The loop update is an example of a cluster algorithm. Detour:

Cluster algorithm for the Ising model

Define bond index corresponding to pair of interacting spins

$$\text{bond} \quad b = 1, 2, \ldots, N_b, \quad \text{interacting spins } \sigma_{i(b)}, \sigma_{j(b)}$$

Number of bonds N_b = dN for a d-dimensional cubic lattice

Write the energy of the Ising ferromagnet as

$$E = -|J| \sum_{b=1}^{N_b} [\sigma_{i(b)} \sigma_{j(b)} + 1] = -\sum_{b=1}^{N_b} E_b$$

Write the partition function as

$$Z = \sum_{\sigma} \mathrm{e}^{-E(\sigma)/T} = \sum_{\sigma} \prod_{b=1}^{N_b} \mathrm{e}^{E_b/T} = \sum_{\sigma} \prod_{b=1}^{N_b} [1 + (\mathrm{e}^{E_b/T} - 1)]$$

Define bond functions with arguments 0,1 (bond variable):

$$F_b(0) = 1$$
$$F_b(1) = \mathrm{e}^{E_b/T} - 1$$

$$Z = \sum_{\sigma} \prod_{b=1}^{N_b} [F_b(0) + F_b(1)]$$

Introduce bond variables

$$\tau_b = 0, 1, \quad \tau = \{\tau_1, \tau_2, \ldots, \tau_{N_b}\}$$

Partition function can be written as sum over spins and bonds

$$Z = \sum_{\sigma} \prod_{b=1}^{N_b} [F_b(0) + F_b(1)] = \sum_{\sigma} \sum_{\tau} \prod_{b=1}^{N_b} F_b(\tau_b)$$

The functions F_b depend on the spins:

$$F_b(0) = 1$$

$$F_b(1) = \mathrm{e}^{E_b/T} - 1 = \begin{cases} \mathrm{e}^{2|J|/T} - 1, & \text{if } \sigma_{i(b)} = \sigma_{j(b)} \\ 0, & \text{if } \sigma_{i(b)} \neq \sigma_{j(b)} \end{cases}$$

$\tau_b = 1$ allowed only between parallel spins

Probabilities: For everything else fixed, probability for a given b

$$P(\tau_b) = \frac{F(\tau_b)}{F(0) + F(1)} = \frac{F(\tau_b)}{\mathrm{e}^{2|J|/T}}$$

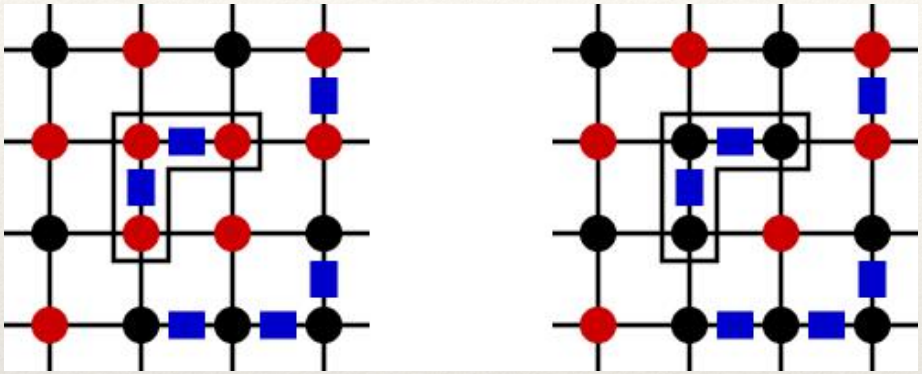
If parallel spins on bond b, probabilities for the bond variable

$$P(\tau_b = 0) = \mathrm{e}^{-2|J|/T}, \quad P(\tau_b = 1) = 1 - \mathrm{e}^{-2|J|/T}$$

If anti-parallel spins on bond b

$$P(\tau_b = 0) = 1, \quad P(\tau_b = 1) = 0$$

For a fixed bond configuration, spins forming clusters (spins connected by "filled" bonds) can be flipped and then give a configuration (term) with the same weight in Z (F_b=1 for all bonds between clusters, F_b unchanged inside cluster).



$$N(\tau_b = 1) = \text{ No. of filled bonds}$$

$$W = (e^{2|J|/T} - 1)^{N(\tau_b=1)}$$

(unchanged after flip)

