量子磁性与多体计算培训班（2024.6.12-2024.6.14）北京计算科学研究中心

Stochastic Series Expansion Quantum Monte Carlo

王艳成

ycwangphys@buaa.edu.cn

北航杭州国际创新研究院（学院）

2024-6-12

北航·杭州国际校园

References

1. *Quantum Monte Carlo simulations for spin systems*, Anders W. Sandvik and Juhanni Kurkijärvi, Phys. Rev. B 43, 5950 (1991).
2. *Stochastic series expansion method with operator-loop update*, Anders W. Sandvik, Phys. Rev. B 59, R14157 (1999).
3. *Quantum Monte Carlo with directed loops*, Olav F. Syljuåsen, and Anders W. Sandvik, Phys. Rev. E 66, 046701 (2002).
4. *Directed loop updates for quantum lattice models*, Olav F. Syljuåsen, Phys. Rev. E 67, 046701 (2003).
5. *A generalization of Handscomb's quantum Monte Carlo scheme-application to the 1D Hubbard model*, Anders W. Sandvik, J. Phys. A: Math. Gen. 25, 3667 (1992).
6. *Generalized directed loop method for quantum Monte Carlo simulations*, Fabien Alet, Stefan Wessel, and Matthias Troyer, Phys. Rev. E 71, 036706 (2005).
7. *Computational Studies of Quantum Spin Systems*, Anders W. Sandvik, AIP Conf. Proc. 1297, 135–338 (2010).
8. 2023多体计算讲习班：张学锋“*Stochastic Series Expansion*”，蔻享学术。

统计物理回顾

配分函数（Partition function）：

$$Z = \sum_{\alpha} e^{-\beta E_\alpha} = tr\left(e^{-\beta H}\right)$$

内能 $$E = tr\left(He^{-\beta H}\right) = -\frac{dlnZ}{d\beta}$$

自由能 $$F = -k_B T lnZ$$

压强 $$P = -k_B T \left(\frac{\partial lnZ}{\partial V}\right)_T$$

Markov Chain Monte Carlo

- **概率分布** $P(\Gamma)$

  观察量A的期望值：$\langle A \rangle = \dfrac{\int P(\Gamma)\,\mathrm{A}(\Gamma)\mathrm{d}\Gamma}{\int P(\Gamma)\,\mathrm{d}\Gamma}$

- **MCMC目标：**产生满足概率分布$P(\Gamma)$的一系列微观态

$$\Gamma_1 \rightarrow \Gamma_2 \rightarrow \Gamma_3 \rightarrow \cdots \ \Gamma_{i-1} \rightarrow \Gamma_i \rightarrow \Gamma_{i+1} \rightarrow \cdots \Gamma_N$$

$$\langle A \rangle = \frac{\sum_{\mathrm{i}=1}^{N} \mathrm{A}(\Gamma_i)}{N}$$

- **两个保证：**
  - 细致平衡（Detail balance） $W(\Gamma)P(\Gamma \rightarrow \Gamma') = W(\Gamma')P(\Gamma' \rightarrow \Gamma)$
  - 各态历经（Ergodicity）

SSE

Taylor expansion:

$$e^{-\beta H} = \sum_{n=0}^{\infty} \frac{(-\beta)^n}{n!} H^n$$

$$Z = \sum_{\alpha} \sum_{n=0}^{\infty} \frac{(-\beta)^n}{n!} \langle \alpha | H^{\mathrm{n}} | \alpha \rangle$$

$$Z = \sum_{n=0}^{\infty} \frac{(-\beta)^n}{n!} \sum_{\{\alpha\}_n} \langle \alpha_0 | H | \alpha_{n-1} \rangle \cdots \langle \alpha_2 | H | \alpha_1 \rangle \langle \alpha_1 | H | \alpha_0 \rangle$$

➢ 内能

$$E = \langle H \rangle = \frac{1}{Z} tr(e^{-\beta H})$$

$$E = \frac{1}{Z} \sum_{n=0}^{\infty} \frac{(-\beta)^n}{n!} \sum_{\{\alpha\}_{n+1}} \langle \alpha_0 | H | \alpha_n \rangle \cdots \langle \alpha_2 | H | \alpha_1 \rangle \langle \alpha_1 | H | \alpha_0 \rangle$$

$$E = -\frac{1}{Z} \sum_{n=1}^{\infty} \frac{(-\beta)^n}{n!} \frac{n}{\beta} \sum_{\{\alpha\}_n} \langle \alpha_0 | H | \alpha_{n-1} \rangle \cdots \langle \alpha_2 | H | \alpha_1 \rangle \langle \alpha_1 | H | \alpha_0 \rangle$$

$$E = -\frac{1}{Z} \sum_{n=0}^{\infty} \frac{(-\beta)^n}{n!} \frac{n}{\beta} \sum_{\{\alpha\}_n} \langle \alpha_0 | H | \alpha_{n-1} \rangle \cdots \langle \alpha_2 | H | \alpha_1 \rangle \langle \alpha_1 | H | \alpha_0 \rangle$$

$$\boxed{E = -\frac{\langle n \rangle}{\beta}}$$

$$\boxed{C_v = \langle n^2 \rangle - \langle n \rangle^2 - \langle n \rangle}$$

➢ **截断** $n \le M$ $M \propto \beta N$

$$Z=\sum_S \frac{(-\beta)^n (M-n)!}{M!} \sum_{\{\alpha\}_M} \sum_{\{S_i\}} \langle \alpha_0 | S_L | \alpha_{L-1} \rangle \cdots \langle \alpha_2 | S_2 | \alpha_1 \rangle \langle \alpha_1 | S_1 | \alpha_0 \rangle$$

$$\frac{n!}{C_M^n} = \frac{(M-n)!}{M!}$$

$$S_i \in \{H, I\}$$

- ## Spin-1/2 XXZ model

$$H = -J_{xy}\sum_{\langle i,j\rangle}(S_i^x S_j^x + S_i^y S_j^y) + J_z\sum_{\langle i,j\rangle} S_i^z S_j^z - h\sum_i S_i^z$$

$$S^{\pm} = (S^x \pm iS^y)$$

$$H = -\frac{J_{xy}}{2}\sum_{\langle i,j\rangle}(S_i^+ S_j^- + h.c.) + J_z\sum_{\langle i,j\rangle} S_i^z S_j^z - h\sum_i S_i^z$$

Holstein-Primakoff transformation

$$\mathrm{S}_i^+ \rightarrow a_i^\dagger, \qquad \mathrm{S}_i^- \rightarrow a_i, \qquad \mathrm{S}_i^z \rightarrow n_i - \frac{1}{2}$$

- ## Hard-core Boson-Hubbard model

$$H = -t\sum_{\langle i,j\rangle}(a_i^\dagger a_j + h.c.) + V\sum_{\langle i,j\rangle} n_i n_j - \mu\sum_i n_i$$

$$t = J_{xy}/2, \qquad \mathrm{V} = J_\mathrm{z}, \qquad \mu = J_\mathrm{z} + zV/2,$$

➢ Bond operator composition of Hamiltonian

$$H_{1,b} = C - \left[J_z S^z_{i(b)} S^z_{j(b)} - \frac{h}{z}\left(S^z_{i(b)} + S^z_{j(b)}\right)\right]$$

$$H_{2,b} = \frac{J_{xy}}{2}\left(S^+_{i(b)} S^-_{j(b)} + h.c.\right) \qquad H_{0,0} = I$$

$$H = -\sum_{a=1}^{2}\sum_{b=1}^{N_b} H_{a,b}$$

$$Z = \sum_{\alpha}\sum_{S_M} \frac{(\beta)^n (M-n)!}{M!} \prod_{p=1}^{M} \langle \alpha(p) | H_{a,b}(p) | \alpha(p-1) \rangle$$

$$= \sum_{\alpha}\sum_{S_M} W(\alpha, S_M)$$

位形： (α, S_M) 权重： $W(\alpha, S_M)$

概率密度： $p(\alpha, S_M) = \frac{W(\alpha, S_M)}{Z}$

➢ Bond operator composition of Hamiltonian

$$H_{1,b} = C - \left[V n_{i(b)} n_{j(b)} - \frac{\mu}{z} \left(n_{i(b)} + n_{j(b)} \right) \right]$$

$$H_{2,b} = t \left(a_{i(b)}^{\dagger} a_{j(b)} + h.c. \right) \qquad H_{0,0} = I$$

$$H = -\sum_{a=1}^{2} \sum_{b=1}^{N_b} H_{a,b}$$

$$Z = \sum_{\alpha} \sum_{S_M} \frac{(\beta)^n (M-n)!}{M!} \prod_{p=1}^{M} \langle \alpha(p) | H_{a,b}(p) | \alpha(p-1) \rangle$$

$$= \sum_{\alpha} \sum_{S_M} W(\alpha, S_M)$$

位形： (α, S_M) 权重： $W(\alpha, S_M)$

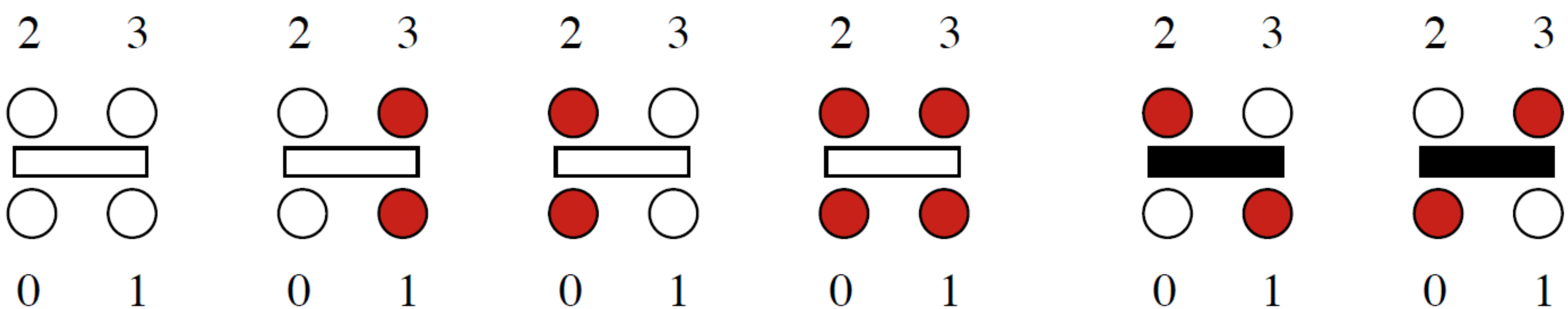
概率密度： $p(\alpha, S_M) = \frac{W(\alpha, S_M)}{Z}$

➢ Bond operator composition of Hamiltonian

$$W(\alpha, S_M) = \frac{(\beta)^n (M-n)!}{M!} \prod_{p=1}^{M} \langle \alpha(p) | H_{a,b}(p) | \alpha(p-1) \rangle$$

- 转移矩阵元(bond operator)（玻色子粒子数表象）

$$\langle \alpha(p) | H_{a,b}(p) | \alpha(p-1) \rangle = \langle n_i(p), n_j(p) | H_{a,b}(p) | n_i(p-1), n_j(p-1) \rangle$$



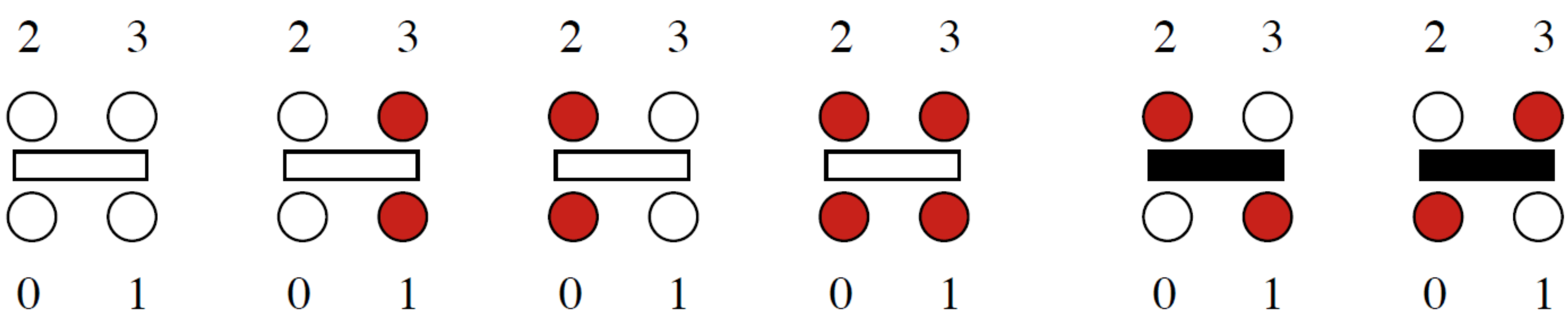
- 顶角(vortex: Ξ), 端口/腿(leg: l =0,1,2,3)

