Algorithm-Non Equilibrium Statistical Mechanics

M. S. Rahman

March 2020

Contents

Chapter 1

Square Lattice Percolation

This document is a common guide but it applies mainly to version 12 or $v12$.

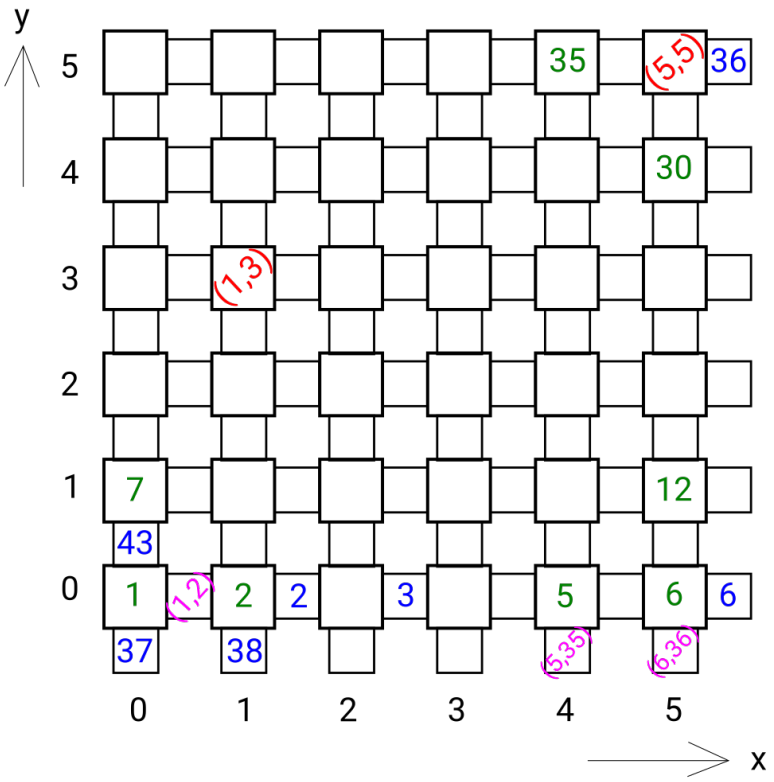
1.1 Components

1.1.1 Site

1. *id* or *label*
2. *group_id*
3. Index
4. Relative Index with respect to

1.1.2 Bond

1. *id* or *label*
2. *group_id* which can coincide with site group id
3. Index. (a, b) where a is the *id* of one site and b is the id of another site.



((a))

Figure 1.1: Red color is for site index, Green color is for site id(or label), Magenta color is for bond index, Blue color is for bond id (or label).

1.1.3 Lattice

We will study $2D$ system specifically with a $2d$ vector in C++.

1. Site.
2. Bond

All the information of sites and bonds must be accessed through lattice.

A $1d$ vector looks like

$$V_i = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

A $2d$ vector looks like

$$V_{i,j} = \begin{bmatrix} 0,0 & 0,1 & 0,2 \\ 1,0 & 1,1 & 1,2 \\ 2,0 & 2,1 & 2,2 \end{bmatrix}$$

But we want grid like structure

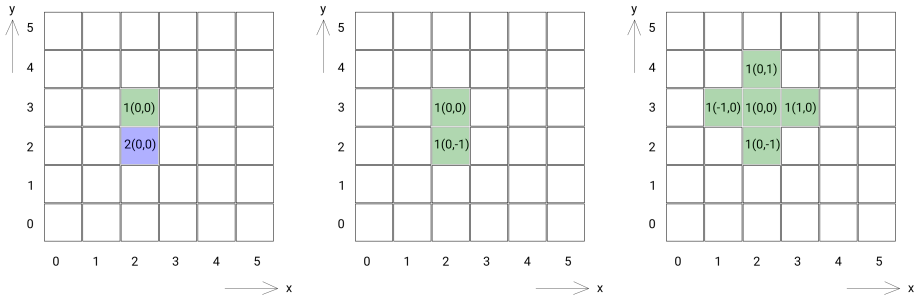
$$V'_{i,j} = \begin{bmatrix} 0,2 & 1,2 & 2,2 \\ 0,1 & 1,1 & 2,1 \\ 0,0 & 1,0 & 2,0 \end{bmatrix}$$

where each column of V' is row of V but backwards. So when viewing the lattice we just need to generate row index backward and switch row and column index. Note that horizontal bonds become vertical and vice versa in this process.

1.1.4 Cluster

A cluster contains site and bond ids as a list. So that when asked which site or bond it contains, cluster class can return the list.

1. site id list
2. bond id list
3. group id or *gid*. This is very important. All site and bonds must have this same *gid* through which we can determine which cluster a particular site or bond belongs to.