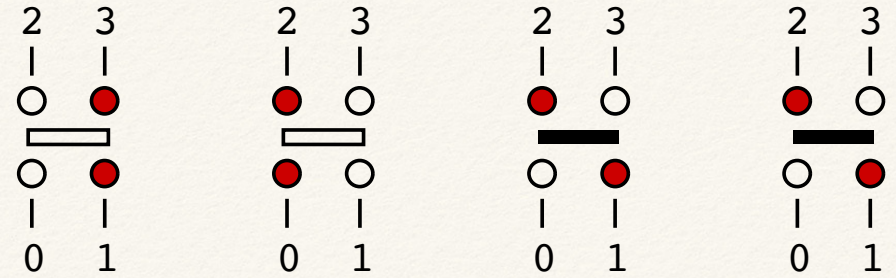
Spins not connected to any filled bonds are single-spin clusters

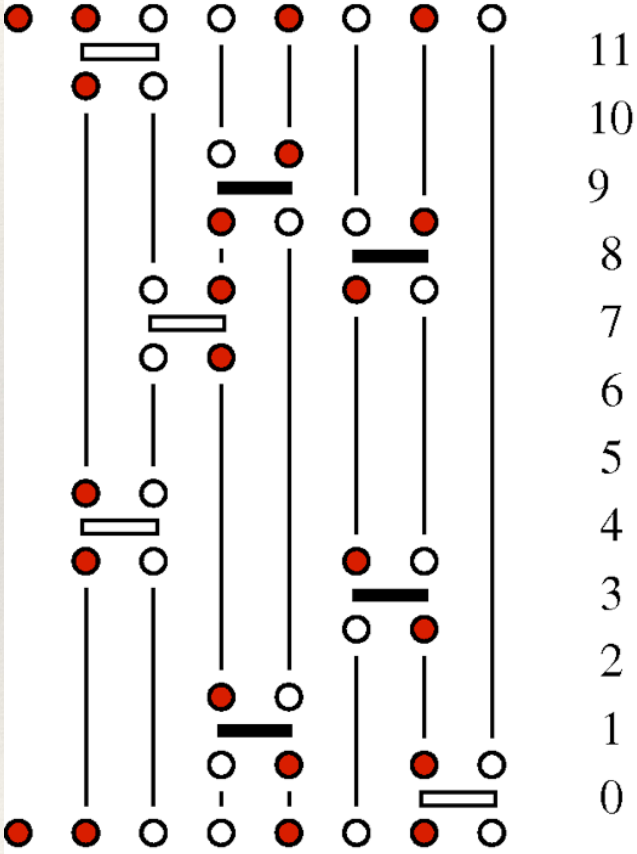
Swendsen-Wang algorithm

- Start from spin configuration
- Generate bond configuration
- Identify clusters of spins connected by bonds
- Flip each cluster with probability 1/2
- Generate new bonds with the current spins, etc

SSE: Linked vertex storage for loop update

The “legs” of a vertex represents the spin states before (below) and after (above) an operator has acted





p	v	X(v)	v	X(v)	v	X(v)	v	X(v)
11	44	18	45	30	46	16	47	17
10	40	-	41	-	42	-	43	-
9	36	31	37	7	38	4	39	5
8	32	14	33	15	34	12	35	0
7	28	19	29	6	30	45	31	36
6	24	-	25	-	26	-	27	-
5	20	-	21	-	22	-	23	-
4	16	46	17	47	18	44	19	28
3	12	34	13	2	14	32	15	33
2	8	-	9	-	10	-	11	-
1	4	38	5	39	6	29	7	37
0	0	35	1	3	2	13	3	1
	l=0		l=1		l=2		l=3	

X( ) = vertex list

- operator at p→X(v)
  v=4p+l, l=0,1,2,3
- links to next and previous leg

Spin states between operations are redundant; represented by links

- network of linked vertices will be used for loop updates of vertices/operators

Pseudocode: Sweep of loop updates

constructing all loops, flip probability 1/2

```
do v_0 = 0 to 4L − 1 step 2
    if (X(v_0) < 0) cycle
    v = v_0
    if (random[0 − 1] < 1/2) then
        traverse the loop; for all v in loop, set X(v) = −1
    else
        traverse the loop; for all v in loop, set X(v) = −2
        flip the operators in the loop
    endif
enddo
```