➢ Bond operator composition of Hamiltonian

$$W(\alpha, S_M) = \frac{(\beta)^n (M-n)!}{M!} \prod_{p=1}^{M} \langle \alpha(p) | H_{a,b}(p) | \alpha(p-1) \rangle$$

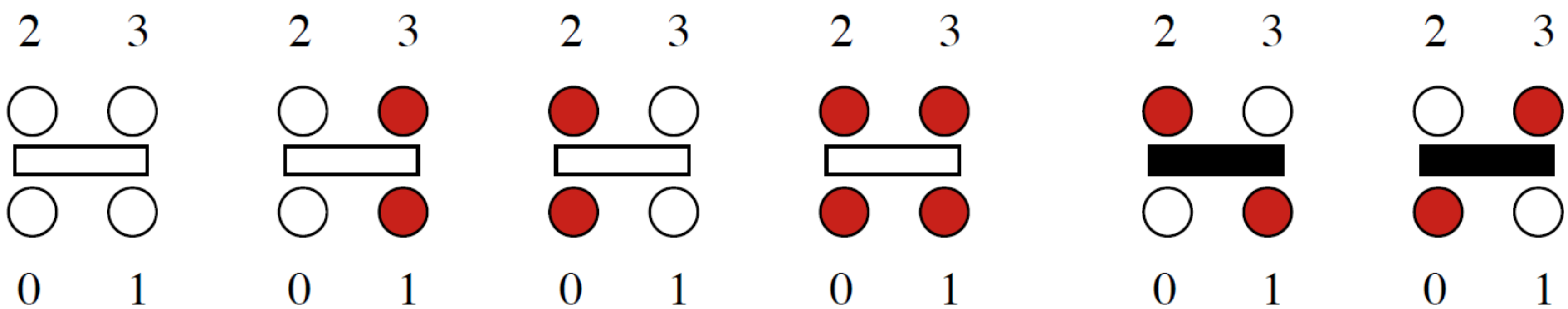
- 转移矩阵元(bond operator)（自旋S^z表象）

$$\langle \alpha(p) | H_{a,b}(p) | \alpha(p-1) \rangle = \langle S_i^z(p), S_j^z(p) | H_{a,b}(p) | S_i^z(p-1), S_j^z(p-1) \rangle$$



□ 顶角(vortex: Ξ), 端口/腿(leg: l=0,1,2,3)

顶角权重: $W(\Xi)$ 硬核玻色子表象



$$\langle 0,0|H_{1,b}|0,0\rangle = C$$

$$\langle 0,1|H_{1,b}|0,1\rangle = C + \mu/z$$

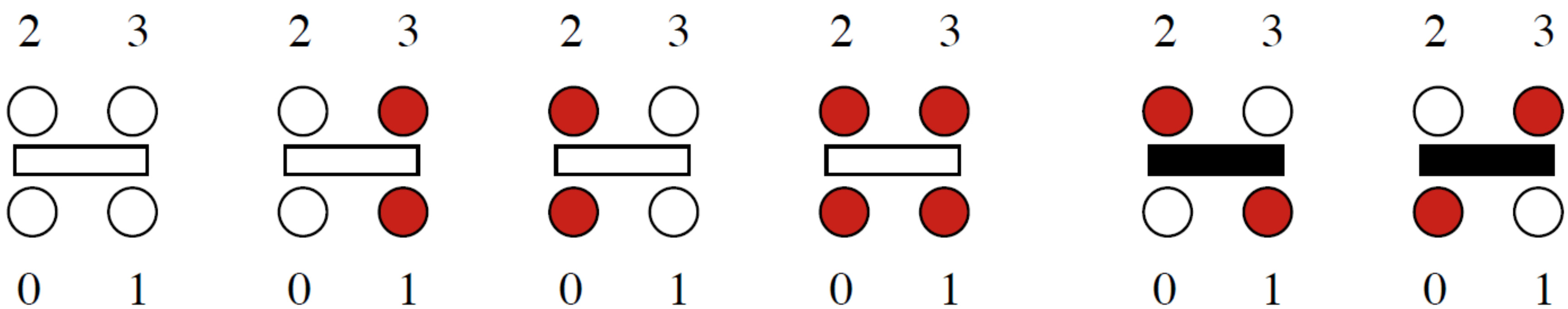
$$\langle 1,0|H_{1,b}|1,0\rangle = C + \mu/z$$

$$\langle 1,1|H_{1,b}|1,1\rangle = C - V + 2\mu/z$$

$$\langle 1,0|H_{2,b}|0,1\rangle = t$$

$$\langle 0,1|H_{2,b}|1,0\rangle = t$$

➢ **顶角权重:** $W(\Xi)$ 自旋表象



$$\langle\downarrow,\downarrow|H_{1,b}|\downarrow,\downarrow\rangle = C - \frac{J_z}{4} + \frac{h}{z} \qquad \langle\downarrow,\uparrow|H_{1,b}|\downarrow,\uparrow\rangle = C + \frac{J_z}{4}$$

$$\langle\uparrow,\downarrow|H_{1,b}|\uparrow,\downarrow\rangle = C + \frac{J_z}{4} \qquad \langle\uparrow,\uparrow|H_{1,b}|\uparrow,\uparrow\rangle = C - \frac{J_z}{4} - \frac{h}{z}$$

$$\langle\uparrow,\downarrow|H_{2,b}|\downarrow,\uparrow\rangle = \frac{J_{xy}}{2} \qquad \langle\downarrow,\uparrow|H_{2,b}|\uparrow,\downarrow\rangle = \frac{J_{xy}}{2}$$

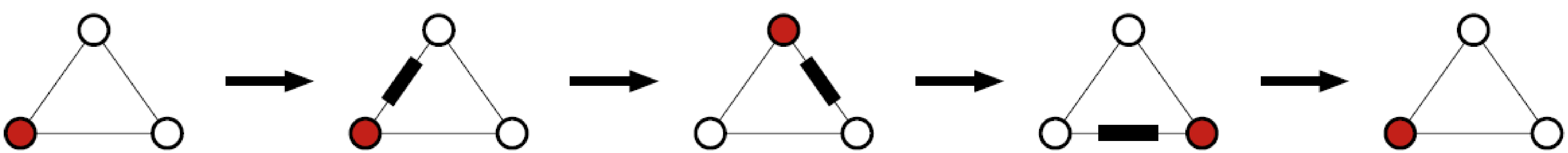
➢ 符号问题

$$W(\alpha, S_M) = \frac{(\beta)^n (M-n)!}{M!} \cdot \prod_{p=1}^{M} \langle \alpha(p) | H_{a,b}(p) | \alpha(p-1) \rangle$$

Sign problem： $\exists\, \boldsymbol{W(\alpha, S_M) < 0}$

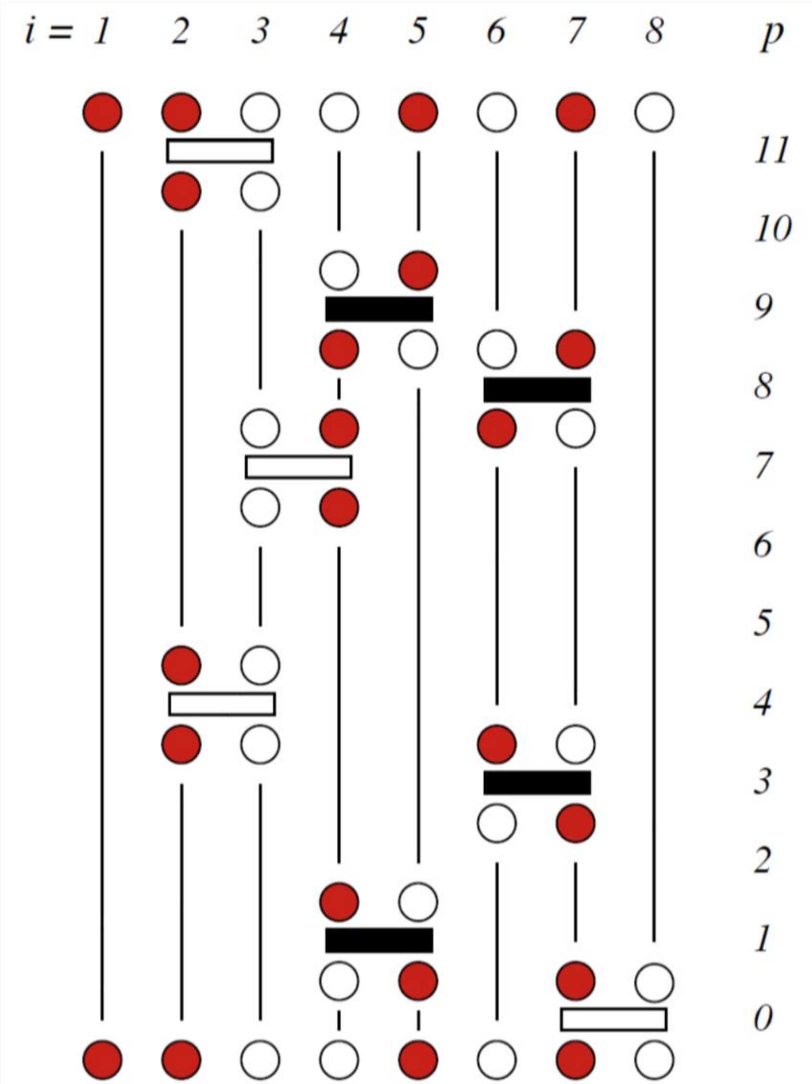
- 比如三角晶格上的反铁磁Heisenberg $J_{xy} < 0$,

$$\langle \uparrow, \downarrow | H_{2,b} | \downarrow, \uparrow \rangle = \frac{J_{xy}}{2} < 0 \qquad \langle \downarrow, \uparrow | H_{2,b} | \uparrow, \downarrow \rangle = \frac{J_{xy}}{2} < 0$$



