

テンソル繰り込み群のためのテンソルネットワークの 構成手法と初期テンソル依存性

[D. Kadoh, K.N. arXiv:1912.02414]
[K.N. arXiv:2307.14191]
[K. N., M.Schneider arXiv:2407.14226]

中山 勝政 (RIKEN)

2025/01/22@Kyoto Univ.

テンソルネットワーク法の概観

● テンソルに関する呼び方

$$A_{xyx'y'} = \begin{array}{c} y' \\ | \\ x \text{ --- } | \text{ --- } x' \\ | \\ y \end{array}$$

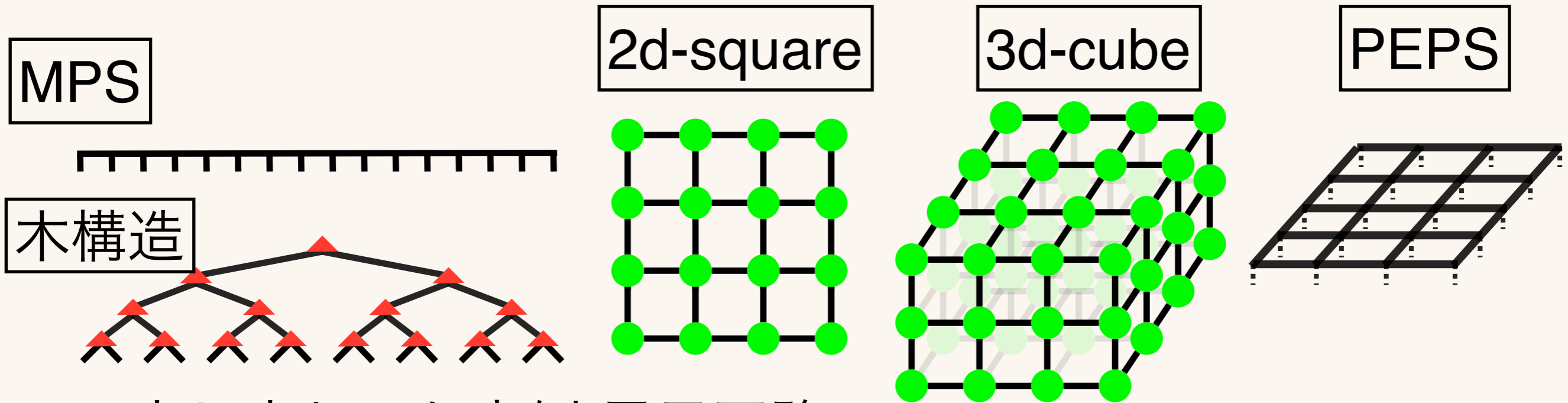
各添字の要素数 D : ボンド次元、添字の最大サイズ
(例 $x = 1, \dots, D$, $y = 1, \dots, D$, etc.)

→ 今回のトークでは添字は全てボンド次元 D で考える

テンソルのオーダー(次数): 添字(ベクトル空間、足)の数
(例 $A_{abcd} \rightarrow$ オーダー 4, $B_{abcdef} \rightarrow$ オーダー 6)