- visited vertices are no longer needed and we set them to a negative value -1 or -2, to indicate that the loop has been visited (-1) or visited and flipped (-2)

construct and flip a loop

```
v = v_0
do
    X(v) = −2
    p = v/4; s(p) = flipbit(s(p), 0)
    v′ = flipbit(v, 0)
    v = X(v′); X(v′) = −2
    if (v = v_0) exit
enddo
```

- p is the location of the operator in the original length-L list of operatotors
- by flipping bit 0 of s(p), the operator changes from diagonal to off-diagonal, or vice versa
- moving on the vertex to the adjacent spin is also done with a bit flip

We also have to modify the stored spin state after the loop update

- we can use the information in $V_{first}()$ and X() to determine spins to be flipped
- spins with no operators, $V_{first}(i)=-1$, flipped with probability 1/2

```
do i = 1 to N
     v = V_first(i)
     if (v = -1) then
          if (random[0-1]< 1/2) σ(i) = -σ(i)
     else
          if (X(v) = -2) σ(i) = -σ(i)
     endif
enddo
```

$v=V_{first}(i)$ is the location of the first vertex leg on site i

- flip the spin if X(v)=–2
- (do not flip it if X(v)=–1)
- no operation on i if $v_{first}(i)=-1$; then it is flipped with probability 1/2

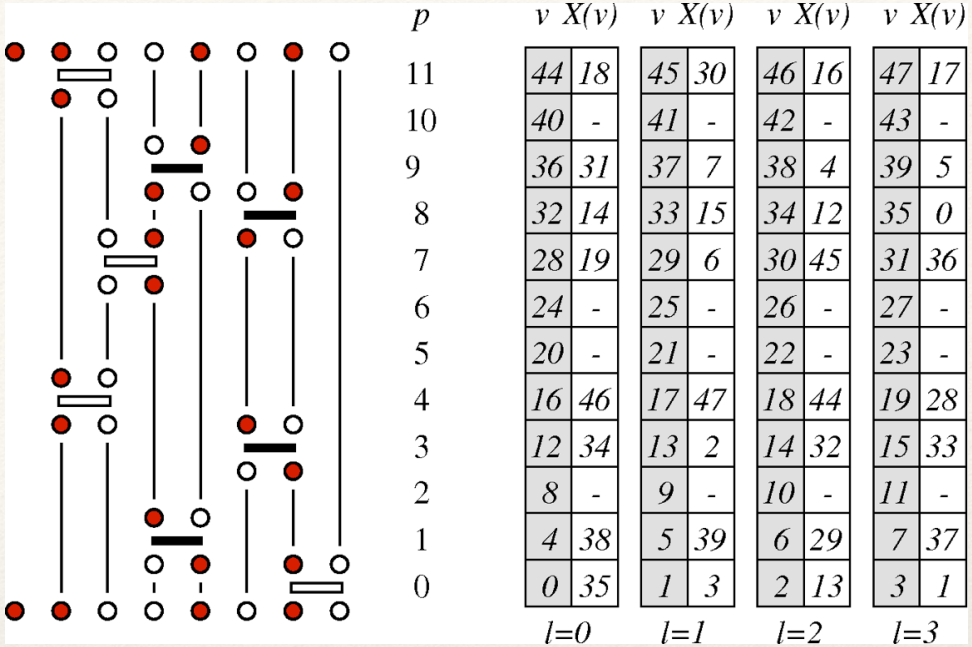
Constructing the linked vertex list

Traverse operator list *s(p)*, *p*=0,...,*L*–1

- vertex legs **v=4p,4p+1,4p+2,4p+3**

Use arrays to keep track of the first and last (previous) vertex leg on a given spin

- $V_{first}(i)$ = location v of first leg on site i
- $V_{last}(i)$ = location v of last (currently) leg
- these are used to create the links
- initialize all elements to –1



$V_{\rm first}(:) = -1;\ V_{\rm last}(:) = -1$
do $p = 0$ **to** $L-1$
  if $(s(p) = 0)$ **cycle**
  $v_0 = 4p;\ b = s(p)/2;\ s_1 = i(b);\ s_2 = j(b)$
  $v_1 = V_{\rm last}(s_1);\ v_2 = V_{\rm last}(s_2)$
  if $(v_1 \neq -1)$ **then** $X(v_1) = v_0;\ X(v_0) = v_1$ **else** $V_{\rm first}(s_1) = v_0$ **endif**
  if $(v_2 \neq -1)$ **then** $X(v_2) = v_0;\ X(v_0) = v_2$ **else** $V_{\rm first}(s_2) = v_0 + 1$ **endif**
  $V_{\rm last}(s_1) = v_0 + 2;\ V_{\rm last}(s_2) = v_0 + 3$
enddo

creating the last links across the "time" boundary

do $i = 1$ **to** N
  $f = V_{\rm first}(i)$
  if $(f \neq -1)$ **then** $l = V_{\rm last}(i);\ X(f) = l;\ X(l) = f$ **endif**
enddo

