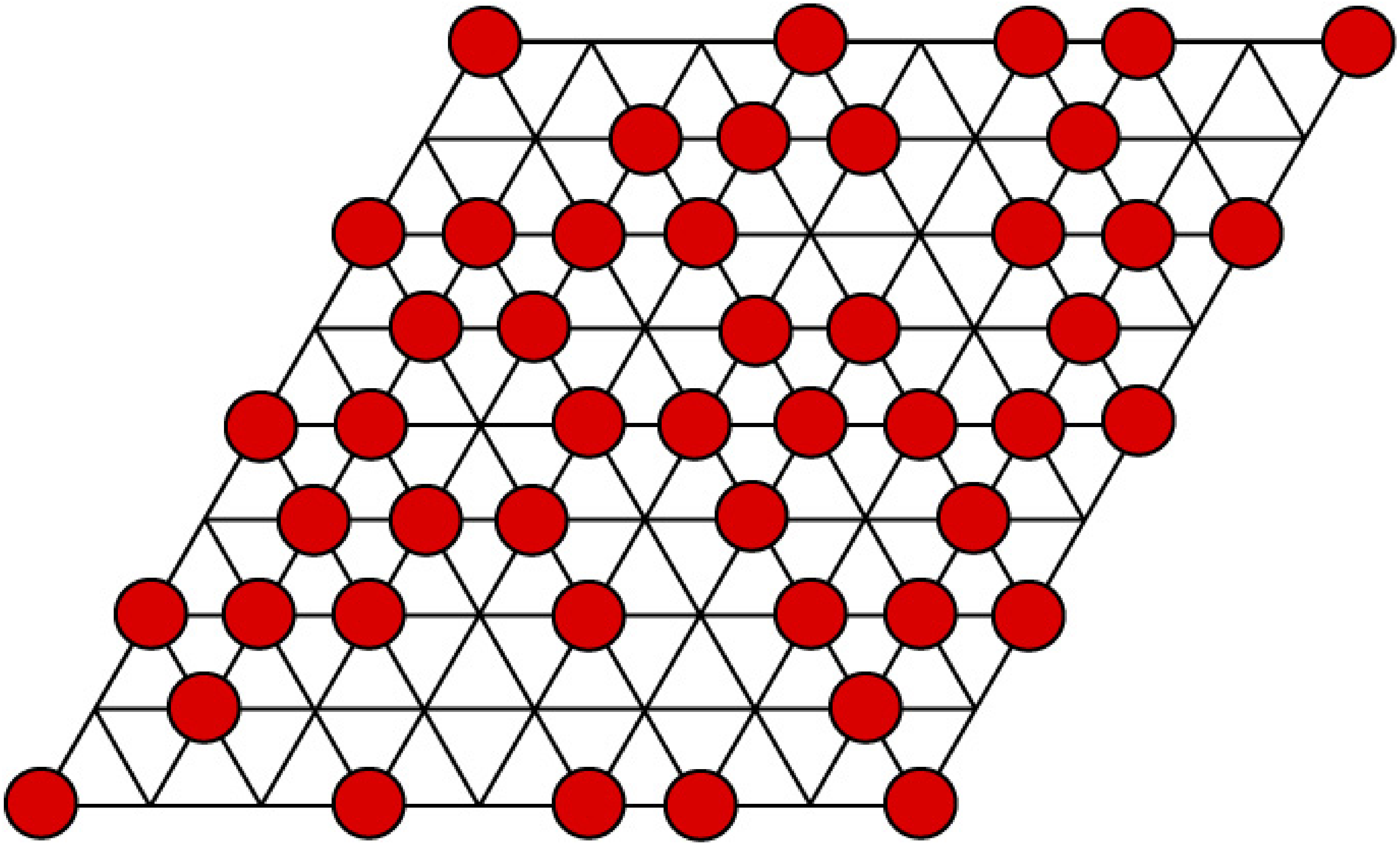
Standard Monte Carlo to study diffusion

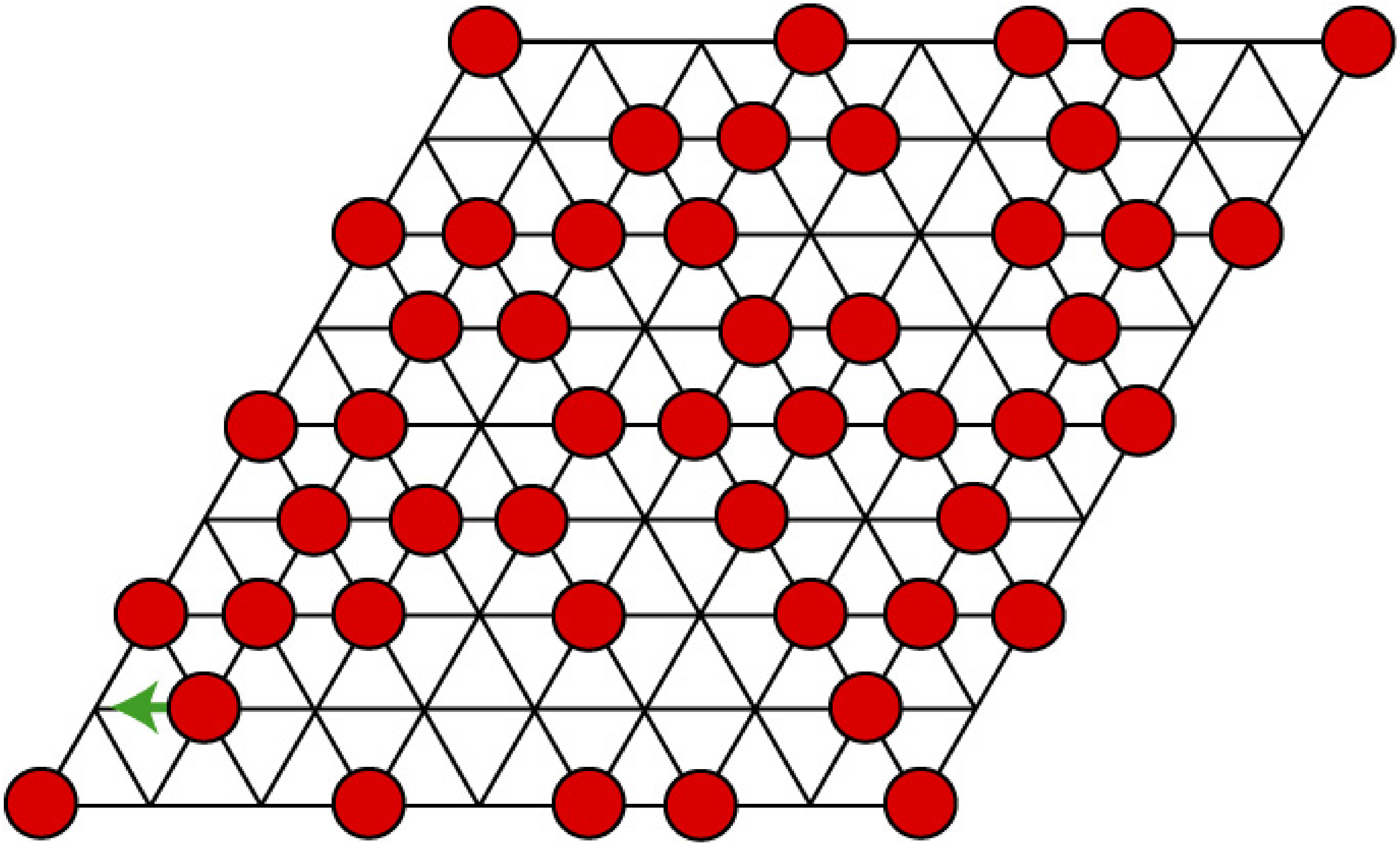


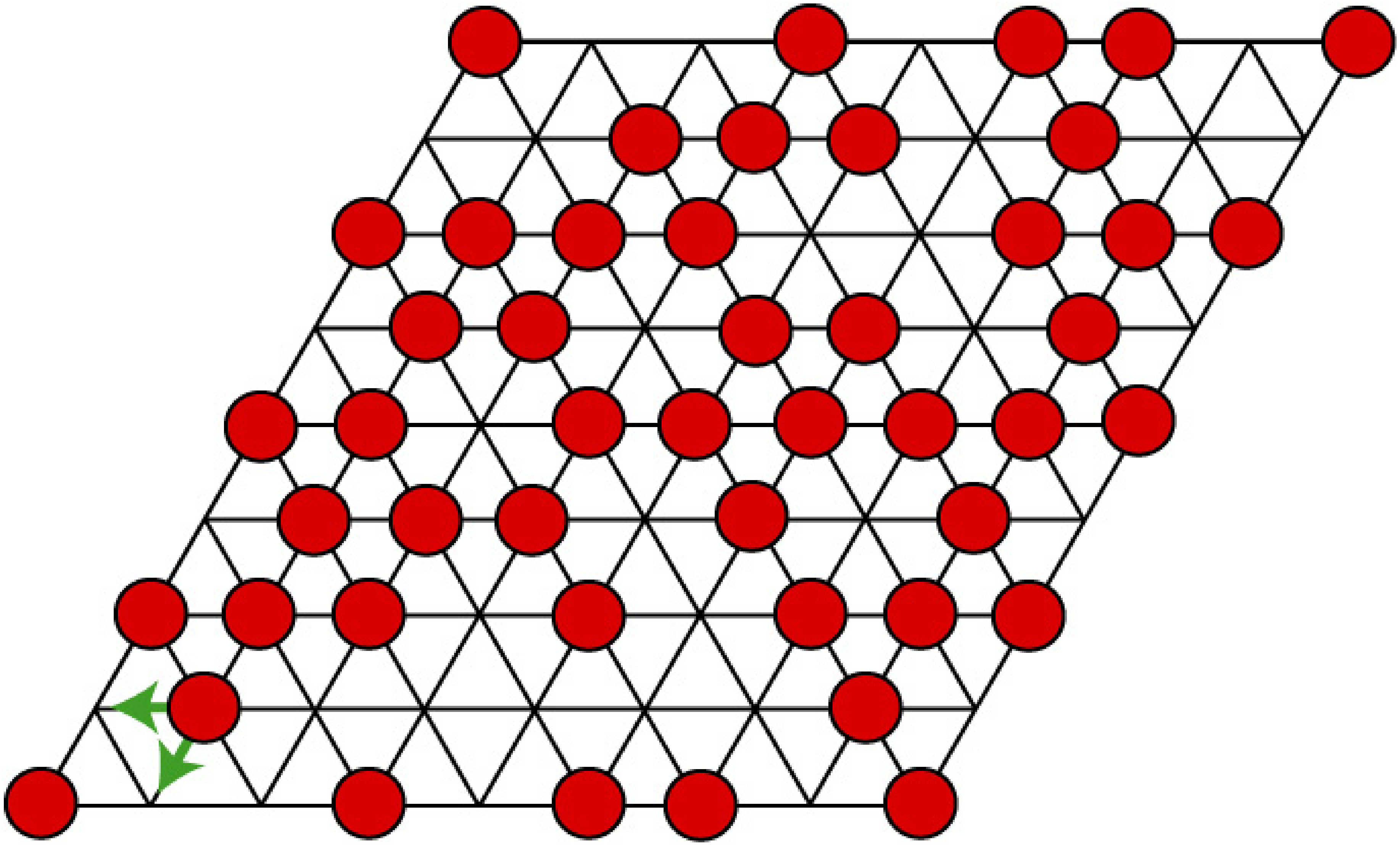
- Pick an atom at random
- Pick a hop direction
- Calculate $\exp\left(-\Delta E_b / k_B T\right)$
- If ($\exp\left(-\Delta E_b / k_B T\right) >$ random number) do the hop