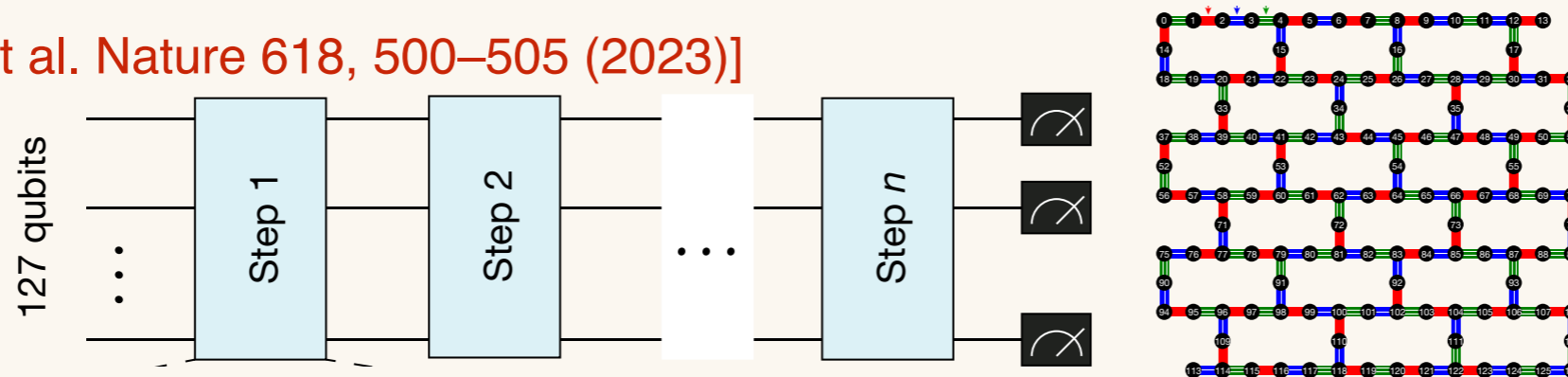
● テンソルネットワーク表現

- ◇ 物理量や量子状態がテンソルネットワークで表現可能。



- ◇ 少し変わった実例:量子回路

[Y.Kim et al. Nature 618, 500–505 (2023)]



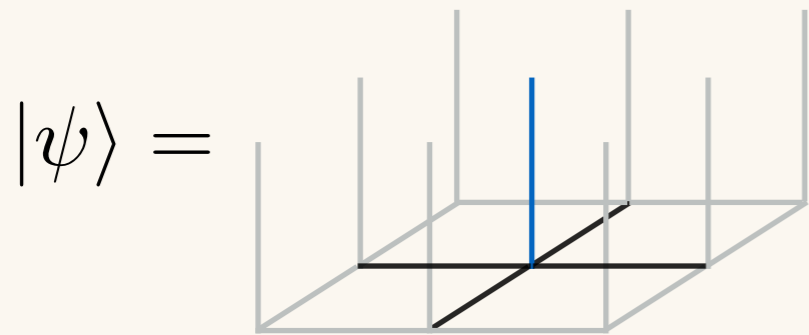
- テンソルネットワーク:テンソル=点、添字=線分で表現
- 添字を全て計算して和を取り切ると物理量に。

● テンソルネットワーク法の種類

◇ (日本で)よく使われる分類:

ハミルトニアン形式

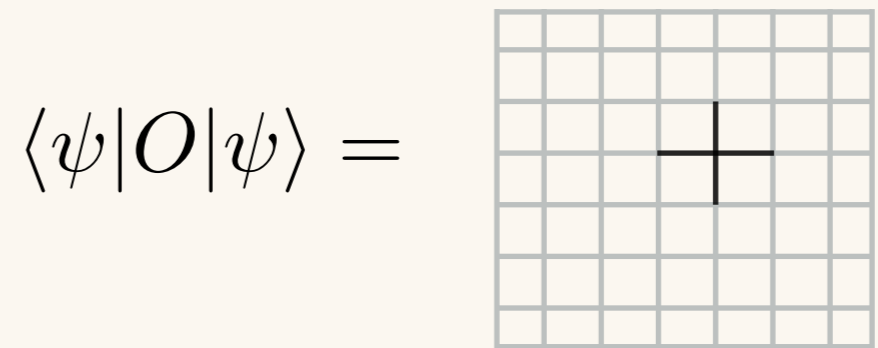
→ 量子状態がネットワーク



- ◇ 密度行列くりこみ群 (DMRG), 1+1次元
- ◇ PEPS, 2+1次元

ラグランジアン形式

→ 物理量がネットワーク



- ◇ テンソル繰り込み群 (TRG)

→ 変分法や特異値分解(SVD)を使うなど共通点は非常に多い

● テンソル繰り込み群(TRG)

[M. Levin, C.P. Nave [arXiv:cond-mat/0611687](https://arxiv.org/abs/cond-mat/0611687)]

◇ TRG = テンソルネットワークの近似計算手法

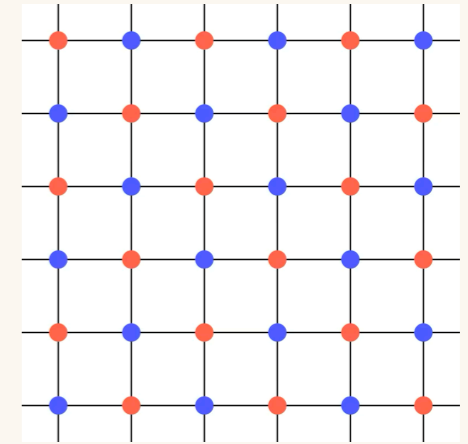
[https://smorita.github.io/TN_animation/]