Determination of the cut-off L

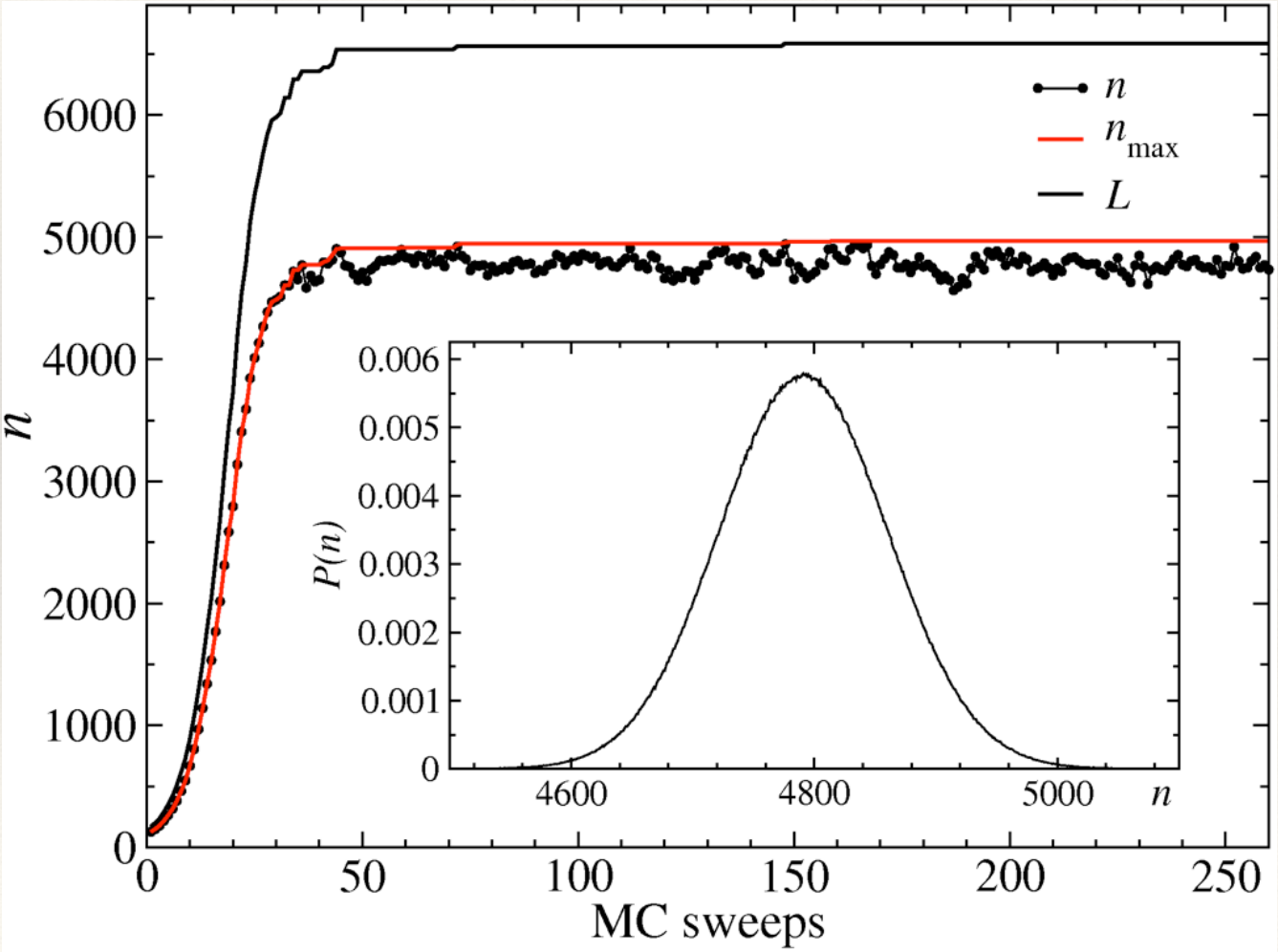
- adjust during equilibration
- start with arbitrary (small) n