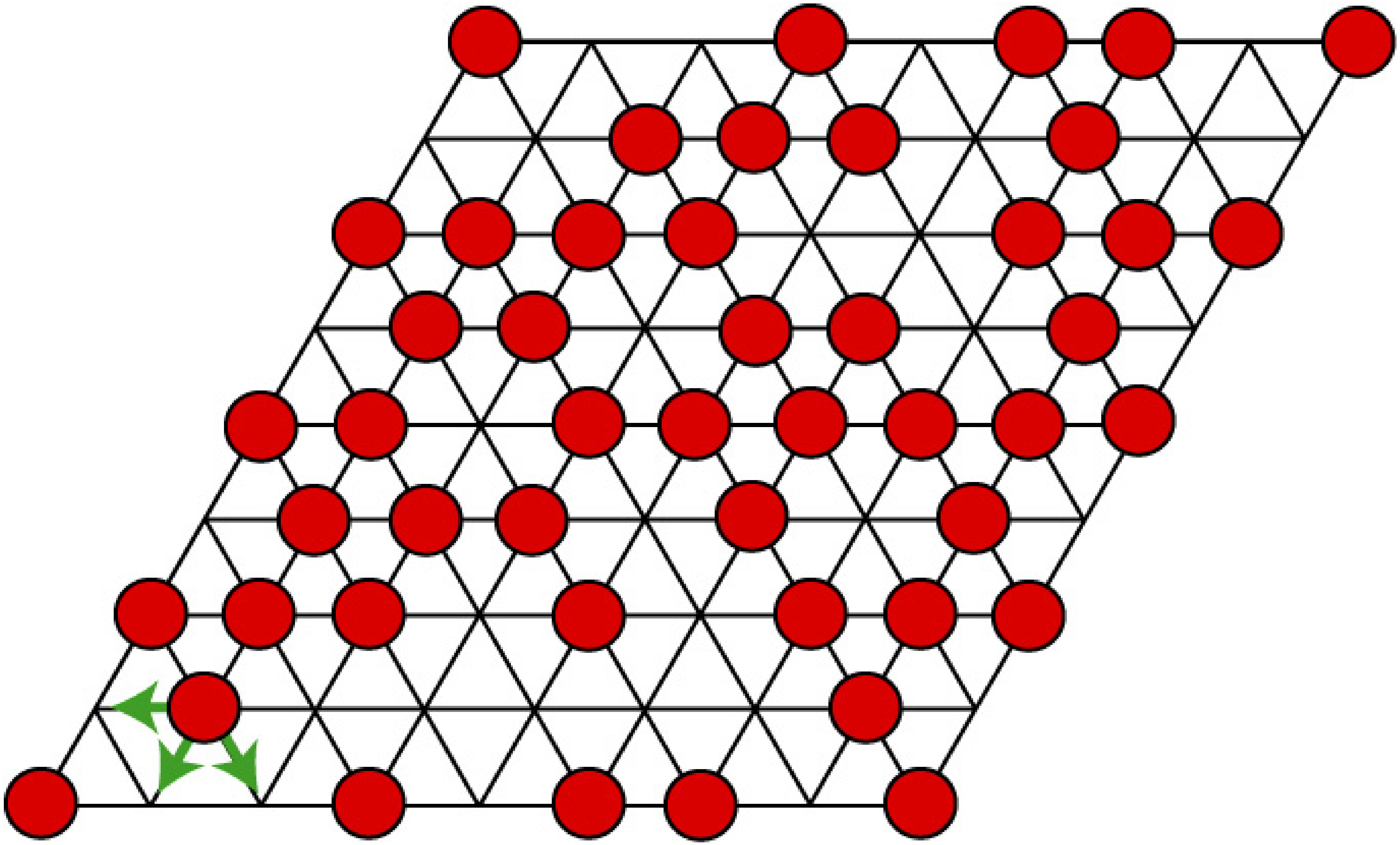
Kinetic Monte Carlo

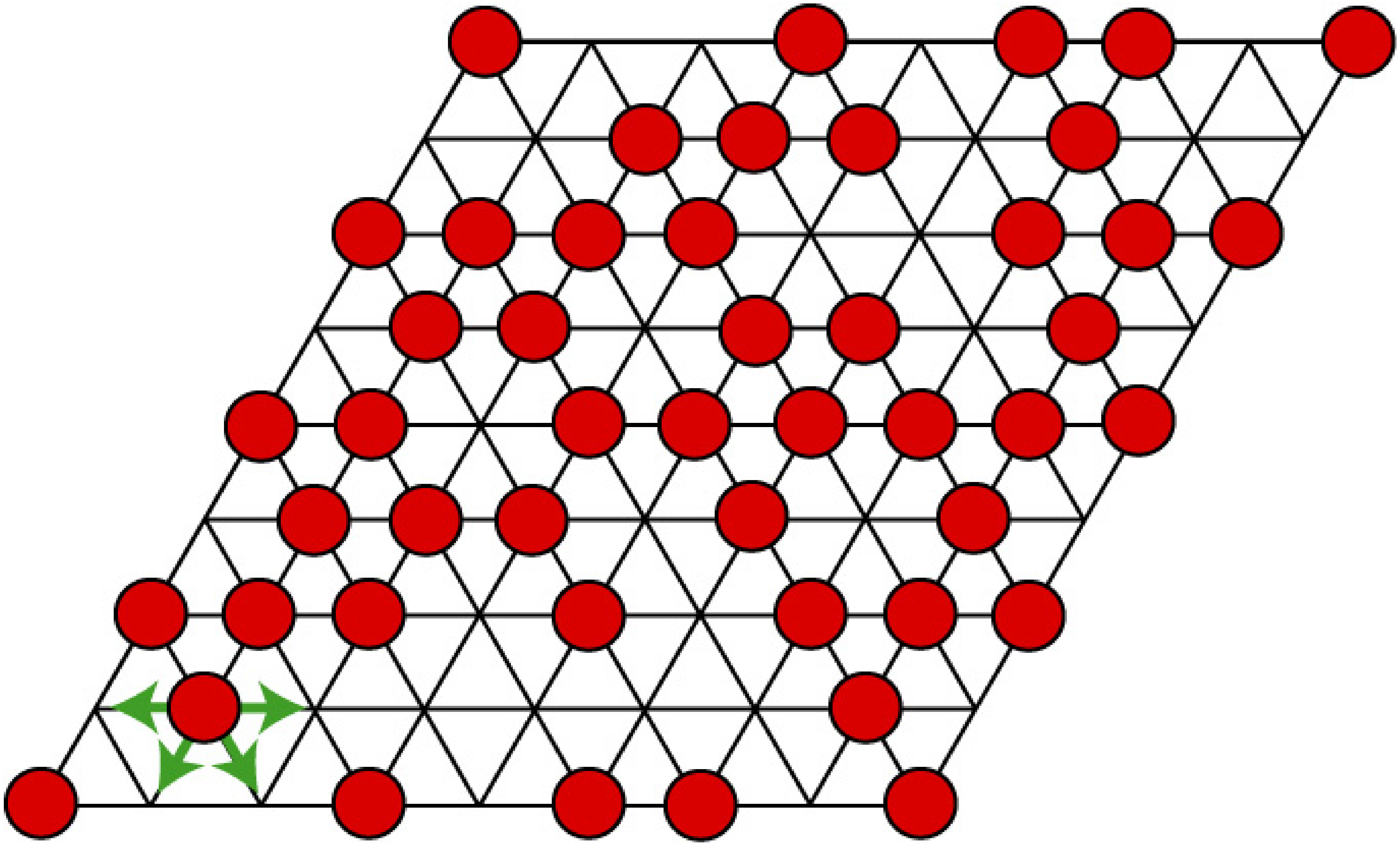
Consider all hops simultaneously

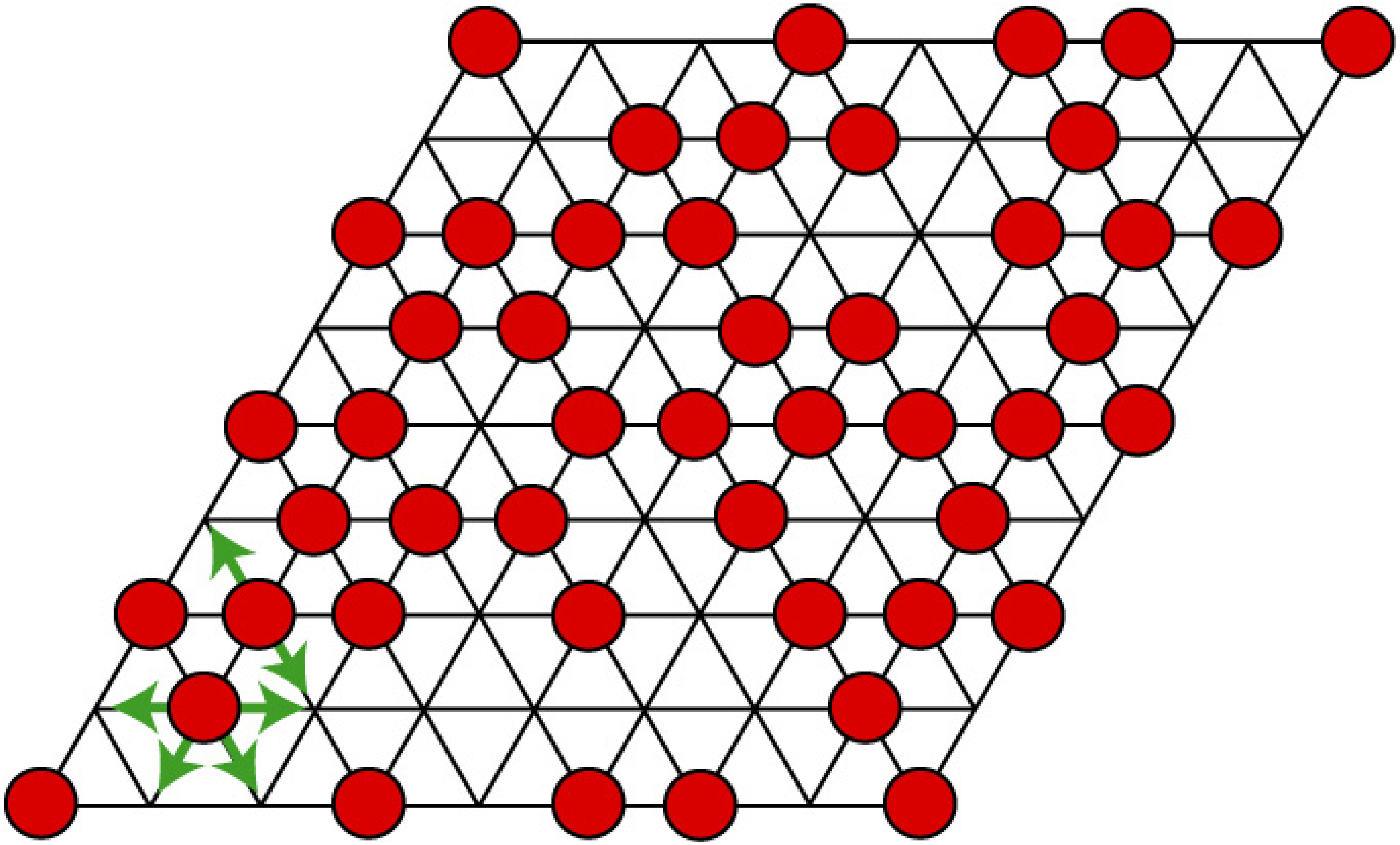


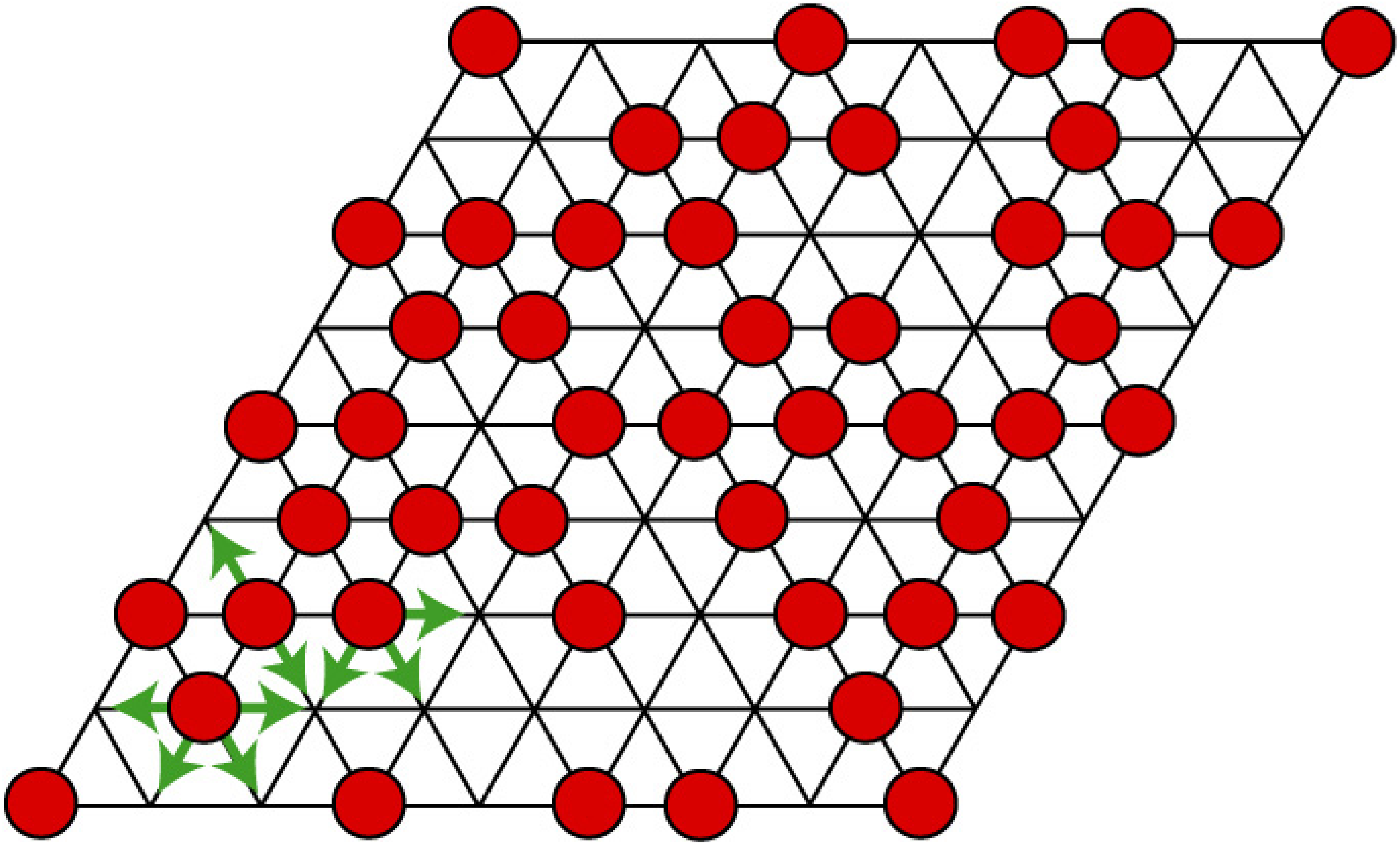


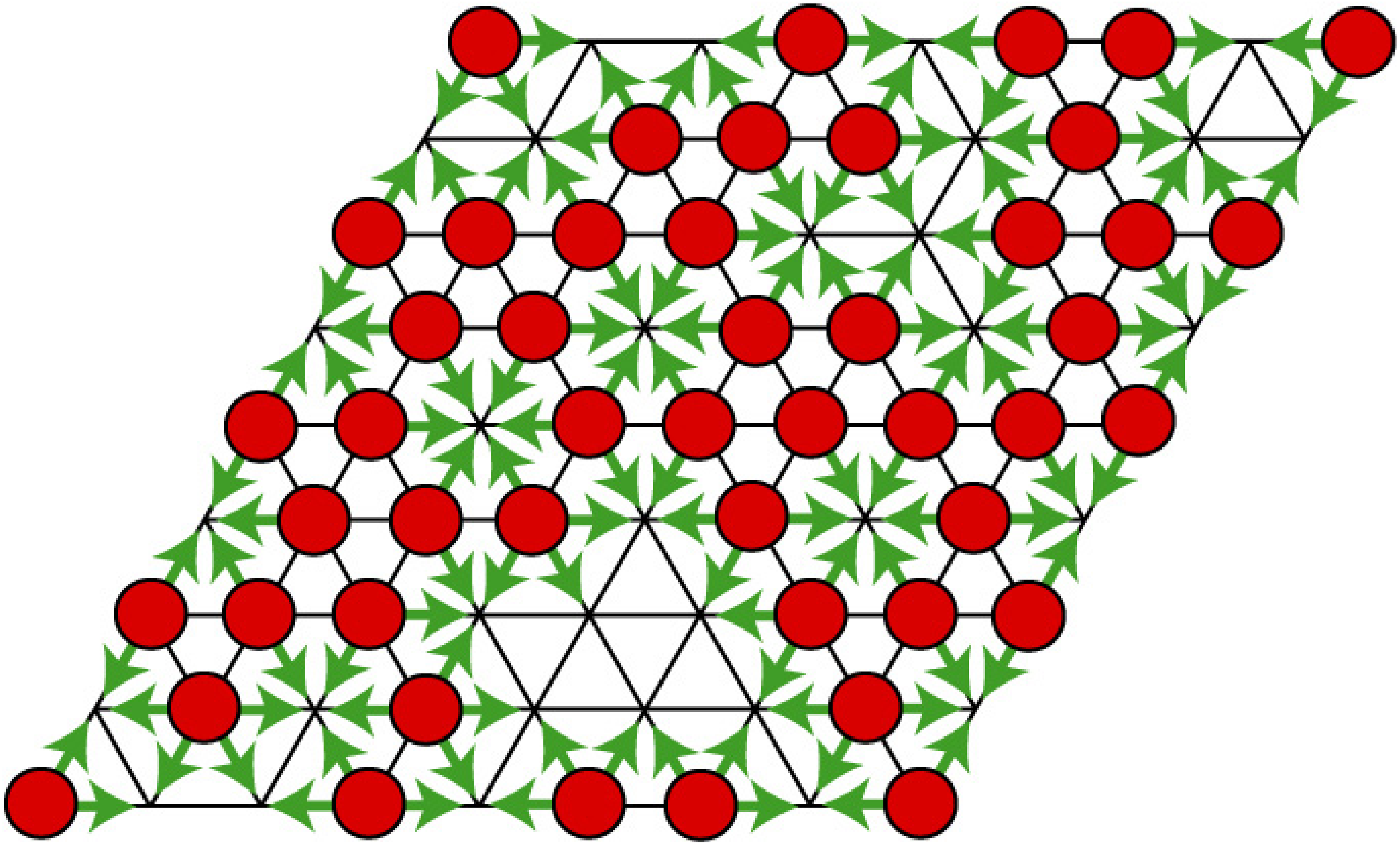


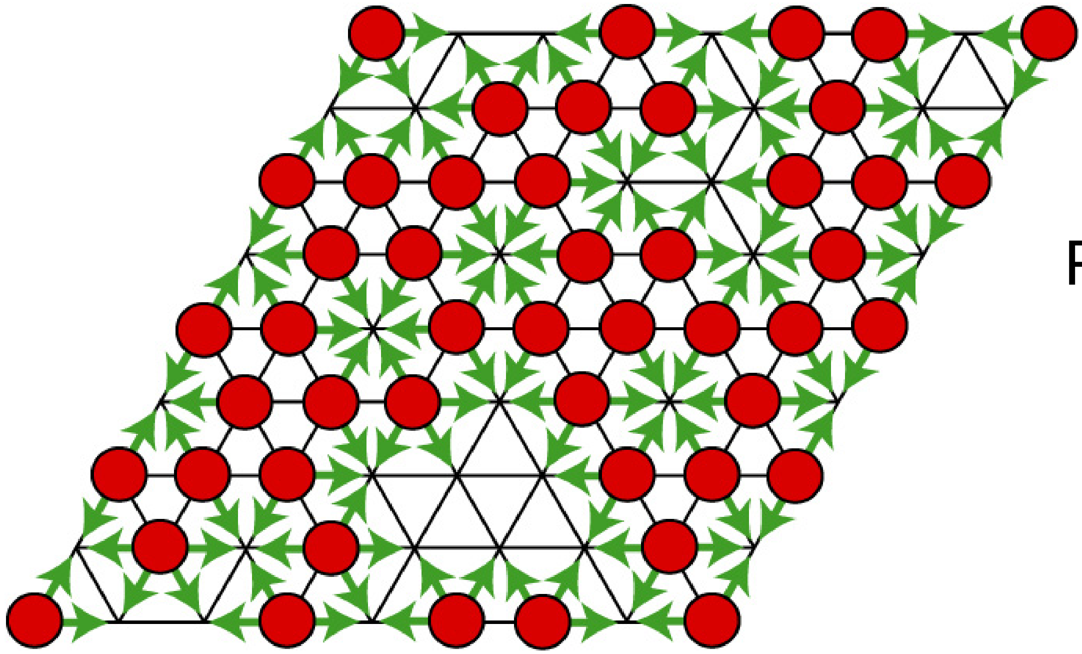