→ 128×128 個の格子点上の

テンソルの厳密計算(イジング模型)では、

$$2^{128} \times 2^{128} \times 2^{128} \times 2^{128}$$

という途方もない大きさのテンソルが登場。



→ TRGなら例えば $16 \times 16 \times 16 \times 16$ まで圧縮。

◇ 長所と短所

○ 符号問題の回避:

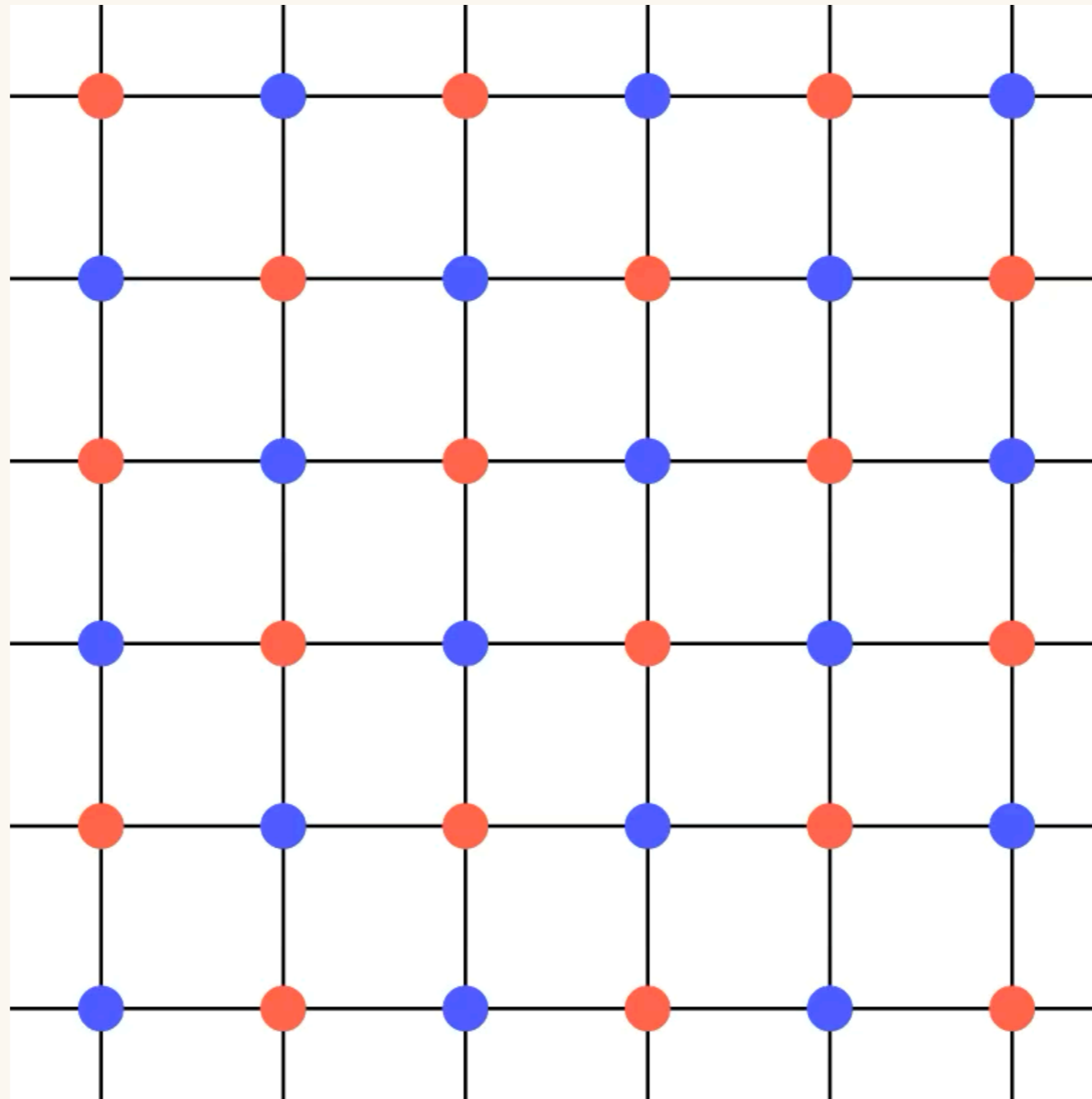
有限密度系の計算が可能

×高次元や複雑な系で高コスト

×系統誤差の存在

● 粗視化の様子

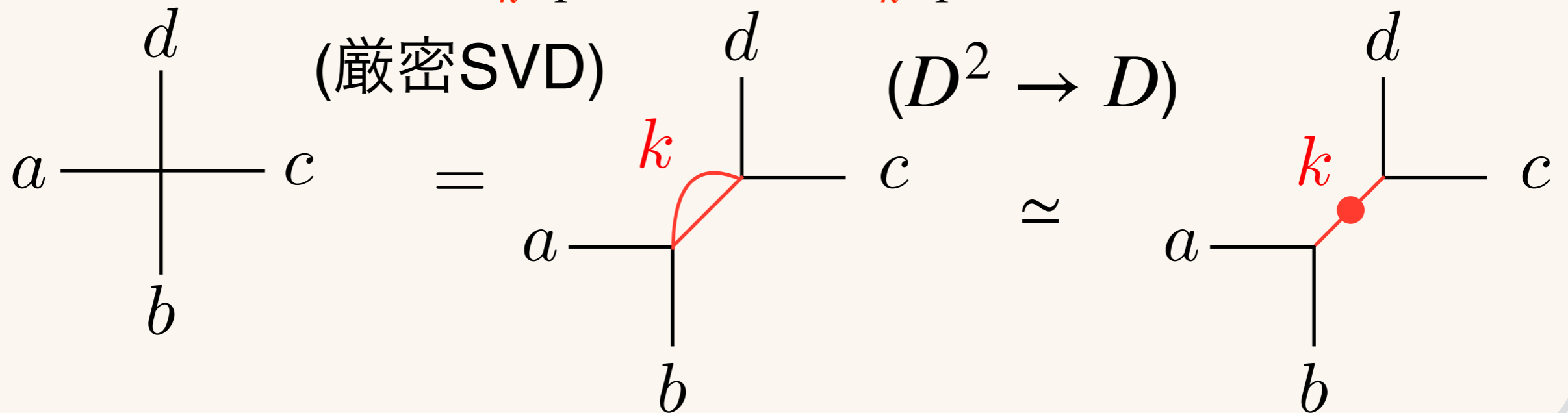
[https://smorita.github.io/TN_animation/]



- TRGは分解と縮約を繰り返して粗視化していく
- ネットワークの圧縮はSVDか変分法。

● 特異値分解 (SVD)

$$T_{abcd} = \sum_{k=1}^{D^2} A_{ab}^k \lambda^k B_{cd}^k \simeq \sum_{k=1}^D A_{ab}^k \lambda^k B_{cd}^k \quad (A, B: (\text{打ち切り}) \text{ ユニタリ行列})$$



◇ より大きな特異値 λ がより近似に重要 (Frobenius norm)