((a)) ((b)) ((c))

Figure 1.2: Relative index of any site is by default is $(0, 0)$ even if they have different group id (1.2(a)). Say we relabel 2 with respect to 1 then we get (1.2(b)). And if the cluster grows a bit it might look like (1.2(c)).

1.1.5 Relative Index

To detect wrapping relative index is very useful. This only applies to sites. Relative index is the relative position of sites with respect to the root site (first site of a cluster).

Expression is $g_{id}(x_r, y_r)$ where subscript r indicates relative index. Relative index have default value $(0, 0)$.

1.1.6 cluster relabeling by relative index

For relative index transformation we just need the relative index of two neighbor of different cluster. Say cluster i have site with relative index (x_{ir}, y_{ir}) and coordinate index (x_{ic}, y_{ic}) and cluster j have site (x_{jr}, y_{jr}) and coordinate index (x_{jc}, y_{jc}). With appropriate transformation the relative index of cluster j will be changed by

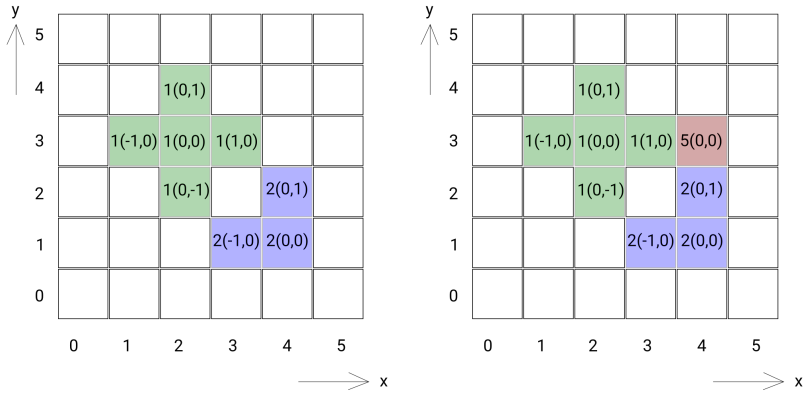
$$\begin{aligned} x_j &\to x_j + \Delta x_r + \Delta x_c \\ y_j &\to y_j + \Delta y_r + \Delta y_c \end{aligned} \tag{1.1}$$

Simplifying

$$\begin{aligned} x_j &\to x_j + \Delta x \\ y_j &\to y_j + \Delta y \end{aligned} \tag{1.2}$$

With

$$\begin{aligned} \Delta x &= \Delta x_r + \Delta x_c \\ \Delta y &= \Delta y_r + \Delta y_c \end{aligned} \tag{1.3}$$



((a)) ((b))

Figure 1.3: Suppose we have two large cluster 1 and 2 which is indicated by green and blue color in the figure (1.3(a)). A third site (red) is occupied which has group id 5 and default relative index (1.3(b)) and we decide to connect it with blue cluster then it's relative index will change from $(0, 0)$ to $(0, 2)$ according to it's neighbor $2(0, 1)$.

Where

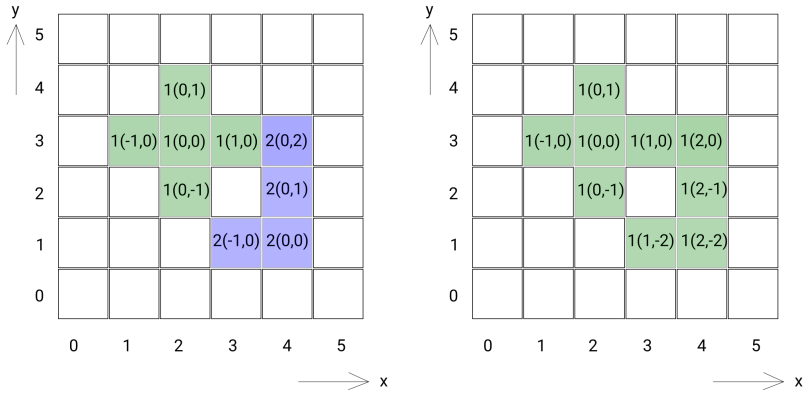
$$\Delta x_r = x_{ir} - x_j r \tag{1.4}$$
$$\Delta y_r = y_{ir} - y_j r$$
$$\Delta x_c = -x_{ic} + x_j c \tag{1.5}$$
$$\Delta y_c = -y_{ic} + y_j c$$

This is demonstrated in figure (1.3, 1.4) where i is the green cluster and j is the blue cluster.

1.1.7 Detect Wrapping using relative index

Wrapping detection is done using relative index. In fact this is the only reason behind using relative index.