$$\boldsymbol{W(\alpha, S_M)} = \left(\frac{J_{xy}}{2}\right)^3 < 0$$

➢ 位形的图形表示：

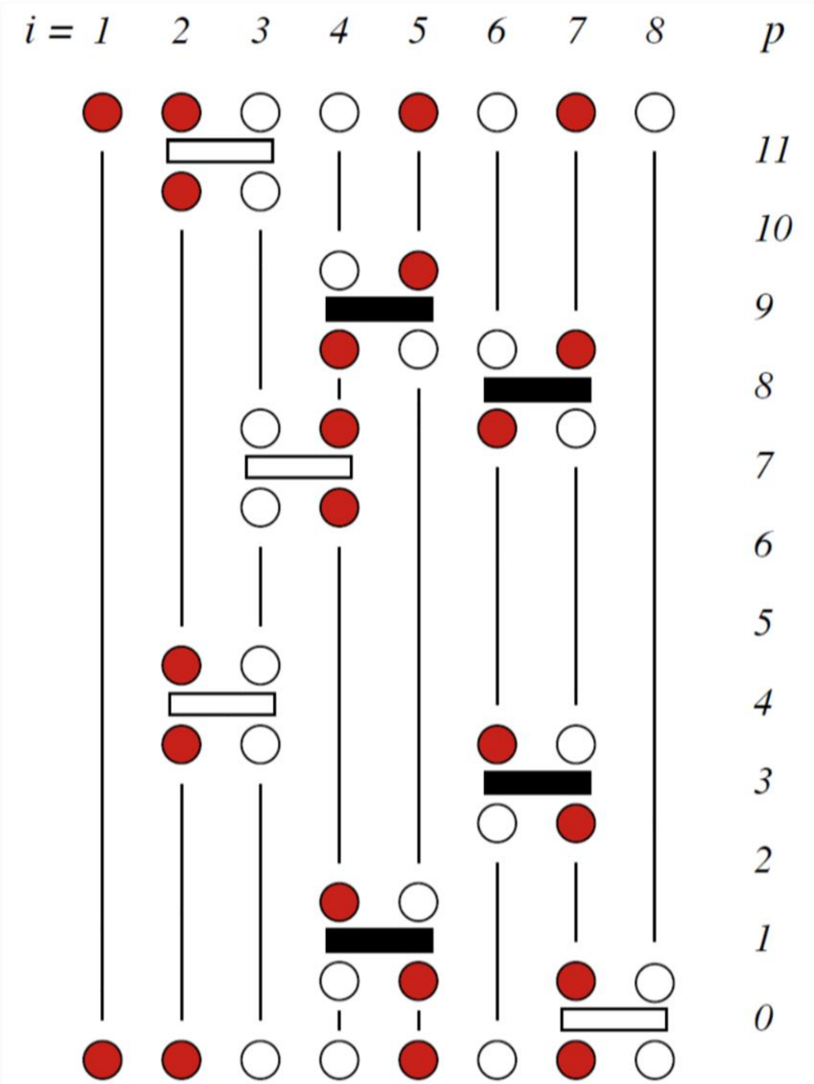


$$W(\alpha, S_M) = \frac{(\beta)^n (M-n)!}{M!}$$

$$\cdot \prod_{p=1}^{M} \langle \alpha(p) | H_{a,b}(p) | \alpha(p-1) \rangle$$

- 零时刻空间态 $\alpha(0)$
- 算符序列 $s(p)$

➢ 位形存储：零时刻空间态 $\alpha(0)$ + 算符序列 $s(p)$ ★★



- 零时刻空间态 $\alpha(0)$

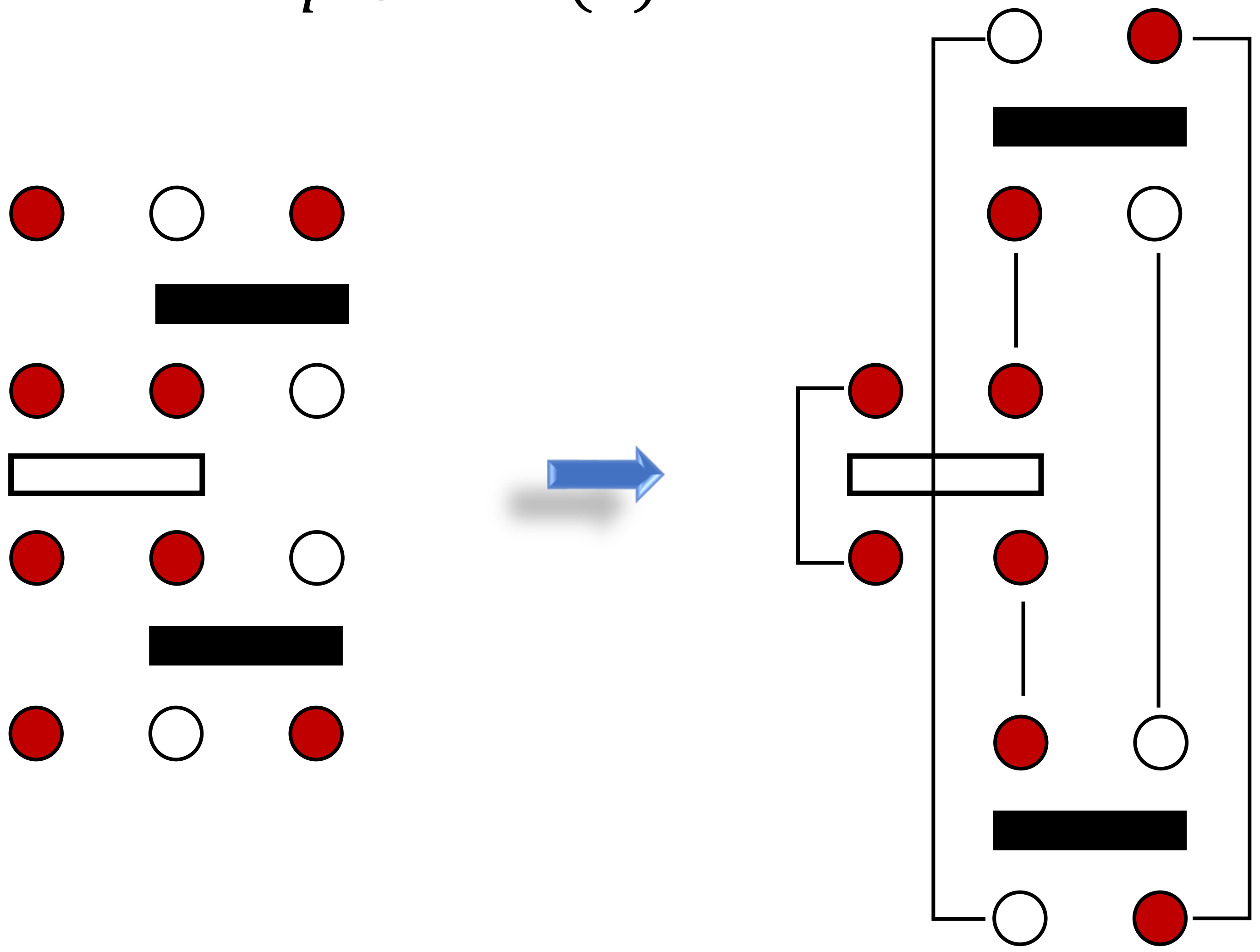
$$\alpha(0) = \{S_i^z\}$$

- 算符序列 $s(p)$

$$s(p) = \begin{cases} 0, \text{ p 时刻是单位算符} \\ 2b(p) + a(p) - 1 \end{cases}$$

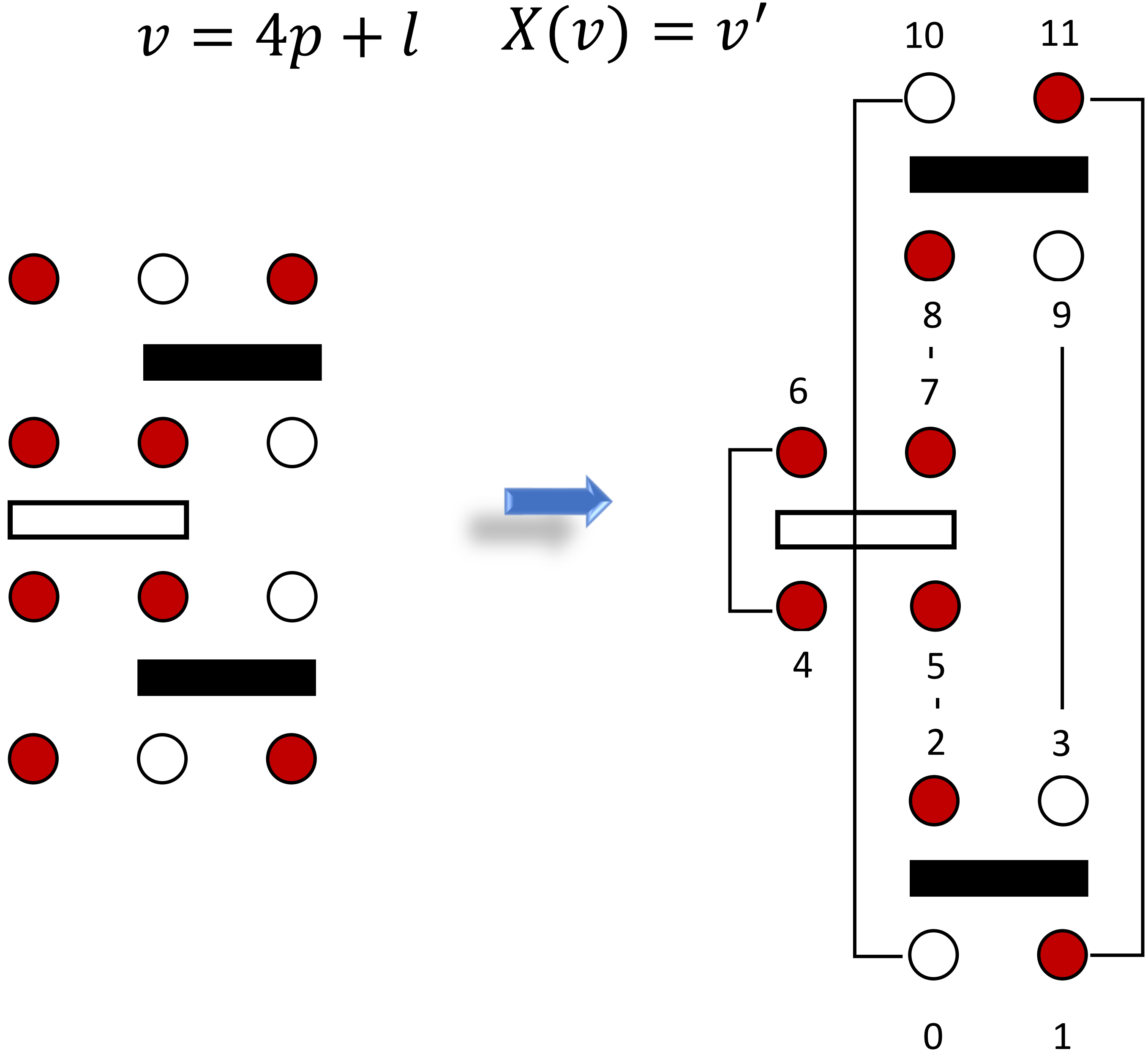
位形存储：端口编号　　链表

$$v = 4p + l \quad X(v) = v'$$



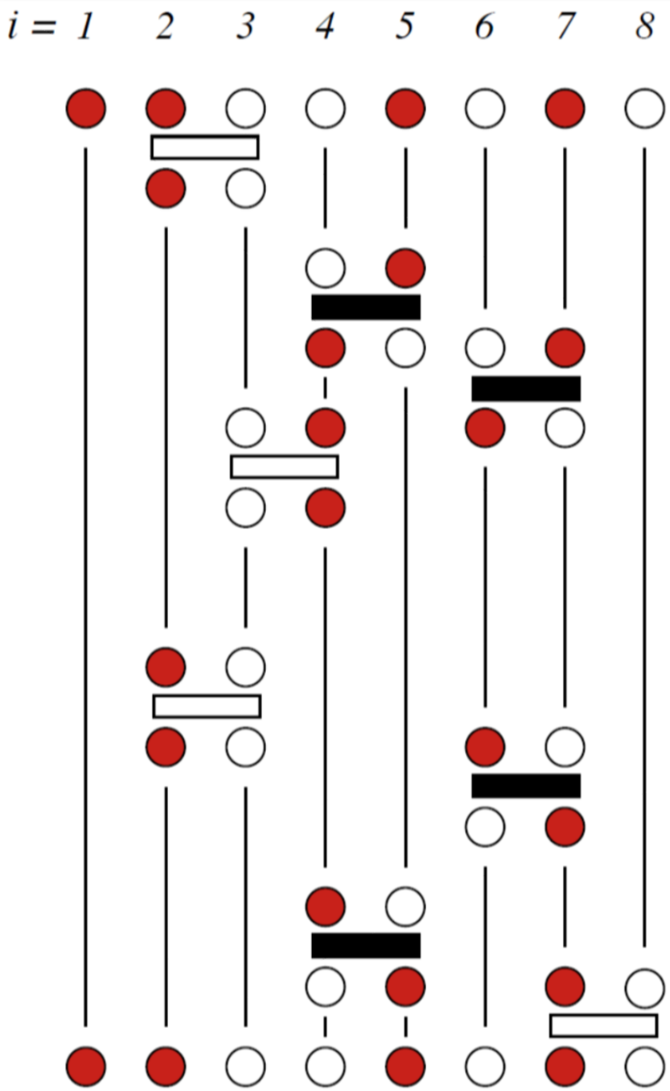
➢ 位形存储：端口编号　　链表

$$v = 4p + l \qquad X(v) = v'$$



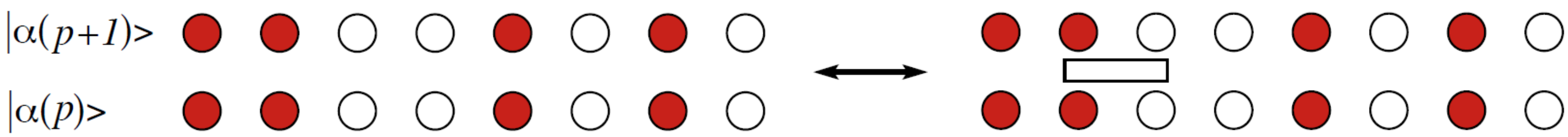
➢ 位形存储：端口编号 链表 ★★

$$v = 4p + l \qquad X(v) = v'$$



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	

➢ 位形演化：对角更新（Diagonal update）



- **Detail balance:**

$$W(\alpha,S_M)P(\{\alpha,S_M\}\rightarrow\{\alpha,S'_M\})=W(\alpha,S'_M)P(\{\alpha,S'_M\}\rightarrow\{\alpha,S_M\})$$