$$\|A\| = \sqrt{\text{tr}(A^\dagger A)} = \sqrt{\sum \lambda^2}$$

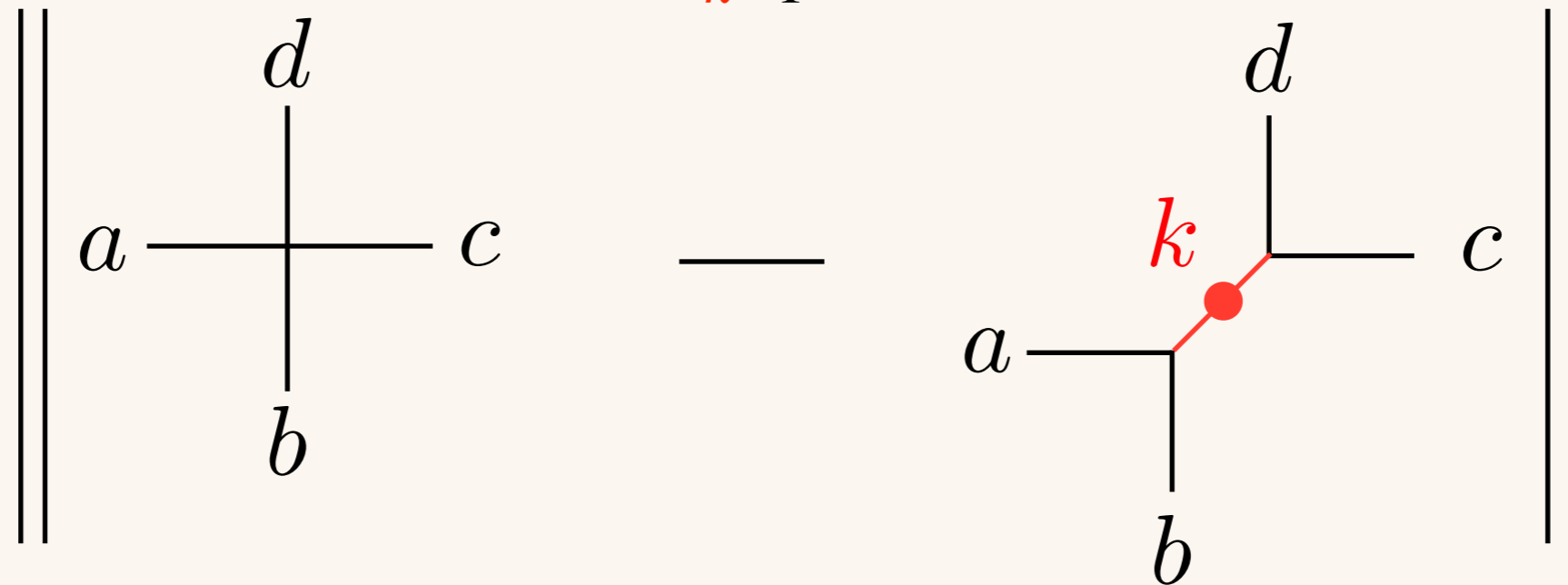
→ 添字 k を打ち切って近似する

厳密な分解からの打ち切り $D^2 \rightarrow D$

● 変分法

$$\|A\| = \sqrt{\text{tr}(A^\dagger A)} = \sqrt{\sum \lambda^2}$$