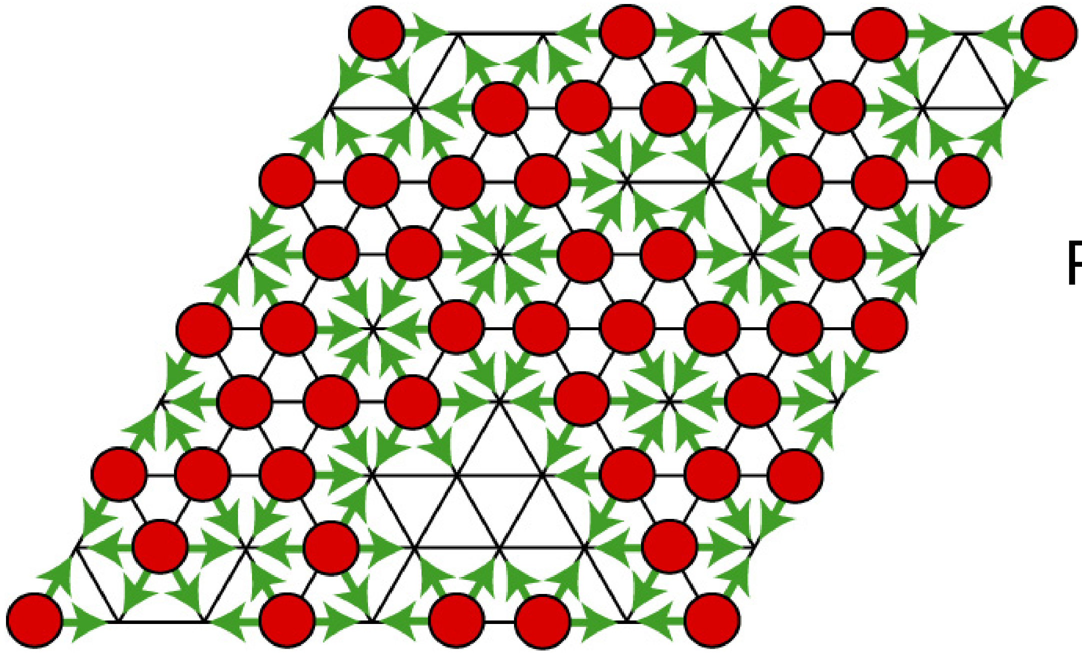






For each potential hop *i*, calculate the hop rate

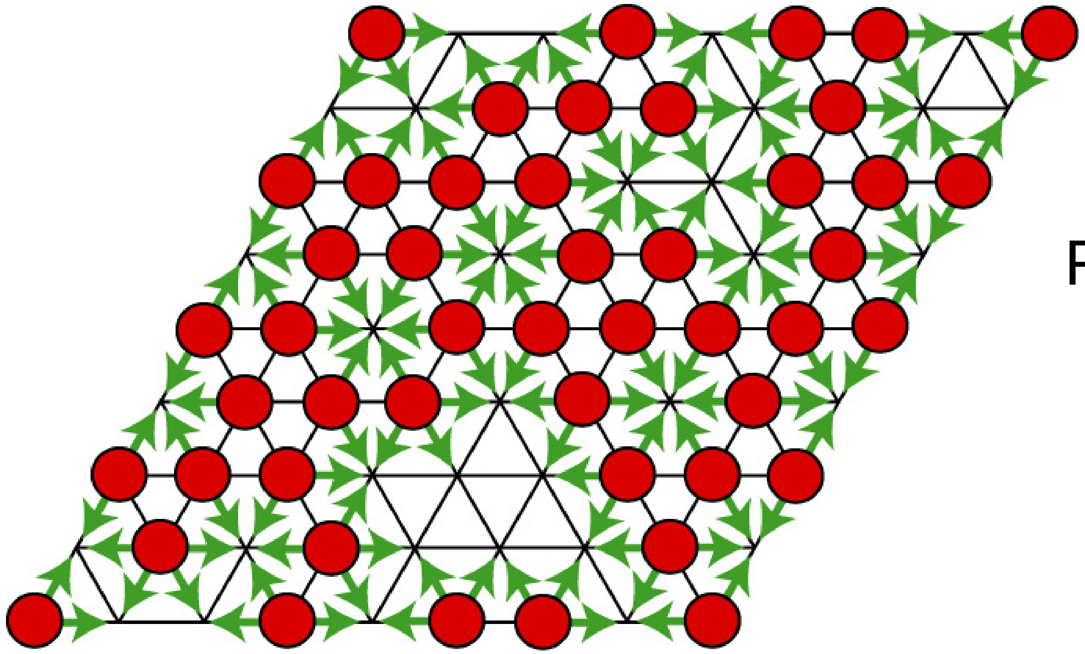
$$W_i = \nu * \exp\left(\frac{-\Delta E_i}{k_B T}\right)$$



For each potential hop i, calculate the hop rate

$$W_i = \nu * \exp\left(\frac{-\Delta E_i}{k_B T}\right)$$