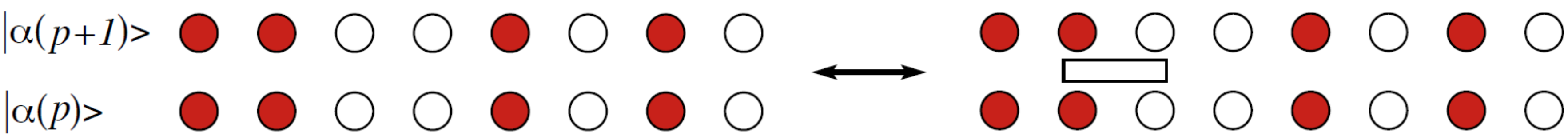
$$W(\alpha,S_M)=\frac{(\beta)^n(M-n)!}{M!}\prod_{p=1}^{M}\langle\alpha(p)|H_{a,b}(p)|\alpha(p-1)\rangle$$

- **Metropolis acceptance probability:**

$$P_{accept}([0,0]\rightarrow[1,b])=min\left[\frac{\beta\langle\alpha\,(p+1)|H_{1,b}|\alpha\,(p)\rangle}{M-n},1\right]$$

$$P_{accept}([1,b]\rightarrow[0,0])=min\left[\frac{M-n+1}{\beta\langle\alpha\,(p+1)|H_{1,b}|\alpha\,(p)\rangle},1\right]$$

➢ 位形演化：对角更新（Diagonal update） ★★



- **Detail balance:**

$$W(\alpha,S_M)P(\{\alpha,S_M\}\to\{\alpha,S'_M\}) = W(\alpha,S'_M)P(\{\alpha,S'_M\}\to\{\alpha,S_M\})$$

$$P(\{\alpha,S_M\}\to\{\alpha,S'_M\}) = P_{select}\cdot P_{accept}$$

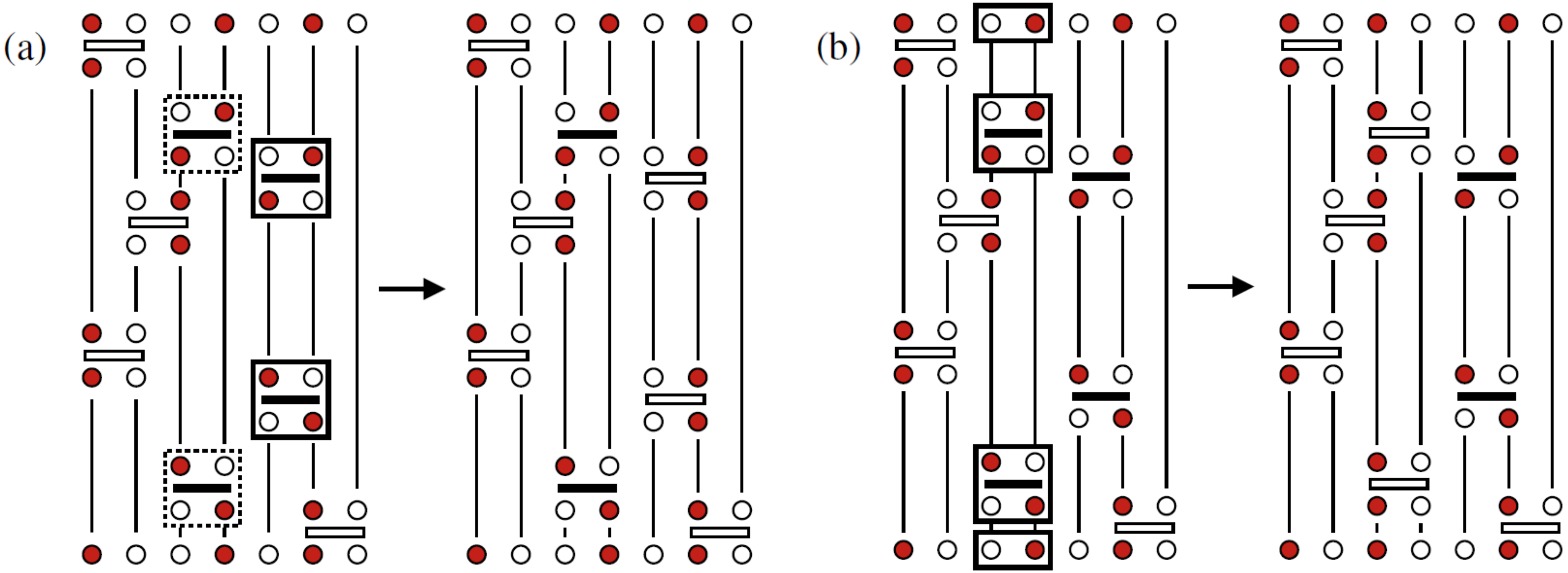
$$\begin{cases} P_{select}([0,0]\to[1,b]) = \dfrac{1}{N_b} \\ P_{select}([1,b]\to[0,0]) = 1 \end{cases}$$

- **Metropolis acceptance probability:**

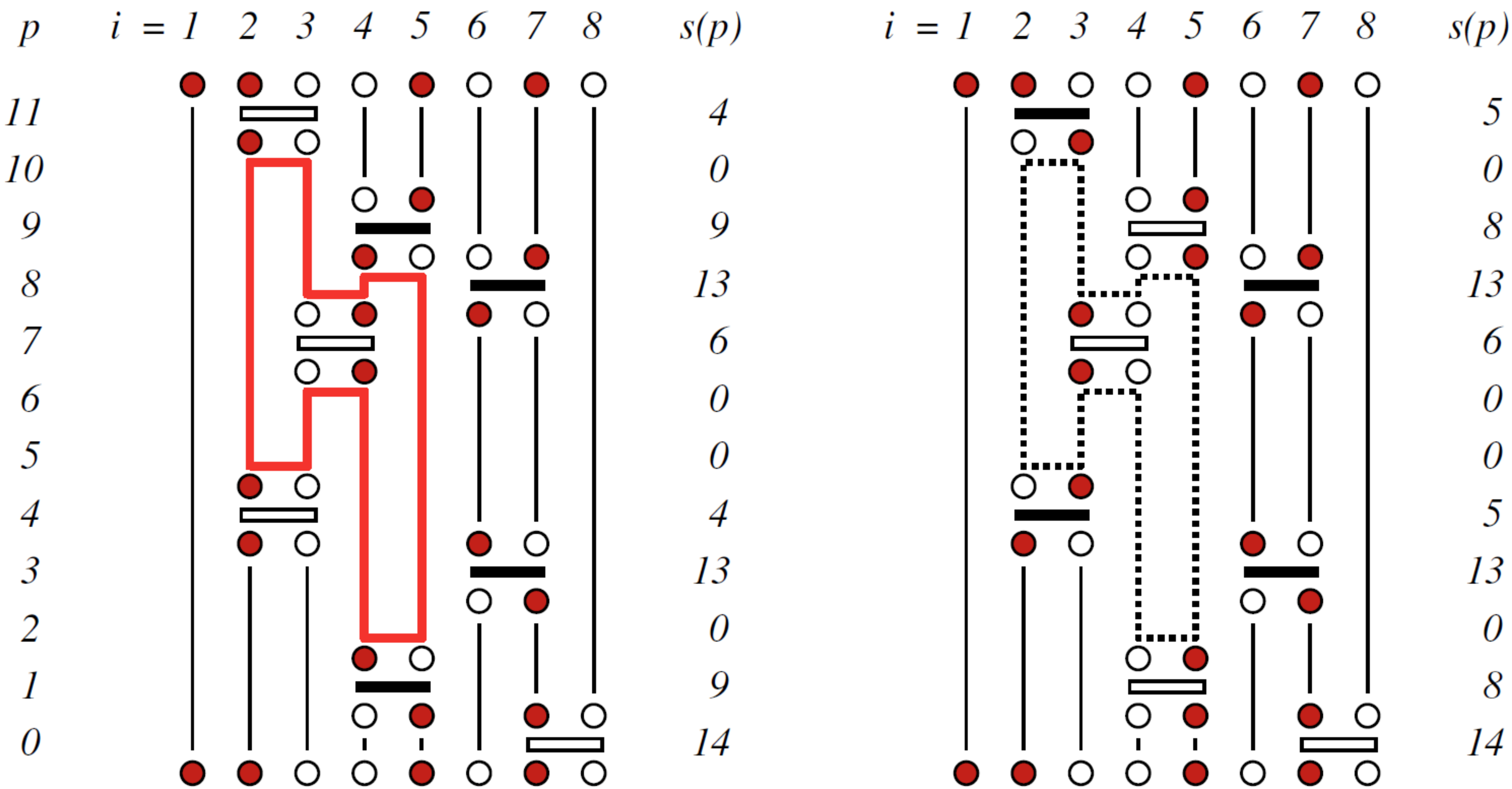
$$P_{accept}([0,0]\to[1,b]) = min\left[\frac{\beta N_b\langle\alpha\,(p+1)|H_{1,b}|\alpha\,(p)\rangle}{M-n},1\right]$$

$$P_{accept}([1,b]\to[0,0]) = min\left[\frac{M-n+1}{\beta N_b\langle\alpha\,(p+1)|H_{1,b}|\alpha\,(p)\rangle},1\right]$$