Keep track of number of operators n

- increase L if n is close to current L
- e.g., ***L=n+n/3***

Example

- 16×16 system, β=16 ⇒
- evolution of L
- n distribution after equilibration
- truncation is no approximation

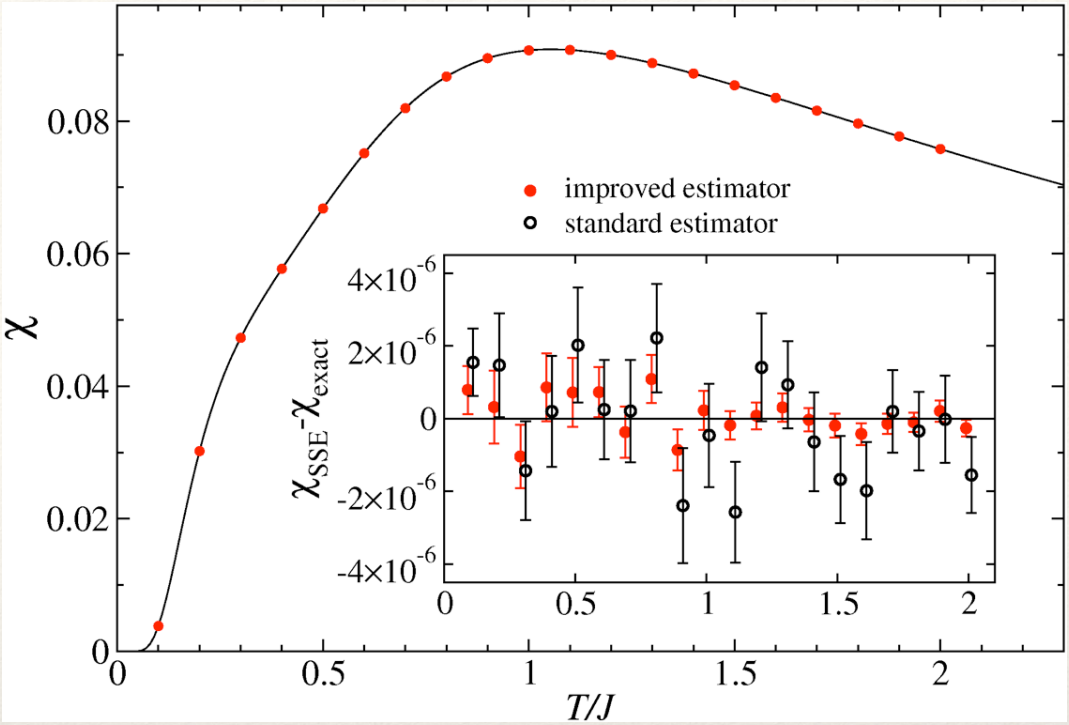


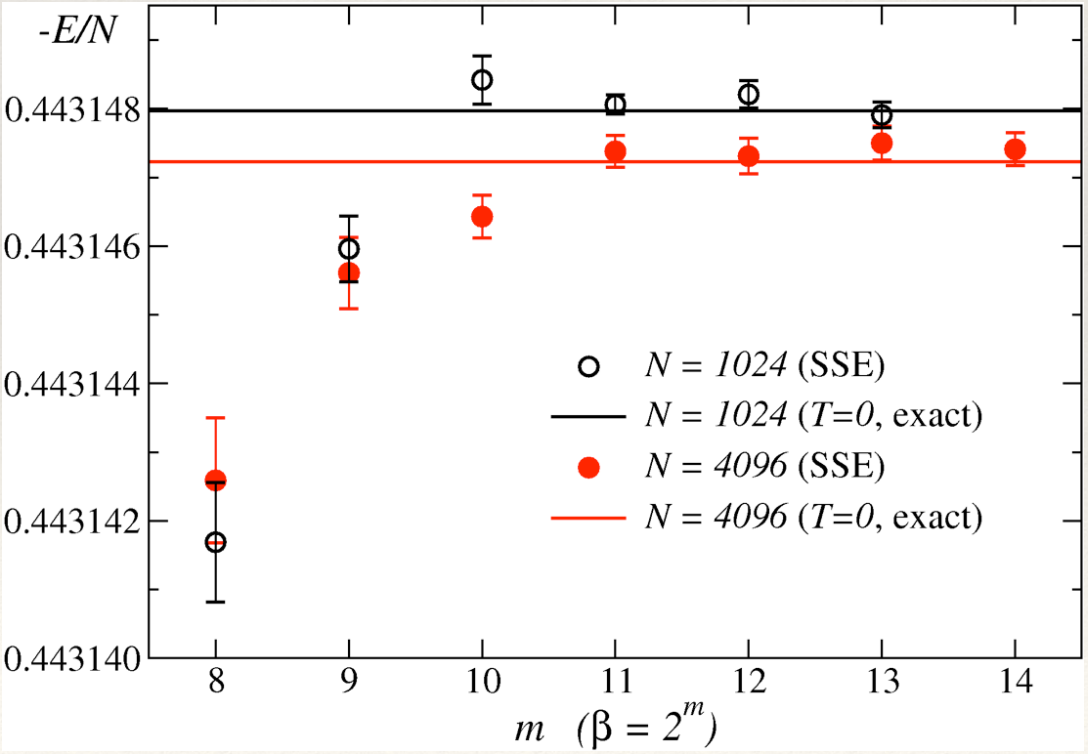
Does it work?
Compare with exact results