$$T_{abcd} \simeq \sum_{k=1}^D A_{ab}^k \lambda^k B_{cd}^k$$



◇ このコスト関数を逐次的に最小化させる手法

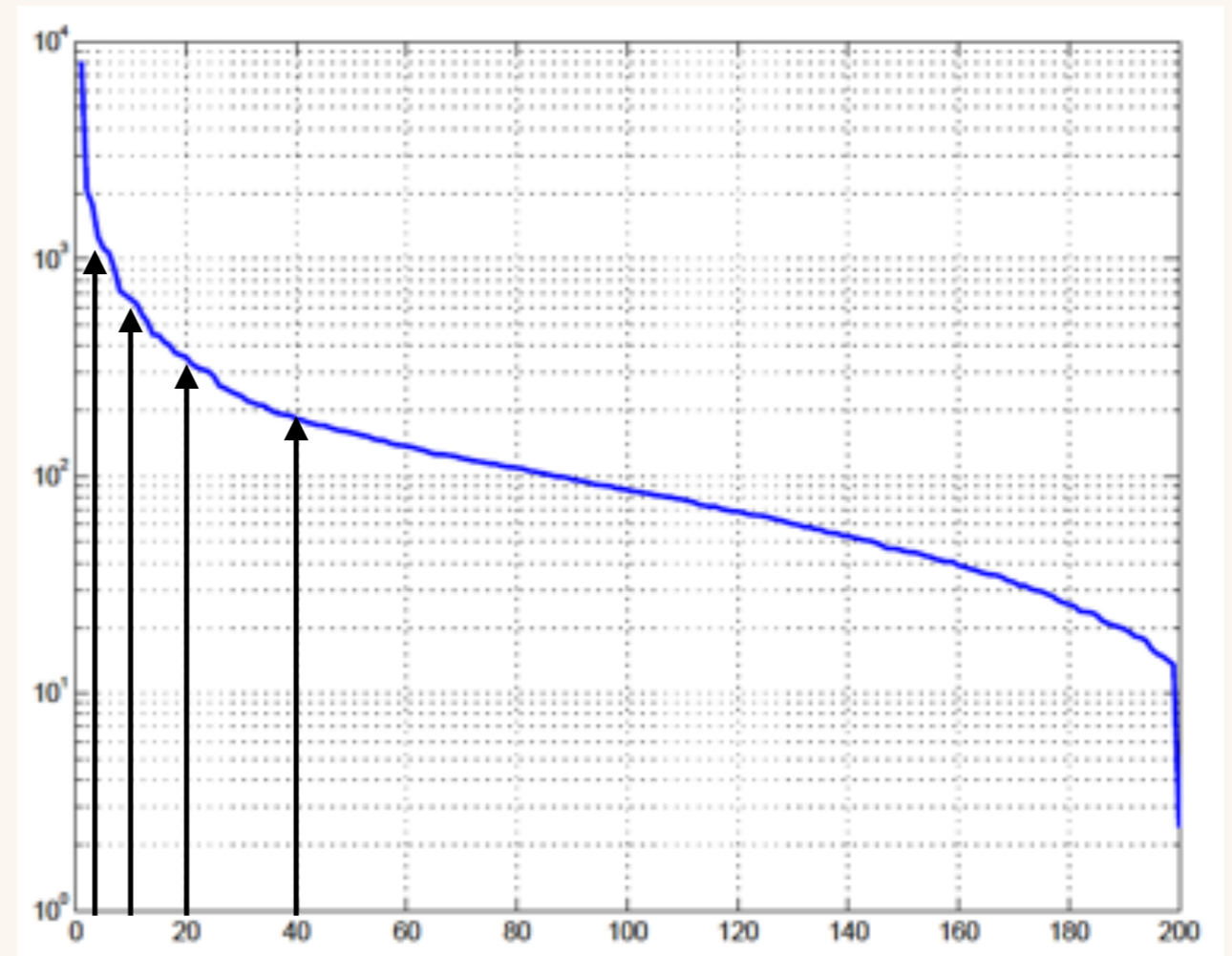
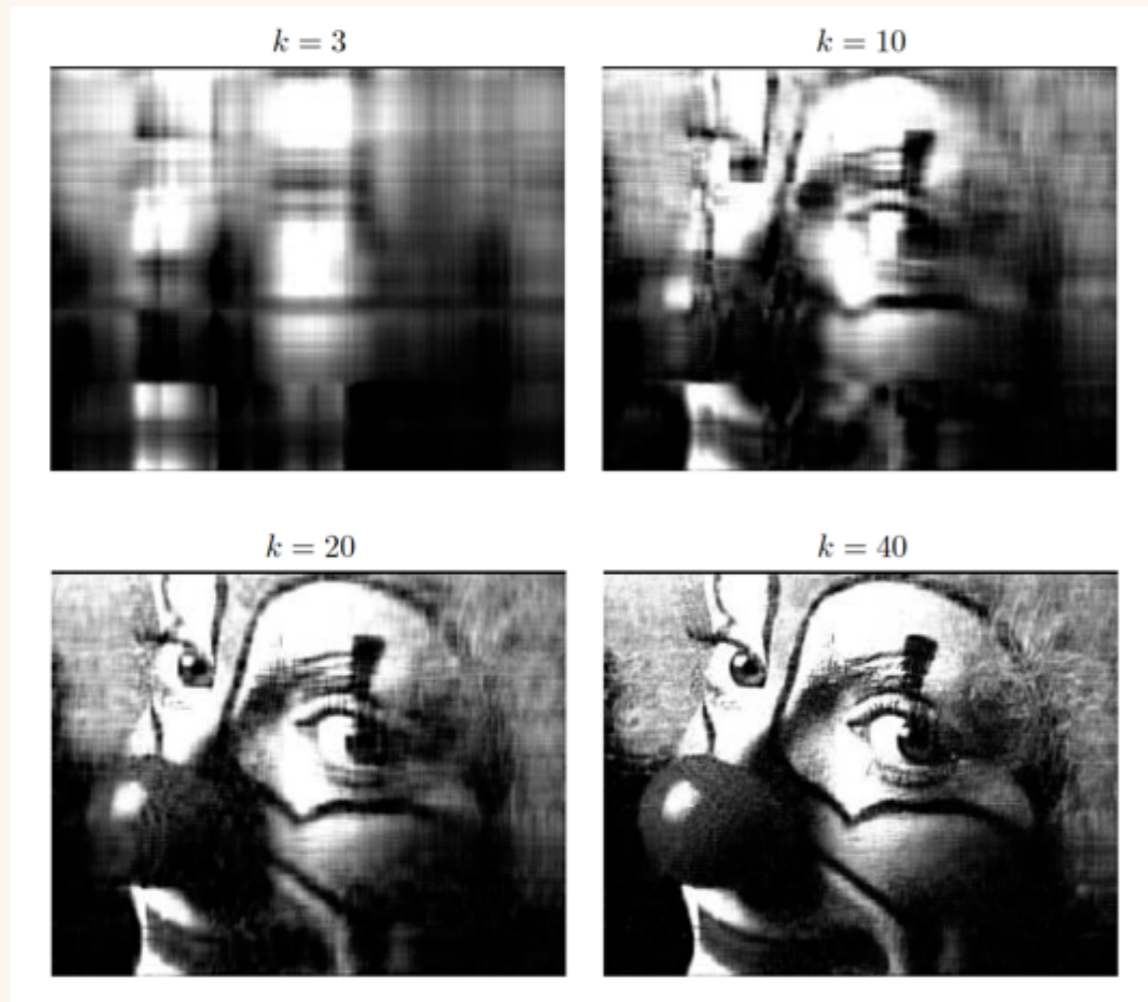
→ 添字 k を最初から小さく限定して最適化する