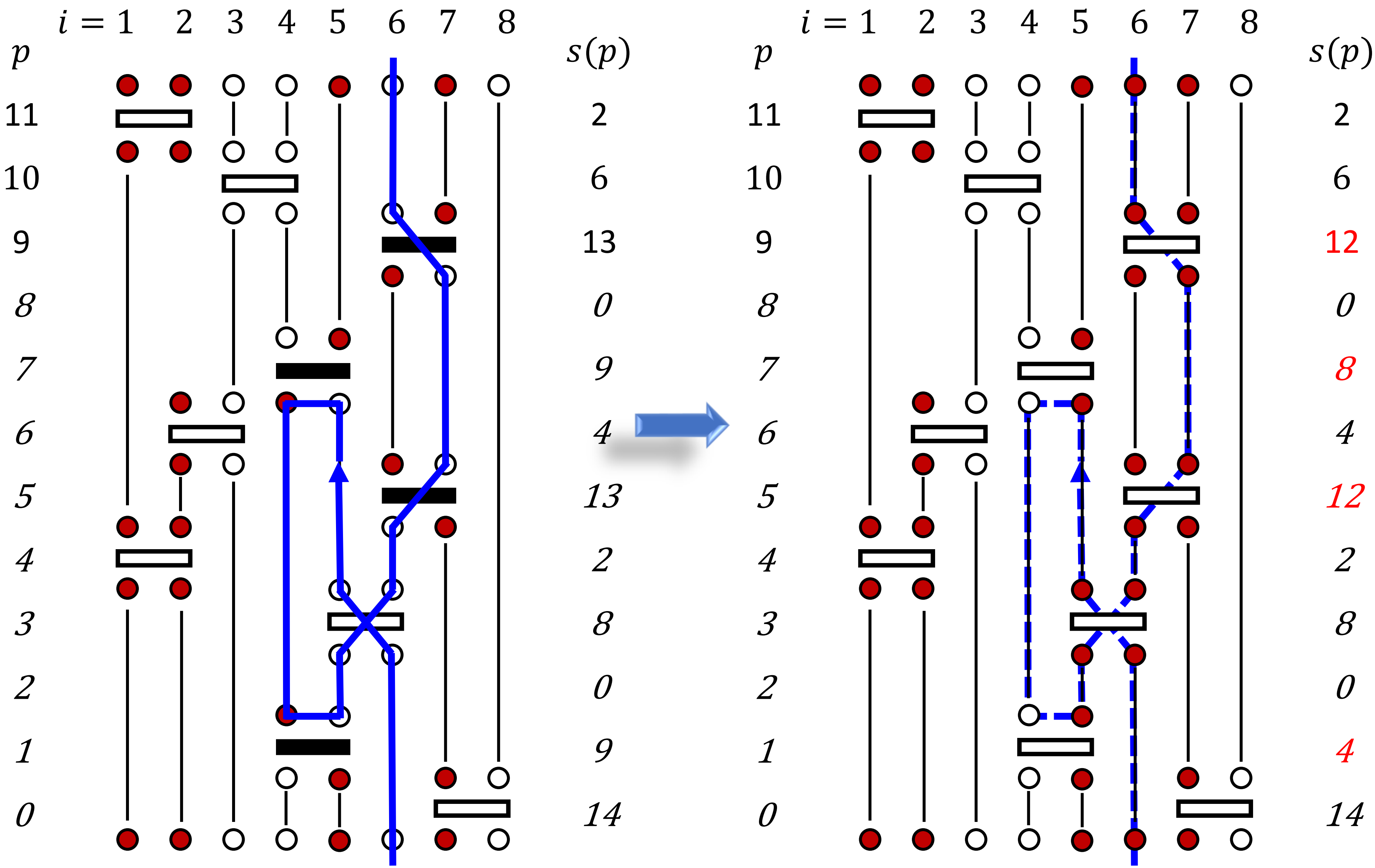
➢ 位形演化：非对角更新（Off-diagonal update）



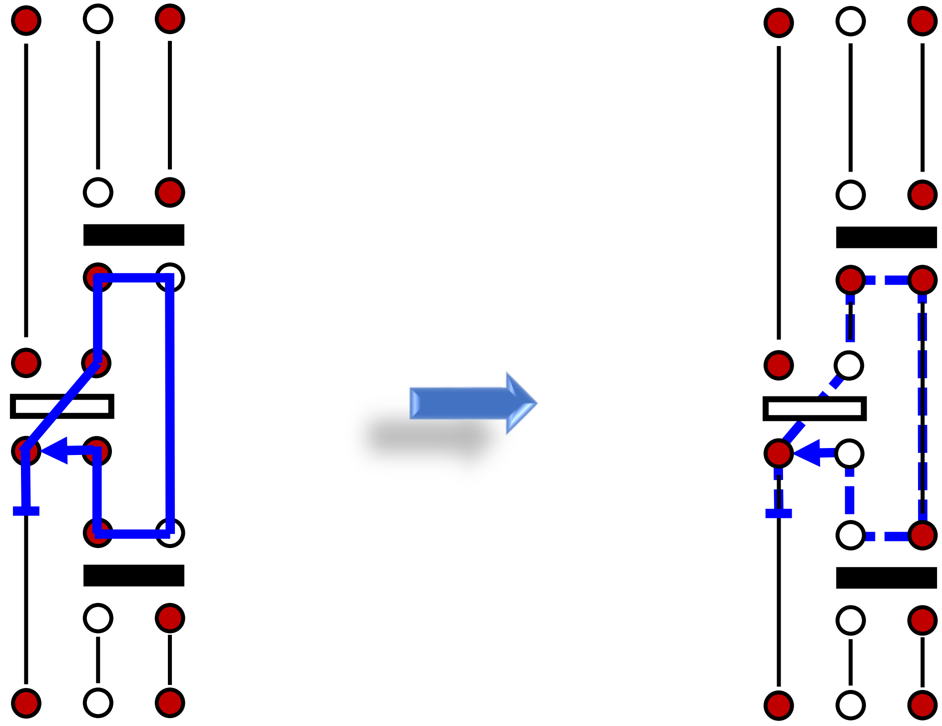
➢ 位形演化：圈更新（Operator-loop update）



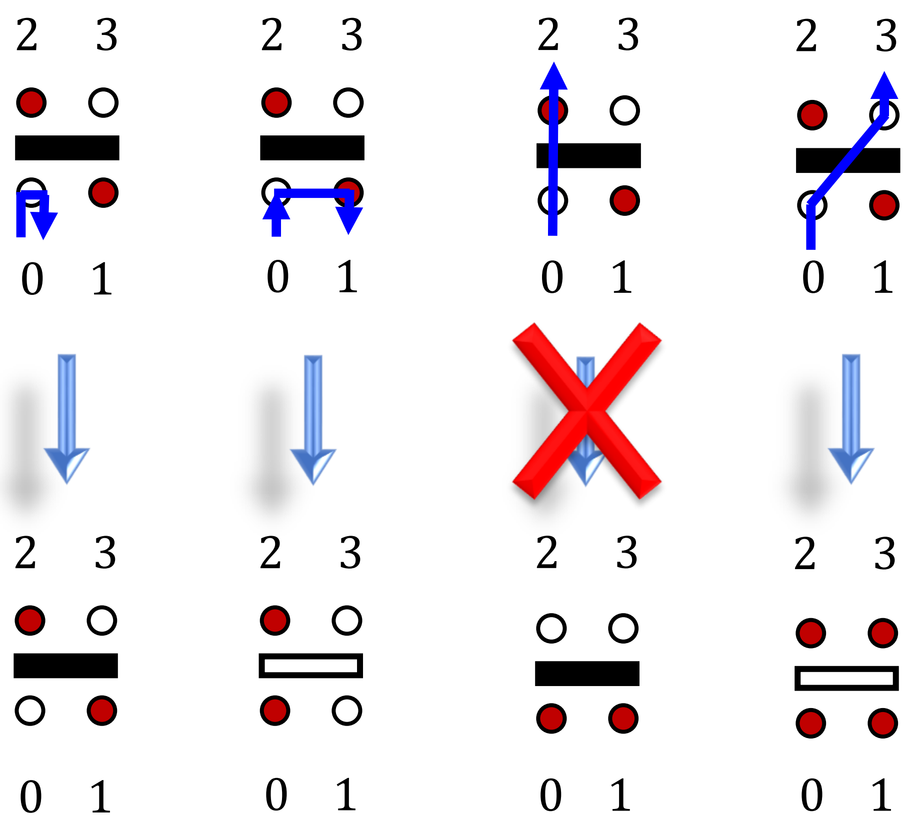
➢ 位形演化：有向圈更新（directed loop update）



➢ 位形演化：有反弹的圈



➢ 位形演化：有向圈（directed loop）



➢ 位形演化：局域细致平衡

- 细致平衡（Detail balance）

$$W(\Gamma)P(\Gamma\rightarrow\Gamma')=W(\Gamma')P(\Gamma'\rightarrow\Gamma)$$

- 局域（顶角）细致平衡

$$W(\Xi)P(\Xi,l\rightarrow l')=W(\Xi')P(\Xi',l'\rightarrow l)$$

$$\sum_{l'=1}^{4}P(\Xi,l\rightarrow l')=1$$

□ 引入端口权重：$a(\Xi\rightarrow\Xi',l\rightarrow l')$

$$P(\Xi,l\rightarrow l')=\frac{a(\Xi\rightarrow\Xi',l\rightarrow l')}{W(\Xi)}$$

➢ 位形演化：有向圈方程（Equs. of directed loop）

★★

$$\sum_{l'=0}^{3} a(\Xi \to \Xi', l \to l') = W(\Xi)$$

$$a(\Xi \to \Xi', l \to l') = a(\Xi' \to \Xi, l' \to l)$$

- $a(\Xi \to \Xi', l \to l')$ 简记为：a_{ij} （i: 入口， j:出口）；

  $W(\Xi)$ 简记为：W_i ； $W(\Xi')$ 简记为：W_j；

  $P(\Xi, l \to l')$ 简记为：P_{ij} 。

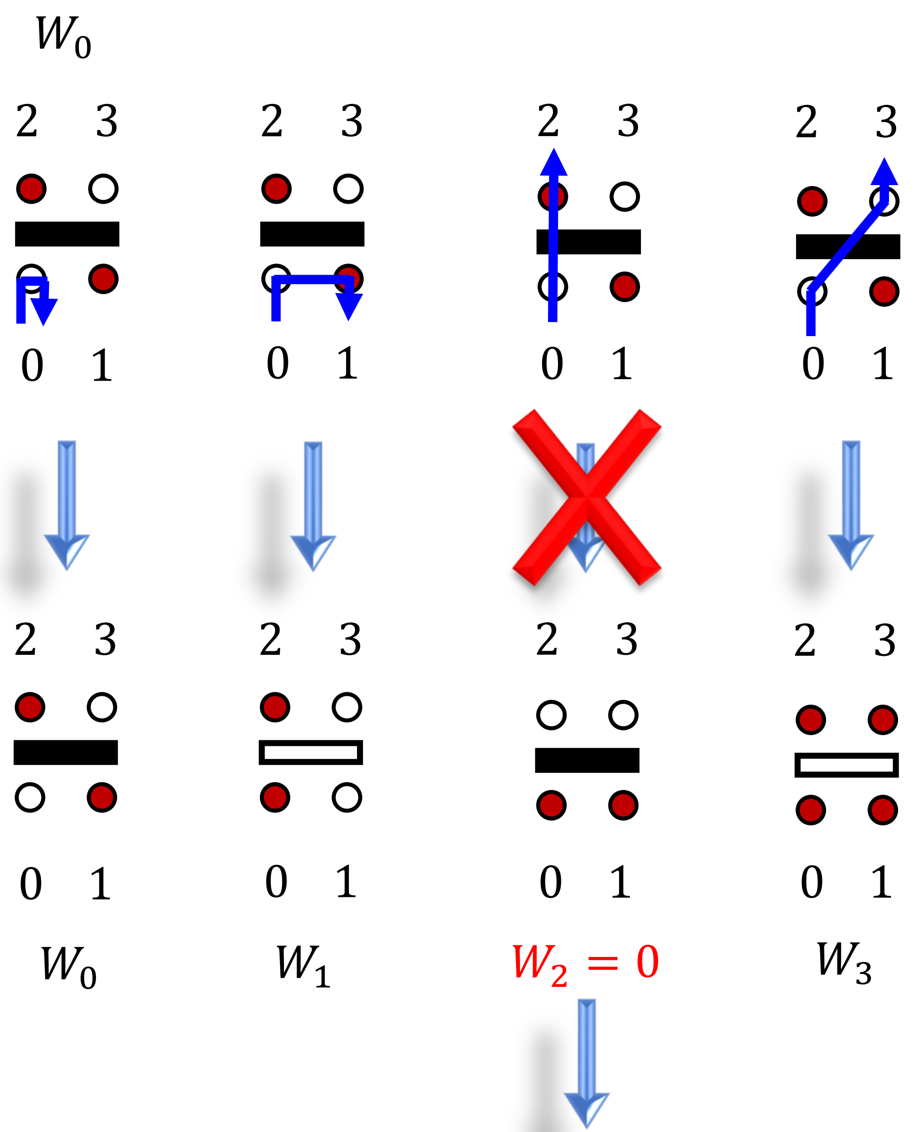
$$\sum_{j=0}^{3} a_{ij} = W_i \,; \qquad a_{ij} = a_{ji}$$

$$P_{ij} = \frac{a_{ij}}{W_i}$$

➢ 位形演化：有向圈方程组（Equs. of directed loop）

$$
\begin{cases}
a_{00}+a_{01}+a_{02}+a_{03}=W_0 \\
a_{10}+a_{11}+a_{12}+a_{13}=W_1 \\
a_{20}+a_{21}+a_{22}+a_{23}=W_2 \\
a_{30}+a_{31}+a_{32}+a_{33}=W_3
\end{cases}
$$