Then randomly choose a hop *k*, with probability W_k

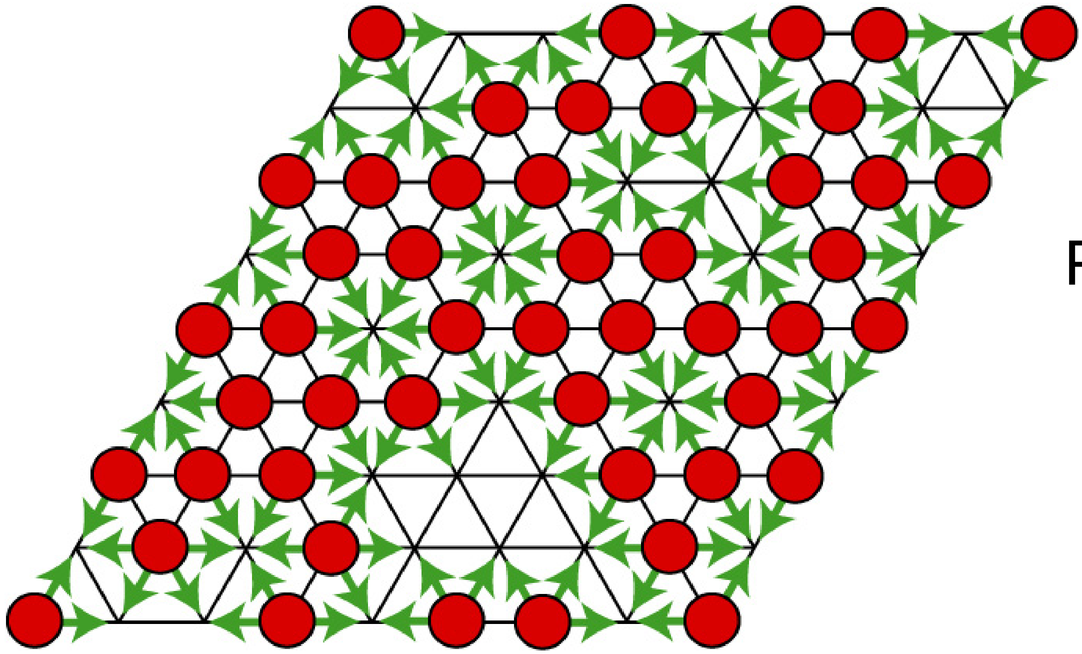


For each potential hop i, calculate the hop rate

$$W_i = \nu * \exp\left(\frac{-\Delta E_i}{k_B T}\right)$$

Then randomly choose a hop *k*, with probability W_k

ξ_1 = random number



For each potential hop i, calculate the hop rate

$$W_i = \nu * \exp\left(\frac{-\Delta E_i}{k_B T}\right)$$

Then randomly choose a hop *k*, with probability W_k

ξ_1 = random number

$$\sum_{i=1}^{k-1} W_i < \xi_1 \cdot W \leq \sum_{i=0}^{k} W_i \qquad W = \sum_{i=0}^{N_{hops}} W_i$$

Time

After hop *k* we need to update the time

ξ_2 = random number

$$\Delta t = -\frac{1}{W}\log \xi_2$$

Two independent stochastic variables: the hop k and the waiting time Δt

$$\sum_{i=1}^{k-1} W_i < \xi_1 \cdot W \le \sum_{i=0}^{k} W_i$$

$$W_i = \nu^* \exp\left(\frac{-\Delta E_i}{k_B T}\right)$$

$$W = \sum_{i=0}^{N_{hops}} W_i$$

$$\Delta t = -\frac{1}{W} \log \xi_2$$

Kinetic Monte Carlo

- Hop every time
- Consider all possible hops simultaneously
- Pick hop according its relative probability
- Update the time such that Δt on average equals the time that we would have waited in standard Monte Carlo