→ 模式的には、左のテンソルを右のテンソルで近似してる。

→ テンソルネット法の近似はほぼSVDか変分法のみ

● SVDによる粗視化(e.g. 画像圧縮)

◇ この強烈な近似はしていい？



[<http://www.na.scitec.kobe-u.ac.jp/~yamamoto/lectures/cse-introduction2009/cse-introduction090512.PPT>]

→ 10/200とか20/200といった圧縮がうまくいく。

● テンソルネットワーク表現の構成方法

- ◇ ハミルトニアン形式ならITensorとかで(簡単なものは)、ハミルトニアンが書き下せれば大体計算できる。

<https://itensor.org/docs.cgi?page=classes/dmrg>

- ◇ TRGでは物理量のテンソルネットワーク表現が必要。
- ◇ 近年の発展が目覚ましいのはいいが、興味のある系にTRGを使いたいときに具体的にどうすればいいのか？

→ 使い慣れたラグランジアンや経路積分、場の理論から、
系の詳細によらずネットワーク表現を見つけたい。

(とりあえず使ってみたい！ くらいで使えるように)

● テンソルネットワーク表現の構成方法

(e.g.): 二次元イジング(分配関数)

$$Z = \sum_{\sigma} \prod_{x,y} e^{\sigma_{x,y}\sigma_{x+1,y} + \sigma_{x,y}\sigma_{x,y+1}}$$

(e.g.): Z_2 ゲージ理論

$$Z = 2^{-3V} \sum_{\sigma} \prod_{n,\mu > \nu} e^{-\beta \sigma_{n,\mu} \sigma_{n+\hat{\mu},\nu} \sigma_{n+\hat{\nu},\mu} \sigma_{n,\nu}}$$

→ 見慣れた形式そのものはネットワーク表現ではない。

テンソル繰り込み群

● テンソル繰り込み群にはどんな種類がある？

特異値分解

近似手法

or

変分法

計算量削減 = 低オーダー近似
追加分解 or SVDの打ち切り方 or 境界条件

精度改善
コスト関数の範囲拡大, 最適化パラメータの増加,
近距離相関を取り除く (Disentangler)

→ 基本的には全ての手法がこれらの組み合わせで理解できる
(近似を表現するコスト関数を見ると違いがわかりやすい)