➢ 位形演化：有向圈（directed loop）



$$a_{20} = a_{21} = a_{22} = a_{23} = W_2 = 0$$

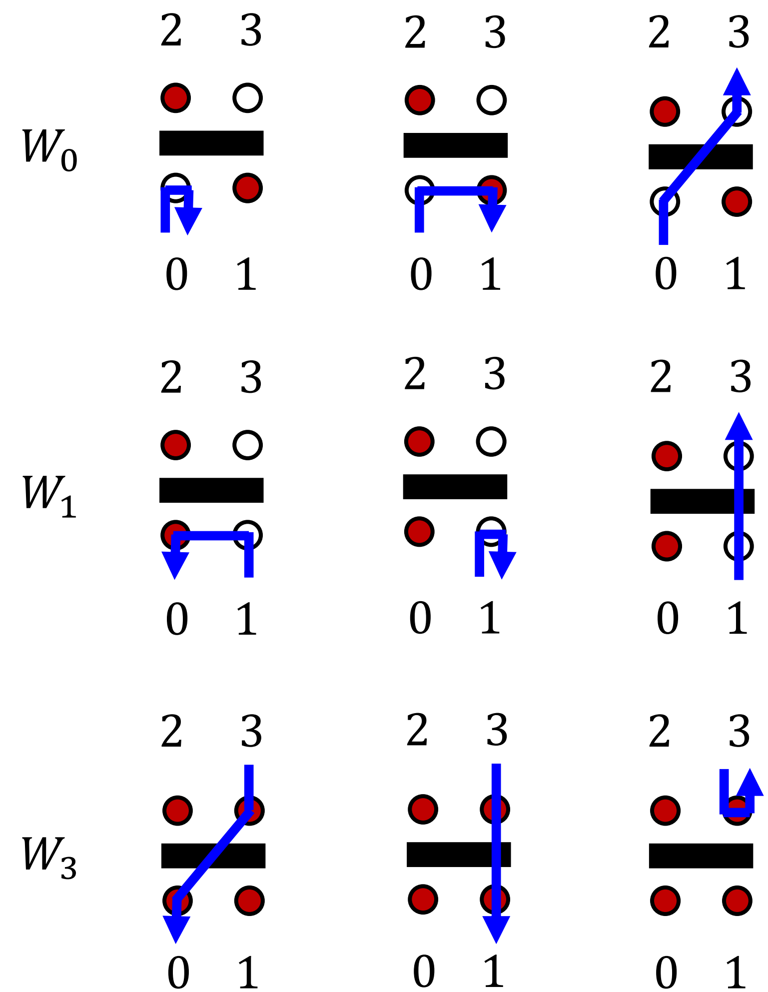
$$a_{02} = a_{12} = a_{32} = 0$$

➢ 位形演化：有向圈方程组（Equs. of directed loop)

$$
\begin{cases}
a_{00} + a_{01} + a_{02} + a_{03} = W_0 \\
a_{10} + a_{11} + a_{12} + a_{13} = W_1 \\
a_{20} + a_{21} + a_{22} + a_{23} = W_2 \\
a_{30} + a_{31} + a_{32} + a_{33} = W_3
\end{cases}
$$

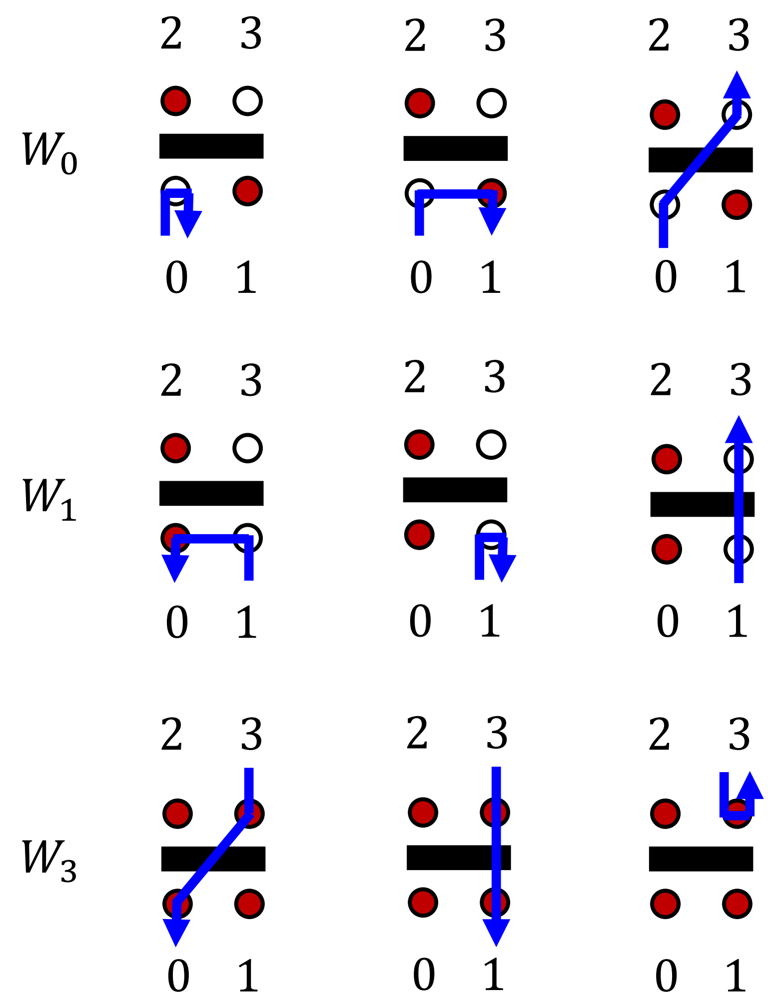
$$a_{ij} = a_{ji}$$

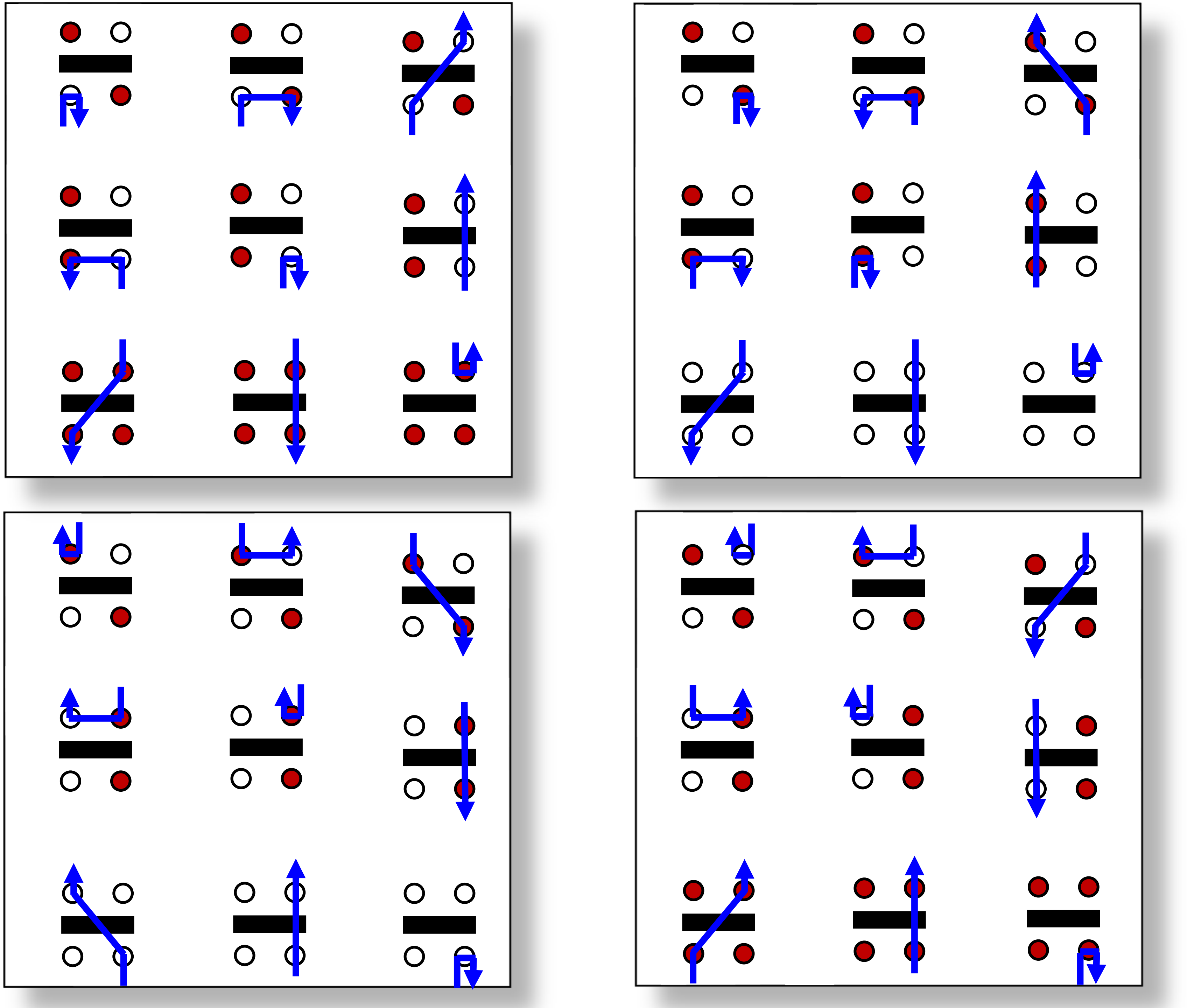
$$
\begin{cases}
a_{00} + a_{01} + a_{03} = W_0 \\
a_{01} + a_{11} + a_{13} = W_1 \\
a_{03} + a_{13} + a_{33} = W_3
\end{cases}
$$



➢ 位形演化：有向圈方程组的解

$$
\begin{cases}
a_{00} + a_{01} + a_{03} = W_0 = \dfrac{J_{xy}}{2} \\[2ex]
a_{01} + a_{11} + a_{13} = W_1 = C + \dfrac{J_z}{4} \\[2ex]
a_{03} + a_{13} + a_{33} = W_3 = C - \dfrac{J_z}{4} - \dfrac{h}{z}
\end{cases}
$$





➢ 位形演化：有向圈方程组的解

$$\begin{cases} b_1 + a + b = \dfrac{J_{xy}}{2} \\ a + b_2 + \mathrm{c} = C + \dfrac{J_z}{4} \\ b + c + b_3 = C - \dfrac{J_z}{4} - \dfrac{h}{z} \end{cases}$$

$$\begin{cases} b_1' + a' + b' = \dfrac{J_{xy}}{2} \\ a' + b_2' + c' = C + \dfrac{J_z}{4} \\ b' + c' + b_3' = C - \dfrac{J_z}{4} + \dfrac{h}{z} \end{cases}$$

- **最小化反弹**：

Bounce $\{b_i\}$ 尽可能小，最好全为零。

$$J_{xy} = 1$$

$$C = C_0 + \epsilon$$

$$C_0 = \frac{J_z}{4} + \frac{h}{z}$$

➢ 位形演化：有向圈方程组的解

$$\begin{cases} b_1 + a + b = \dfrac{1}{2} \\ a + b_2 + \mathrm{c} = \dfrac{J_z}{2} + \dfrac{h}{z} + \epsilon \\ b + c + b_3 = \epsilon \end{cases}$$

$$\begin{cases} b_1' + a' + b' = \dfrac{1}{2} \\ a' + b_2' + c' = \dfrac{J_z}{2} + \dfrac{h}{z} + \epsilon \\ b' + c' + b_3' = \dfrac{2h}{z} + \epsilon \end{cases}$$