- 4×4 exact diagonalization
- Bethe Ansatz; long chains

Susceptibility of the 4×4 lattice ⇒

- SSE results from 10^{10} sweeps
- improved estimator gives smaller error bars at high T (where the number of loops is larger)





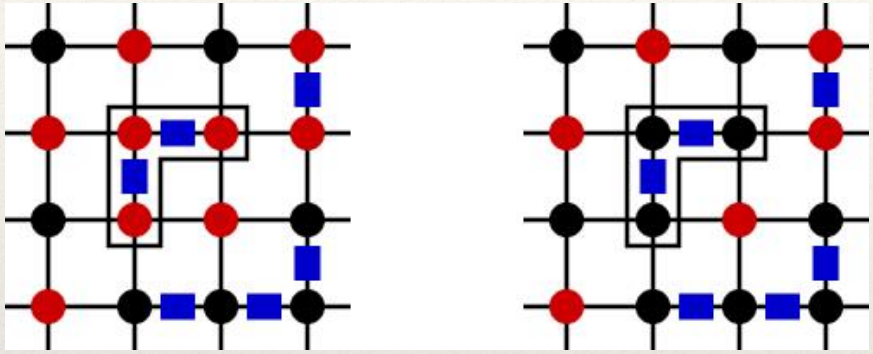
⇐ Energy for long 1D chains

- SSE results for 10^6 sweeps
- Bethe Ansatz ground state E/N
- SSE can achieve the ground state limit (T→0)

Improved Estimators

SSE with loop updates is an example of a cluster algorithms
- we can utilize improved estimators for many observables

Classical example: Swendsen-Wang Ising cluster algorithm



$N(\tau_b = 1) =$ No. of filled bonds

$$W = (e^{2|J|/T} - 1)^{N(\tau_b=1)}$$

(unchanged after flip)

Write magnetization as sum over clusters of size n_C, sign s_C:

$$M = \sum_{i=1}^{N} \sigma_i = \sum_{C=1}^{N_{\rm clus}} \sum_{i \in C} \sigma_i = \sum_{C=1}^{N_{\rm clus}} s_C n_C \qquad \langle M^2 \rangle = \sum_{C=1}^{N_{\rm clus}} \sum_{C'=1}^{N_{\rm clus}} \langle n_C n_{C'} s_C s_{C'} \rangle$$

All cluster orientations (signs) have same weight
- average over all $2^{N_{\rm clus}}$ orientations →

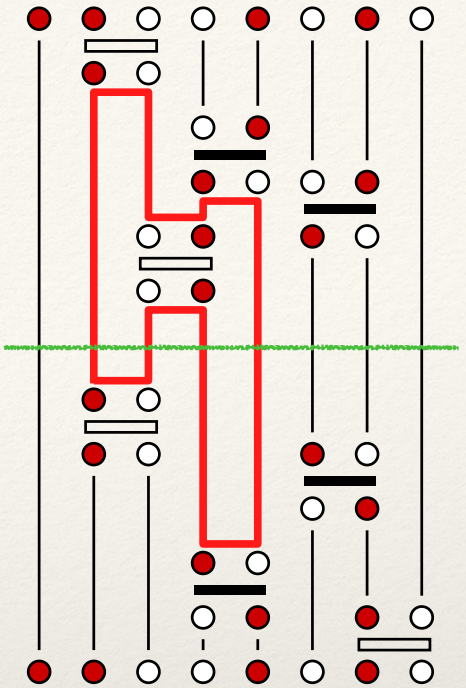
$$\langle M^2 \rangle = \sum_{C=1}^{N_{\rm clus}} \langle n_C^2 \rangle$$