At some state lattice can reach a state like (1.5(a)) where just one site at index $(2, 5)$ can trigger wrapping. We mark it by red color (1.5(b)) which is relabeled by site at index $(3, 5)$ with relative index $(1, -4)$. The red site becomes green by relative index transformation and acquires new relative index $(1, -5)$. Note that Index of any site is unchanged only relative index changes. Now we compare neighbor sites of new site indexed $(2, 5)$. One is showed in figure (1.5(c)), we can easily see that the absolute value of difference between y value of the relative index is greater than 1 which indicates



((a)) ((b))

Figure 1.4: Thus we have blue cluster of size 4 and group id 2 (1.4(a)). Then we need to merge blue and green cluster together. Green neighbor of $1(1,0)$ is $2(0,2)$. The transformation of is $2(0,2)$ goes to $1(2,0)$, i.e., we add $(\Delta x, \Delta y) = (2,-2)$ to all the element of blue cluster and we are done (1.4(b)). Here $\Delta x_r = 1$ and $\Delta y_r = -2$ for relative index difference and $\Delta x_c = 1$ and $\Delta y_c = 0$ for coordinate index difference.

that the cluster wrapped around the lattice once vertically. If absolute value of difference between x values were greater than 1 then we would say it is a horizontal wrapping.

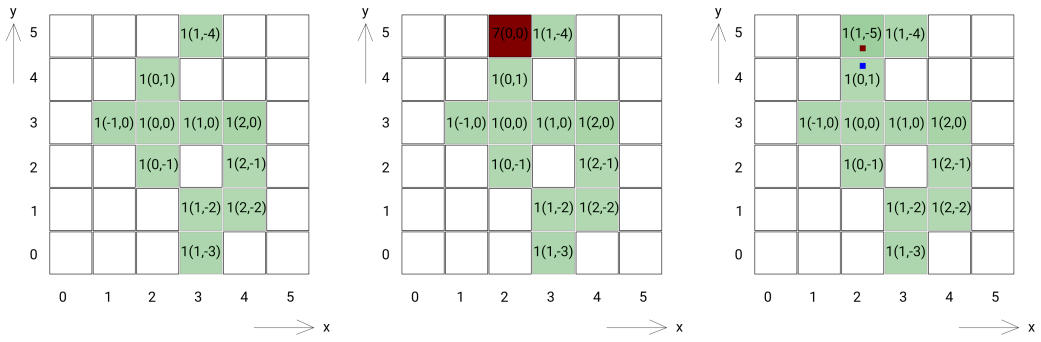
1.2 Calculating Entropy Efficiently

There are three different ways of calculating Shannon entropy. Theoretically all are equivalent but due to computer precision these different methods are slightly different which is visible when calculating specific heat, i.e., derivative of entropy.

1.2.1 method 1: Slow Entropy measuring

In this method one just perform the sum over all cluster to get the entropy by using the definition for Shannon entropy.

$$H = -\sum_{i}^{n} \mu_i \log \mu_i \tag{1.6}$$



((a)) ((b)) ((c))

Figure 1.5

where n is the number of cluster and $\mu_i = \frac{\text{size of i-th cluster}}{\text{sum of size of all cluster}}$ is the cluster picking probability.

Although this method is the slowest but it is the most accurate if suitable data type is used, e.g. *long double* in $C++$.

1.2.2 method 2: Fastest entropy measuring

At each step of percolation process we join smaller clusters to one big cluster and thus entropy changes. Say cluster $A_i : i = 10, 11, 12, 13$'s are merged together to form cluster B then we can subtract entropy for $A_i : i = 10, 11, 12, 13$ and add entropy for $B_j : j = 10$ to the total entropy. This method is the fastest and theoretically is the same but due to computer precision it may give a slightly different result for large system. This is visible if we calculate specific heat.

1.2.3 method 3

We create a list of size n, where n is the maximum number of cluster the system can have. We store i-th value of the sum in equation (1.6) to the i-th element of this list and call it $entropy_list$. Now if we perform a sum over all the elements of this array then we get the total entropy.

If we are to merge $A_i : i = 10, 11, 12, 13$ clusters to cluster $B_j : j = 10$ then we set $i = 10, 11, 12, 13$-th element of $entropy_list$ to zero and merge the clusters. Then we calculate the entropy for B_j cluster and store it in the j-th element.

Note that if $i = 10, 11, 12, 13$ then j will be one of these i indices, meaning out of the four element that we have set to zero, one will regain a non zero

length	method	time (sec)	accuracy
200	1	62	best
200	2	0.122	–
200	3	3.84	to be measured

Table 1.1: comparison of different entropy calculation methods

value.

1.2.4 Time comparison