- **最小化反弹**：

  Bounce $\{b_i\}$ 尽可能小，最好全为零。

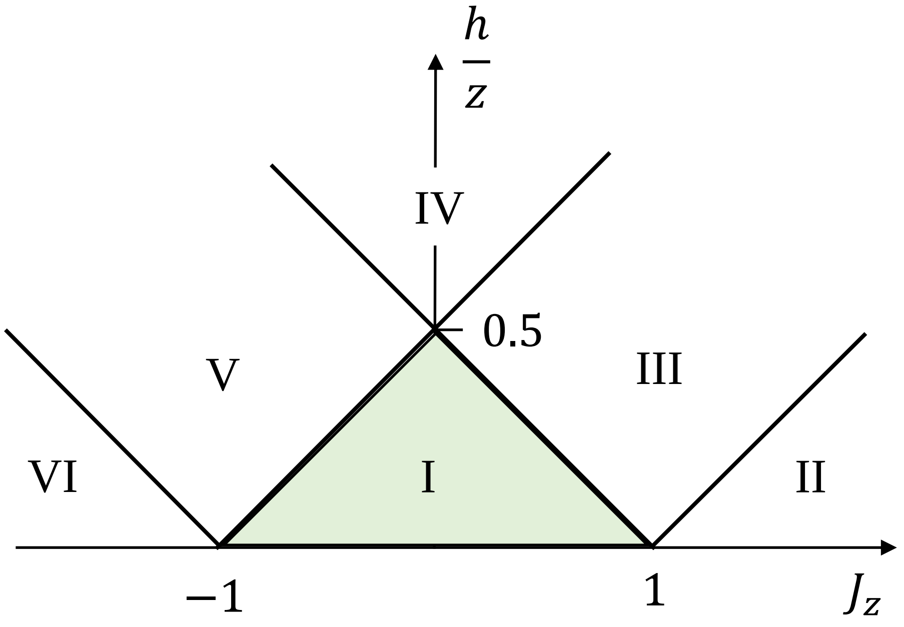
➢ 位形演化：有向圈方程组的解

$$\begin{cases} a = \dfrac{1+J_z}{4} + \dfrac{h}{2z} + \dfrac{-b_1 - b_2 + b_3}{2} \\ b = \dfrac{1-J_z}{4} - \dfrac{h}{2z} + \dfrac{-b_1 + b_2 - b_3}{2} \\ c = \dfrac{J_z - 1}{4} + \dfrac{h}{2z} + \varepsilon + \dfrac{b_1 - b_2 - b_3}{2} \end{cases}$$

$$\begin{cases} a' = \dfrac{1+J_z}{4} - \dfrac{h}{2z} + \dfrac{-b_1' - b_2' + b_3'}{2} \\ b' = \dfrac{1-J_z}{4} + \dfrac{h}{2z} + \dfrac{-b_1' + b_2' - b_3'}{2} \\ c' = \dfrac{J_z - 1}{4} + \dfrac{3h}{2z} + \varepsilon + \dfrac{b_1' - b_2' - b_3'}{2} \end{cases}$$

- **最小化反弹**：

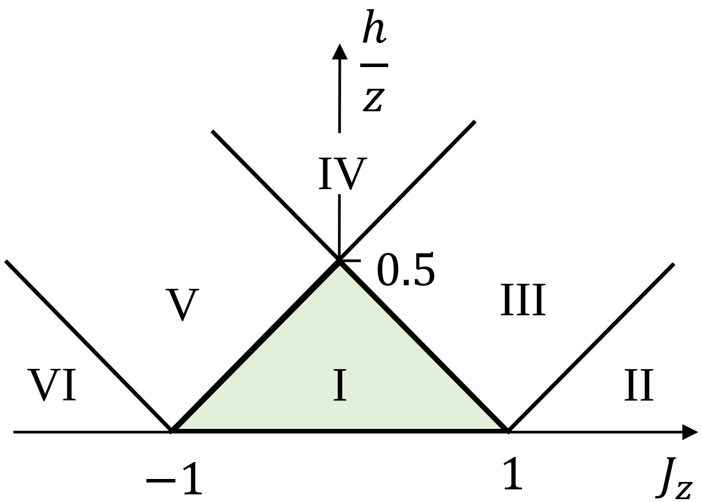
Bounce $\{b_i\}$ 尽可能小，最好全为零。



➢ 位形演化：有向圈方程组的解

★★

$$J_z^{\pm} = (1 \pm J_z)/2$$



	Bounce weights		ϵ_{min}
I			$\frac{J_z^-}{2} - \frac{h}{2z}$
II	$b_2 = \frac{h}{z} - J_z^-$	$b_2' = -\frac{h}{z} - J_z^-$	0
III	$b_2 = \frac{h}{z} - J_z^-$		0
IV	$b_2 = \frac{h}{z} - J_z^-$	$b_3' = -\frac{h}{z} - J_z^+$	0
V		$b_3' = -\frac{h}{z} - J_z^+$	$\frac{J_z^-}{2} - \frac{h}{2z}$
VI	$b_3 = -\frac{h}{z} - J_z^+$	$b_3' = -\frac{h}{z} - J_z^+$	$-\frac{h}{z} - \frac{J_z}{2}$

➢ 位形演化：建立概率表 ★★★

$$\{a_{ij},\ W_i\} \Rightarrow P_{ij} = \frac{a_{ij}}{W_i}$$

$$\Downarrow$$

$$P(\Xi, l \to l') = \frac{a(\Xi \to \Xi', l \to l')}{W(\Xi)}$$

$$\Downarrow$$

$$P(\Xi = 1{:}6, l = 1{:}4, l' = 1{:}4)$$

算法的程序实现

一、定义及初始化变量或数组:
晶格结构、初始化自旋位形、
Vortex权重、概率表

↓

二、位形演化:
Diagonal update
Directed-loop update

↓

三、抽样测量:
能量、比热、磁化强度等

↓

四、结果输出:
可采用bin输出，方便误差估计

主程序(main.f90)

```fortran
!==========================================================!!!
! directed loop SSE fot XXZ model                         !!!
! Written by  Yan-Cheng Wang @BUAA     2024-06-09~06-12  !!!
!==========================================================!!!
include "variable.f90"
program Square_XXZ
USE variable
IMPLICIT NONE
DOUBLE PRECISION :: aig,anh,ra
INTEGER :: itoss,isample,ibin

open(1,file='sysize.in')
read(1,*), nx,ny
close(1)

open(2,file='evol.in')
read(2,*), ntoss,nsample,nbin
read(2,*), iseed
close(2)

open(3,file='para.in')
read(3,*), Jz,hh
read(3,*), beta
close(3)

call allocate_para

call ransi

call lattice

call weight

call initconf

 nloop=20
 anh=0.d0
 aig=0.d0
!----------------------------------------------------------------
```

```fortran
!!!!!!: themolization
ibin=0
do itoss=1,ntoss
   call dupdate
   call lupdate
   anh=anh+nh
   aig=aig+ig

   ra=aig/anh/2.d0
   if (ra.lt.0.8) then
      nloop=nloop*5/3
   elseif (ra.gt.1.5) then
      nloop=nloop*3/4
      if(nloop.lt.1)nloop=1
   endif

   call adjlgth

   if (mod(itoss,ntoss/10).eq.0) then
      call openlog
      write(12,10)itoss,l,nh,ra
10    format('Equilibrated. ',I7,'   L = : ',I8,'   nh= : ',I8, &
             '    ra=:', f8.3)
      call closelog
!---------------------------------------------------------
   endif
enddo
!---------------------------------------------------------
!!   sample
        do ibin=1,nbin
          call cleardata
          do isample=1,nsample
           call dupdate
           call lupdate()
           call measure()
          if (mod(isample,nsample/10).eq.0) then
            call openlog
            write(12,11) ibin,l,nh
```

```
!--------------------------------------------------------
!!    sample
        do ibin=1,nbin
          call cleardata
          do isample=1,nsample
           call dupdate
           call lupdate()
           call measure()
          if (mod(isample,nsample/10).eq.0) then
            call openlog
            write(12,11) ibin,l,nh
11      format('Measured ',I7,'   L= ',I8,'   nh= ',I8)
            call closelog
          endif
        enddo
        call results(ibin)
        enddo
!---------------------------------------------------
!----------------------------------------------------------------------------
      contains
      include "allocate.f90"
      include "initial.f90"
      include "lattice.f90"
      include "weight.f90"
      include "update.f90"
      include "cleardata.f90"
      include "measure.f90"
      include "results.f90"
      include "random.f90"
      end program ! Square_XXZ
                                                              96,1
```

一、定义及初始化变量或数组

- 定义变量和数据(variable.f90)

```
MODULE variable
!Lattice structure and parmeters of Hamitonian
IMPLICIT NONE; SAVE
INTEGER :: nx,ny                    ! system sizes
INTEGER  :: Nsite                   ! total sites
INTEGER  :: Nb                      ! total bonds
INTEGER, allocatable :: kbst(:,:)   ! lattice def.
REAL(8) :: beta,Jz,hh
INTEGER, allocatable :: spin(:)     ! initial state of the spin-config

!MC management: probability tabel  of vortex
INTEGER :: lmax                    ! max of time seq.
INTEGER :: l                       ! number of time seq.
INTEGER :: nh                      ! number of vortex
INTEGER, allocatable :: ns(:,:,:,:)! legs-spin-config
INTEGER, allocatable :: ktype(:)   ! property of the vortex:0->diagonal, 1->off-diagonal
INTEGER, allocatable :: nv(:,:)    ! tell the state of every leg of each vortex
REAL(8), allocatable :: w(:)      ! give the weight of each vortex
REAL(8), allocatable :: pb(:,:,:)  ! give the probability tabels
INTEGER, allocatable :: new(:,:,:) ! tell the updated vortex
INTEGER, allocatable :: kstr(:)    ! operator position and property seq along time
INTEGER, allocatable :: ntype(:)   ! which vortex   =1,2,3,4,5,6

!parameters for MC update and sample
INTEGER :: ntoss                   ! thomo steps
INTEGER :: nsample                 ! sample steps
INTEGER :: nbin                    ! number of bins

INTEGER :: ig                      ! number of visit legs not includ ...
INTEGER :: nloop                   ! number of loop update

DOUBLE PRECISION :: Mz,Mz2
DOUBLE PRECISION :: En,En2,Cadd
```

➢ 定义变量和数据 (设置数组长度：allocate_para.f90)

```
subroutine allocate_para
Nsite=nx*ny
Nb=2*Nsite
lmax=100000
allocate(spin(Nsite))
allocate(kbst(2,Nb))
allocate(ntype(lmax),kstr(lmax))
allocate(ktype(6))
allocate(nv(6,0:3))
allocate(w(0:6))
allocate(pb(0:3,0:3,0:6))
allocate(new(0:3,0:3,0:6))
allocate(ns(0:1,0:1,0:1,0:1))
end subroutine allocate_para
```

➢ 定义变量和数据 (定义晶格结构：lattice.f90)

```fortran
!square lattice  with 6 sites in one cell.
!------------------------------------------------------------------------------
subroutine lattice
!-------------------------------------------------------------------------------
!     Constructs the list of sites kbst(1,b) and kbst(2,b) connected by bond b.
!     The x-bonds correspond to b=1,...,N, y-bonds b=N+1,...,2*N
!-------------------------------------------------------------------------------
implicit none
integer :: is
integer :: x0,y0
integer :: xp,yp

is=0
DO y0=1,ny
   DO x0=1,nx
      xp=mod(x0,nx)+1
      yp=mod(y0,ny)+1
      is=is+1
!----------------------------x
      kbst(1,is)=(y0-1)*nx+x0
      kbst(2,is)=(y0-1)*nx+xp
!----------------------------y
      kbst(1,is+Nsite)=(y0-1)*nx+x0
      kbst(2,is+Nsite)=(yp-1)*nx+x0
!-----------------------------------
    ENDDO
ENDDO
!------------------------------------------------------check-lattice
!       print*,'check lattice-bond'
!       do i=1,Nb
!          print*,i,kbst(1:2,i)
!       enddo
!       stop
!------definiton of lattice is OK
!---------------------------------------------------------------------------
return
endsubroutine lattice
```

➢ 定义变量和数据 (构造概率表：weight.f90)

```fortran
subroutine weight
implicit none
integer :: zr,i,j
integer :: is0,is1,ip,lin,lout
real(8) :: w0,w1,w2,w3
real(8) :: a00,a01,a02,a03
real(8) :: a10,a11,a12,a13
real(8) :: a20,a21,a22,a23
real(8) :: a30,a31,a32,a33
integer :: out0,out1,out2,out3
integer :: new0,new1,new2,new3

Cadd=Jz/4.d0+hh/4.d0+0.1d0
pb(0:3,0:3,1:6)=0.d0
new(0:3,0:3,1:6)=-1

ip=0
do i=-1,1,2
   do j=-1,1,2
      ip=ip+1
      nv(ip,0)=i
      nv(ip,1)=j
      nv(ip,2)=i
      nv(ip,3)=j
      w(ip)=Cadd-Jz*i*j/4.d0+hh*(i+j)/2.d0/4.d0
      ns(i,j,i,j)=ip
      ktype(ip)=0
   enddo
enddo

ip=5
nv(ip,0)=0
nv(ip,1)=1
nv(ip,2)=1
nv(ip,3)=0
w(ip)=0.5d0
ns(0,1,1,0)=5
ktype(5)=1
```

```fortran
ip=6
nv(ip,0)=1
nv(ip,1)=0
nv(ip,2)=0
nv(ip,3)=1
ns(1,0,0,1)=6
w(ip)=0.5d0
ktype(6)=1

do i=-1,1,2
   do j=-1,1,2
      if(i==-1.and.j==-1)then
        ip=1
        do lin=0,3
          if(lin==0)then
!   case 1:      in=0, out0=2, out1=3, new0=3,new1=6
          out0=2
          out1=3
          new0=3
          new1=6
         else if(lin==1) then
!   case 2:      in=1, out0=3, out1=2, new0=2, new1=5
          out0=3
          out1=2
          new0=2
          new1=5
         else if(lin==2) then
!   case 3:      in=2, out0=0, out1=1, new0=3, new1=5
          out0=0
          out1=1
          new0=3
          new1=5
         else if(lin==3) then
!   case 4:      in=3, out0=1, out1=0, new0=2, new1=6
```