This is the improved estimator of $\langle M^2 \rangle$
- only depends on cluster structure

Improved estimators in SSE

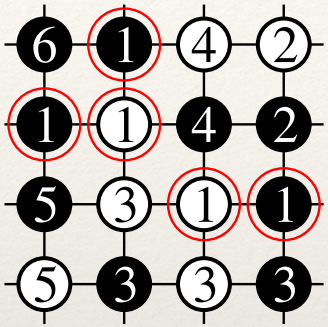
Consider a given slice (propagated state) of an SSE configuration

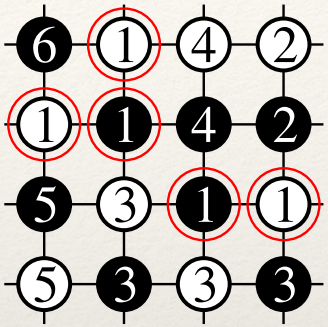
- label the sites according to the loops passing through → clusters



In given loop

All spins on given sub-lattice A or B are same, different on A, B





Staggered magnetization on a cluster is 1/2 of the size of the cluster

- changes sign when loop flipped
- similar to magnetization in SW algorithm

$$\langle M^2_{z,\mathrm{stagg}} \rangle = \frac{1}{4} \sum_{C=1}^{N_{\mathrm{clus}}} \langle n_C^2 \rangle$$

The uniform magnetization requires the staggered phases

$$\chi = \frac{\beta}{4N} \left\langle \sum_{j=1}^{C} \left(\sum_{i=1}^{n_j} \phi_i \right)^2 \right\rangle \qquad \phi_i = \begin{cases} +1 & i \text{ on A site} \\ -1 & i \text{ on B site} \end{cases}$$

Transverse-field Ising model [PRE 68, 056701]

$$H = \sum_{i,j} J_{ij}\sigma_i^z\sigma_j^z - h\sum_i \sigma_i^x$$

Arbitrary interactions (incl. random, long-range,…)

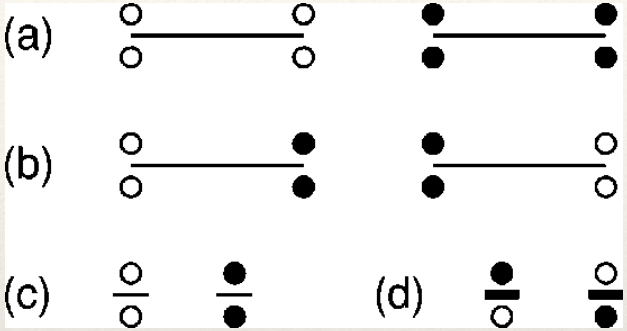
Elementary operators for the SSE strings:

