We use conventional Metropolis Monte Carlo (MMC) Algorithm to simulate protein clustering.
Each protein with length L can interact with its nearest neighbors (without head-to-head interaction as in Fig.1).

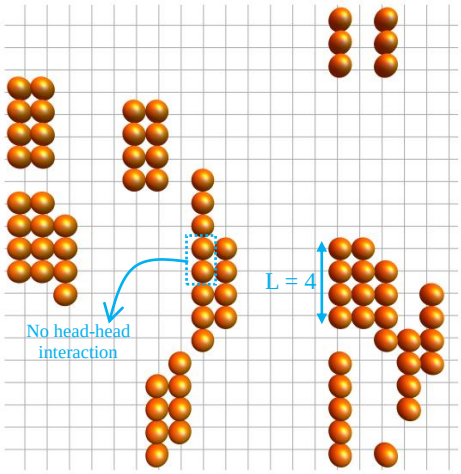

Fig.1 Nearest neighbor interaction

We initialize a 3-dimensional simulation box with N randomly distributed blocks of length L. At each step, we apply the MMC algorithm as following:
1) choose a random block,
2) propose a new (random) position,
3) calculate the energy different ΔE, if new position were to change,
4) generate a random number $0 < r < 1$,
5) if $r < \exp(-\Delta E)$, accept the new position,
6) repeat from step 2.

We also impose the periodic boundary condition on the simulation box as in Fig. 2. Because we are interested in the equilibrium states of the clusters, the new proposed position may be non-physical (chosen block can pass through others) if it obeys the detailed balance.

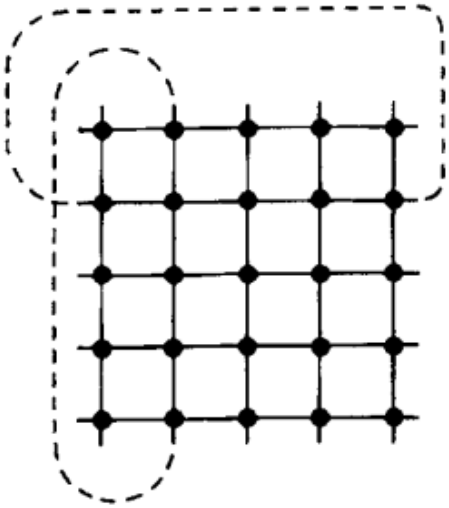

Fig.2 Periodic boundary condition

We tested the code on a small system with the following parameters:
box size: 150x150x150, interaction energy: 0.6 (k_BT), block length: 5 (units or monomers), total blocks: 20250, total accepted: moves 3 billion.

The results were promising, there were some clear clusters at the end of the simulations. Below are the snapshots from one of the simulations.

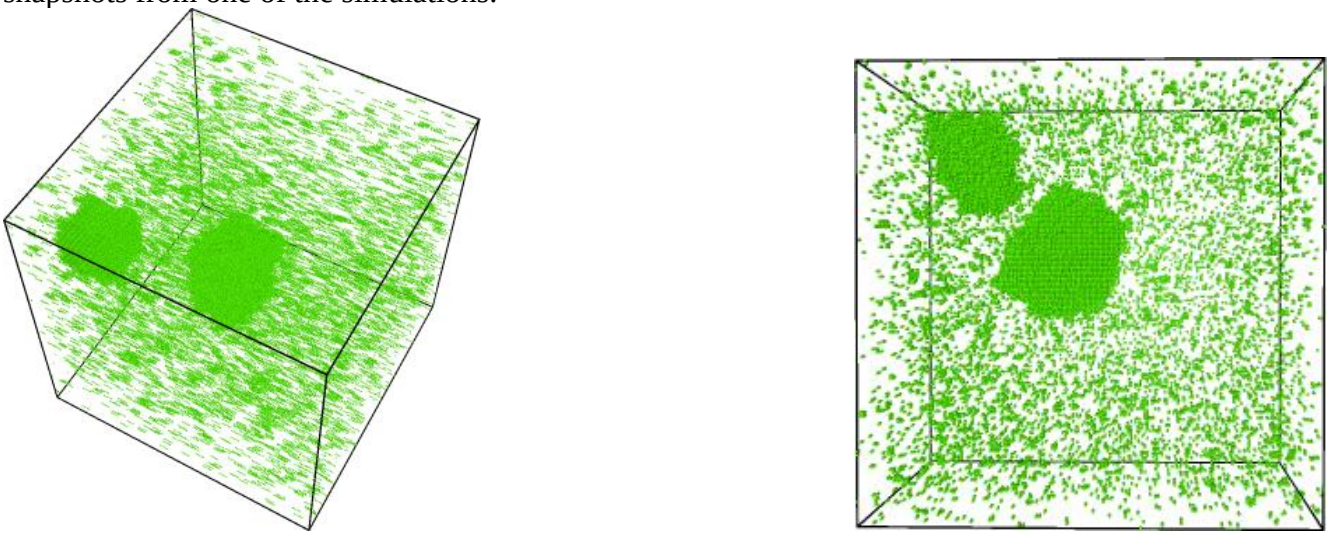

Fig. 3 Snapshots

We want to study the aspect ratio of the biggest cluster as a function of *interaction energy* (and *length*). If we increase the energy or block length, at some point we must increase the box size and as a result it takes longer running time.
I think there is much room for improvement. In the code, I use two arrays to keep track of positions of the blocks: one 3D array for storing the blocks labeled from 1 to N, another 2D array for storing coordinates. I suspect doing this way may affect the performance.
I am learning a way to write the code in parallel, which means I can propose multiple moves at the same time. It turns out a challenging problem because I must decompose the domain so that one update from an accepted move does not affect the other moves. Please help me!

I compile the code using intel compiler:

```
icpc -std=c++11 -O3 MMC3D.cpp -o MMC3D
```

and run it as

```
./MMC3D --bondEn 0.6 --iterations 3000000000 --split 1000000000 --blocks
20250 --length 5 --runId 1
```

`--bondEn:` interaction energy
`--iterations:` total number of accepted moves
`--splits:` frequency of saving files
`--blocks:` total blocks in simulation box
`--length:` total units in a protein
`--runId:` number for submitting jobs in array.

If you have any questions, please email me
`minhtien@phys.ksu.edu`