➢ 定义变量和数据 (构造概率表：weight.f90)

```fortran
!  case 4:     in=3, out0=1, out1=0, new0=2, new1=6
          out0=1
          out1=0
          new0=2
          new1=6
         endif
         w0=w(ip)
         w1=w(new0)
         w2=w(new1)
         a00=0.d0
         a11=0.d0
         a22=0.d0

       if(w2+w1>w0.and.w0+w1>w2.and.w0+w2>w1)then
         a00=0.d0
         a01=(w0+w1-w2)/2.d0
         a02=(w0-w1+w2)/2.d0
         a12=(w2+w1-w0)/2.d0
         a10=a01
         a20=a02
         a21=a12
       else if(w1+w2<=w0)then
         a00=w0-w1-w2
         a01=w1
         a02=w2
         a10=a01
         a12=0.d0
         a20=a02
         a21=a12
       else if(w0+w2<=w1)then
         a11=w1-w0-w2
         a10=w0
         a12=w2
         a01=a10
         a02=0.d0
         a20=a02
         a21=a12
```

```fortran
       else if(w0+w1<=w2)then
         a22=w2-w0-w1
         a20=w0
         a21=w1
         a12=a21
         a02=a20
         a01=0.d0
         a10=a01
       endif

        pb(lin,lin,ip)=a00/w0
        pb(out0,lin,ip)=a01/w0
        pb(out1,lin,ip)=a02/w0
        pb(lin,out0,new0)=a10/w1
        pb(out0,out0,new0)=a11/w1
        pb(out1,out0,new0)=a12/w1
        pb(lin,out1,new1)=a20/w2
        pb(out0,out1,new1)=a21/w2
        pb(out1,out1,new1)=a22/w2

        new(lin,lin,ip)=ip
        new(out0,lin,ip)=new0
        new(out1,lin,ip)=new1
        new(lin,out0,new0)=ip
        new(out0,out0,new0)=new0
        new(out1,out0,new0)=new1
        new(lin,out1,new1)=ip
        new(out0,out1,new1)=new0
        new(out1,out1,new1)=new1
      enddo
      endif    !end i=-1,j=-1 case
```

➢ 定义变量和数据 (构造概率表：weight.f90)

```
!---------------------test---!!!
!do ip=1,6
!   do lin=0,3
!      sum=0.d0
!      do lout=0,3
!         if(pb(lout,lin,ip,ib).lt.eps)then
!            print *, 'negative pb'
!            print *, 'lout=',lout, ' lin=',lin, ' ip=',ip, 'ib=',ib
!            print *, 'pb=',pb(lout,lin,ip,ib)
!         endif
!         sum=sum+pb(lout,lin,ip,ib)
!      enddo
!  write(7,99)(pb(lout,lin,ip,ib),lout=0,3)
!99    format(4f10.5)
!    if(dabs(sum-1.d0).le.1.d-15)then
!    else
!        print *, 'pb sum=',sum
!        print *, 'wrong pb, lin=',lin,'ip=',ip
!    endif
!      enddo
!    enddo
!enddo

return
endsubroutine weight
~
```

➢ 初始化数组或变量 (initial.f90)

```
!------------------------------------------------------------------------
subroutine initconf
!------------------------------------------------------------------------
! Initializes a configuration with random spin state and empty string
!------------------------------------------------------------------------
implicit none
integer :: i
l=20
do i=1,Nsite
   spin(i)=2*min(int((2.d0)*rn()),1)-1
enddo
do i=1,lmax
   kstr(i)=0
enddo
nh=0
return
end subroutine initconf
```

二、位形演化

➢ **Diagonal update(update.f90)**

```
!!!!!!: themolization
ibin=0
do itoss=1,ntoss
   call dupdate
   call lupdate
   anh=anh+nh
   aig=aig+ig

   ra=aig/anh/2.d0
   if (ra.lt.0.8) then
      nloop=nloop*5/3
   elseif (ra.gt.1.5) then
      nloop=nloop*3/4
      if(nloop.lt.1)nloop=1
   endif

   call adjlgth

   if (mod(itoss,ntoss/10).eq.0) then
      call openlog
      write(12,10)itoss,l,nh,ra
10    format('Equilibrated. ',I7,'   L = : ',I8,'   nh= : ',I8, &
             '    ra=:', f8.3)
      call closelog
!-------------------------------------------------------
   endif
enddo
!-------------------------------------------------------
```

```
subroutine dupdate
implicit none
integer :: b,i,is0,is1,is2
integer :: kop,ip
real(8) :: ap1,dp1
do i=1,l
   kop=kstr(i)
   if(kop==0)then
      ntype(i)=0
      b=min(int(rn()*dfloat(Nb))+1,Nb)
      is0=kbst(1,b)
      is1=kbst(2,b)
      ip=ns(spin(is0),spin(is1), &
            spin(is0),spin(is1))
      ap1=beta*Nb*w(ip)/(l-nh)
      if(rn()<ap1)then
         kstr(i)=2*b
         nh=nh+1
         ntype(i)=ip
      endif
   elseif(mod(kop,2)==0)then
      b=kop/2
      ip=ntype(i)
      dp1=1.d0*(l-nh+1)/beta/Nb/w(ip)
      if(rn()<dp1)then
         kstr(i)=0
         nh=nh-1
         ntype(i)=0
        endif
   else
      b=kop/2
      is0=kbst(1,b)
      is1=kbst(2,b)
      ip=ntype(i)
      spin(is0)=nv(ip,2)
      spin(is1)=nv(ip,3)
     endif
enddo
return
end subroutine dupdate
```

➢ Directed-loop update (update.f90)

```fortran
!--------------------------------------------------
subroutine lupdate
!--------------------------------------------------
implicit none
integer :: i,j,b,p0,p1,p2,s0,s1,s2
integer , allocatable ::frst(:)
integer , allocatable ::last(:)
integer , allocatable ::ntime(:)
integer , allocatable ::link(:)
integer :: t0,t01,t02,t1,t2
integer :: i0,i1,i2,it,ip1
integer :: leg,leg0,leg1,leg2
integer :: ni,ki,nj,kj,j0,n0
integer :: ic,ir,irt,ib,ip
logical a1,a2
double precision :: wr,wa,wb
!!**** Construction of the linked vertex list vrtx
allocate(frst(Nsite))
allocate(last(Nsite))
allocate(ntime(0:nh))
allocate(link(0:4*nh))   !!--link table

frst(:)=-1
last(:)=-1

i0=0
i1=1
it=0

do j=1,l
   if(kstr(j)/=0) then
      ntime(it)=j
      b=kstr(j)/2

      s0=kbst(1,b)
      s1=kbst(2,b)
```

```fortran
      p0=last(s0)
      p1=last(s1)

      if (p0/=-1) then
          link(p0)=i0
          link(i0)=p0
      else
          frst(s0)=i0
      endif

      if (p1/=-1) then
          link(p1)=i1
          link(i1)=p1
      else
          frst(s1)=i1
      endif


      last(s0)=i0+2
      last(s1)=i1+2
      i0=i0+4
      i1=i1+4
      it=it+1
   endif
enddo

 do s0=1,Nsite
   i0=frst(s0)
   if (i0/=-1) then
      p0=last(s0)
      link(p0)=i0
      link(i0)=p0
   endif
enddo
```

➢ Directed-loop update（update.f90）

```fortran
!*******Loop update
ig=0
if (nh==0) return
do i=1,nloop
   p0=min(int(dfloat(4*nh)*rn()),4*nh-1)
!-----------------------------------------
   p1=p0
602  i1=p1/4
     leg0=mod(p1,4)
     i0=ntime(i1)
     ip=ntype(i0)
     wr=rn()
     wb=0
     do j=0,3
        wa=wb
        wb=wb+pb(j,leg0,ip)
        if(wr>wa.and.wr<=wb) then
           leg1=j
           goto 601
        endif
     enddo

601  ntype(i0)=new(leg1,leg0,ip)
     p2=4*i1+leg1
     p1=link(p2)
!-----------------------------------------
     if(leg0/=leg1) then
       ig=ig+1
     endif
     if(p1/=p0.and.p2/=p0) then
        goto 602
     else
        goto 607
     endif
607  continue
enddo
```

```fortran
!!!!**** Mapping of updated vertices to operator sequence
do i0=1,l
   if(kstr(i0)/=0) then
      b=kstr(i0)/2
      ip=ntype(i0)
      kstr(i0)=2*b+ktype(ip)
   endif
enddo

!!!**** Flipping of spins affected by loop updates,
!!!**** and random free spins
do j0=1,Nsite
   i0=frst(j0)
   if(i0/=-1) then
     i1=ntime(int(i0/4))
     leg0=mod(i0,4)
     ip=ntype(i1)
     n0=nv(ip,leg0)
     spin(j0)=n0
   else
     spin(j0)=2*min(int(2.d0*rn()),1)-1
   endif
enddo
!-----------------------------------------
deallocate(frst)
deallocate(last)
deallocate(ntime)
deallocate(link)
return
end subroutine lupdate
```

➢ 增加序列长度（initial.f90）

```fortran
!-----------------------------------------------------------------------------
subroutine adjlgth
!-----------------------------------------------------------------------------
!     Increases the cut-off l to nh+nh/4 if l is currently less than this.
!     Distributes the nh non-0 operators randomly among the new l positions.
!-----------------------------------------------------------------------------
implicit none
integer :: l1,i,j
double precision :: r
!-----------------------------------------------------------
l1=nh+nh/5
if (l1.le.l) then
   return
else if(l1.gt.lmax) then
   call openlog
   write(12,*)'L exceeded maximum ',l1,lmax
   call closelog
stop
endif

j=0
DO i=1,l
   IF (kstr(i).NE.0) THEN
      j=j+1
      kstr(j)=kstr(i)
      ntype(j)=ntype(i)
   ENDIF
ENDDO
DO i=nh+1,l1
   kstr(i)=0
   ntype(i)=0
ENDDO
l=l1
j=nh
r=DFLOAT(nh)/DFLOAT(l)
DO i=l,1,-1
   IF (j.GT.0.AND.j.LT.i.AND.RN().LT.r) THEN
      kstr(i)=kstr(j)
      kstr(j)=0
      ntype(i)=ntype(j)
      ntype(j)=0
      j=j-1
   ENDIF
ENDDO
return
end subroutine adjlgth
```

三、抽样测量

➢ **测量物理量(measure.f90)**

```fortran
subroutine measure
implicit none
real(8) :: tot,en_tmp
en_tmp=1.d0*nh
En=En+en_tmp
En2=En2+en_tmp*en_tmp

tot=1.d0*sum(spin(1:Nsite))
Mz=Mz+tot
Mz2=Mz2+tot*tot
return
end subroutine measure
```

四、结果输入

➢ 输出测量物理量（results.f90）

```
subroutine results(ibin)
implicit none
INTEGER :: ibin

En=-En/dfloat(nsample)/dfloat(Nsite)/beta+Cadd/2.d0
En2=En2/dfloat(nsample)/dfloat(Nsite*Nsite)/beta/beta

Mz=Mz/dfloat(nsample)/dfloat(Nsite)
Mz2=Mz2/dfloat(nsample)/dfloat(Nsite)/dfloat(Nsite)

open(unit=68,file='result.dat',status='unknown', access='append')

write(68,66)nx,beta,En,En2,Mz,Mz2,Jz,hh,iseed,ibin
66    format(i5,f12.6,4f22.14,2f9.5,2i5)

close(68)
return
end subroutine
```

Thanks for your attentions！