$$H_{0,0} = 1,$$

$$H_{i,0} = h(\sigma_i^+ + \sigma_i^-), \qquad i>0,$$

$$H_{i,i} = h, \qquad i>0,$$

$$H_{i,j} = |J_{ij}| - J_{ij}\sigma_i^z\sigma_j^z, \qquad i,j>0,\ i\neq j$$



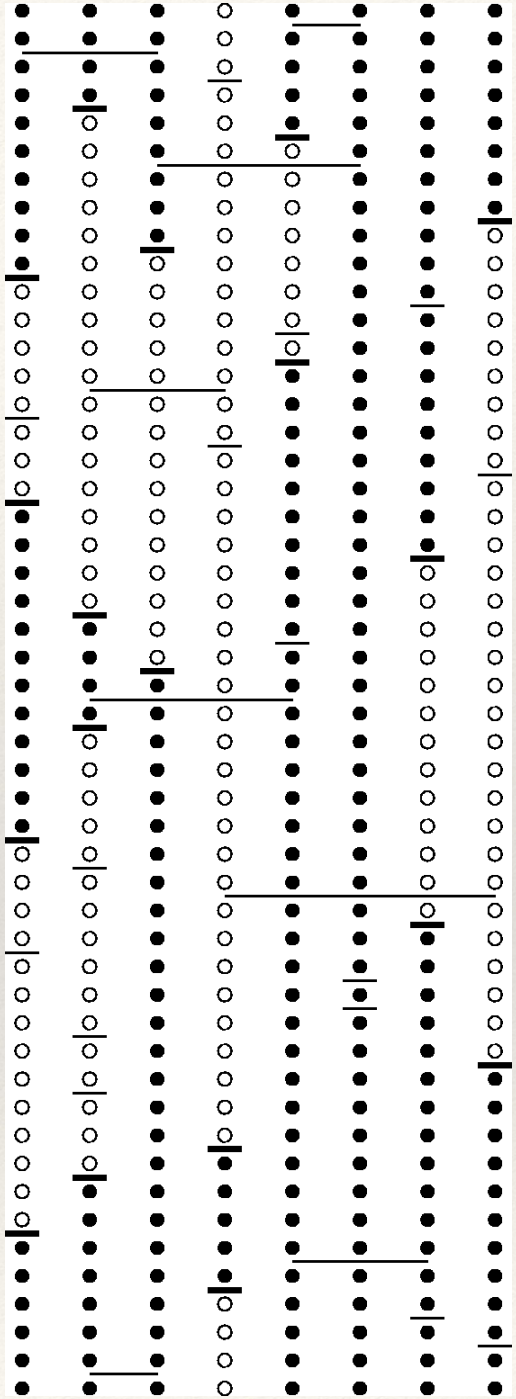
$$H = -\sum_{i=1}^{N}\sum_{j=0}^{N} H_{i,j} \qquad S_n = [i(1),j(1)],\ldots,[i(n),j(n)]$$

$$Z = \frac{1}{L!}\sum_{\alpha}\sum_{S_L} \beta^n (L-n)!\langle\alpha|\prod_{l=1}^{L} H_{i(l),j(l)}|\alpha\rangle$$

Possible local updates:

$$[0,0]_p \leftrightarrow [i,j]_p, \qquad i,j\neq 0, \qquad \text{(diagonal)}$$

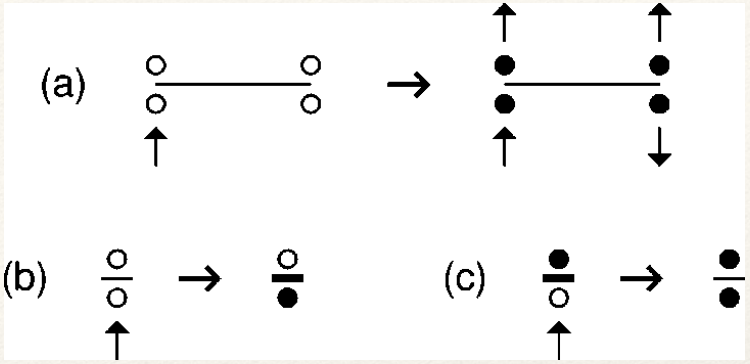
$$[i,i]_{p_1}[i,i]_{p_2} \leftrightarrow [i,0]_{p_1}[i,0]_{p_2}, \qquad i\neq 0 \qquad \text{(off-diagonal)}$$



Cluster update for Transverse-field Ising models

Start at arbitrary in-leg and apply weight-preserving operator changes

- growing “to do” list (stack) from each out-leg
- until stack is empty
- divides the entire system into clusters
- similar to classical S-W algorithm



Long-range interactions

Goal: avoid summations over interactions J_{ij}

- we can first assume that all diagonal operators are allowed, then

$$P([0,0]\to[i,j]) = \frac{\beta\left(Nh + 2\sum_{ij}|J_{ij}|\right)}{L-n+\beta\left(Nh+2\sum_{ij}|J_{ij}|\right)} \qquad P([i,j]\to[0,0]) = \frac{L-n+1}{L-n+1+\beta\left(Nh+2\sum_{ij}|J_{ij}|\right)}$$

where [i,j] in [0,0]->[i,j] is a still undetermined interaction

- the actual [i,j] is chosen in a second step using cumulative probabilities:

$$P_c(k) = \frac{\sum_{i=1}^{k} P(i)}{\sum_{i=1}^{N} P(i)}$$

N is the total number of diagonal ops

Choose op #k using bisection in the table $P_c()$

- time scales as ln(N)

Cluster update remains the same for any kind of interaction

- total time for updating sweep scales as βNln(N) with long-range interactions