Simple TRG,

Anisotropic TRG,

Bond-weighted TRG,

Core TRG,

CTMRG,

GILT,

HOTRG,

Randomized TRG,

SRG,

TNR,

Loop-TNR,

Triad TRG,

MDTRG,

ALL-mode TRG,

Branching TRG,

Boundary HOTRG

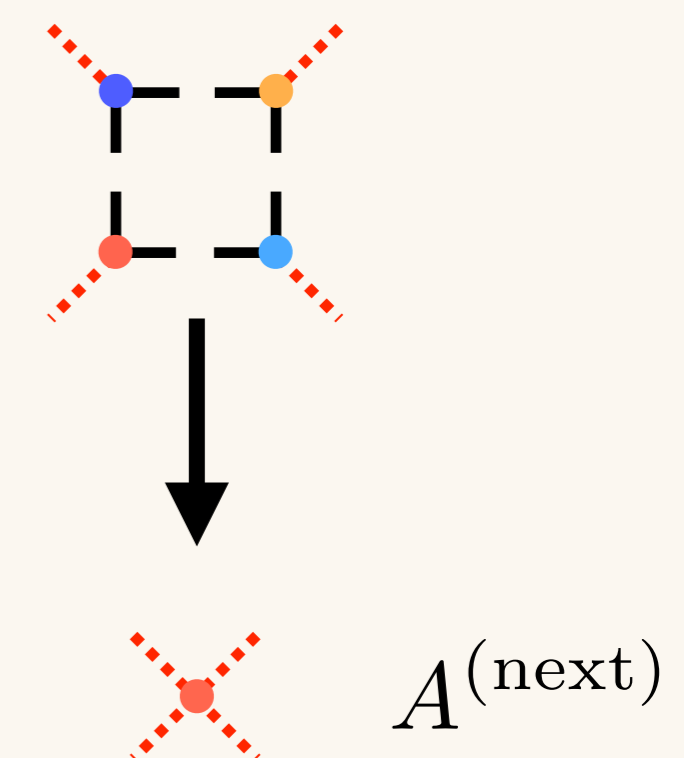
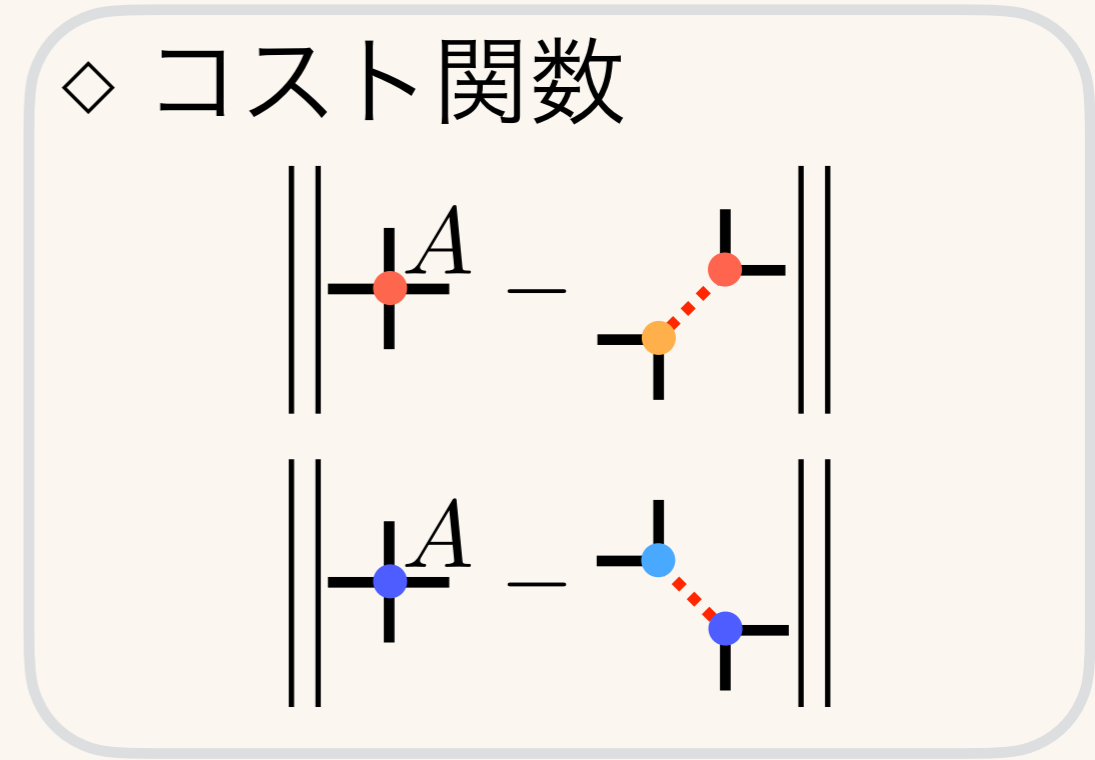
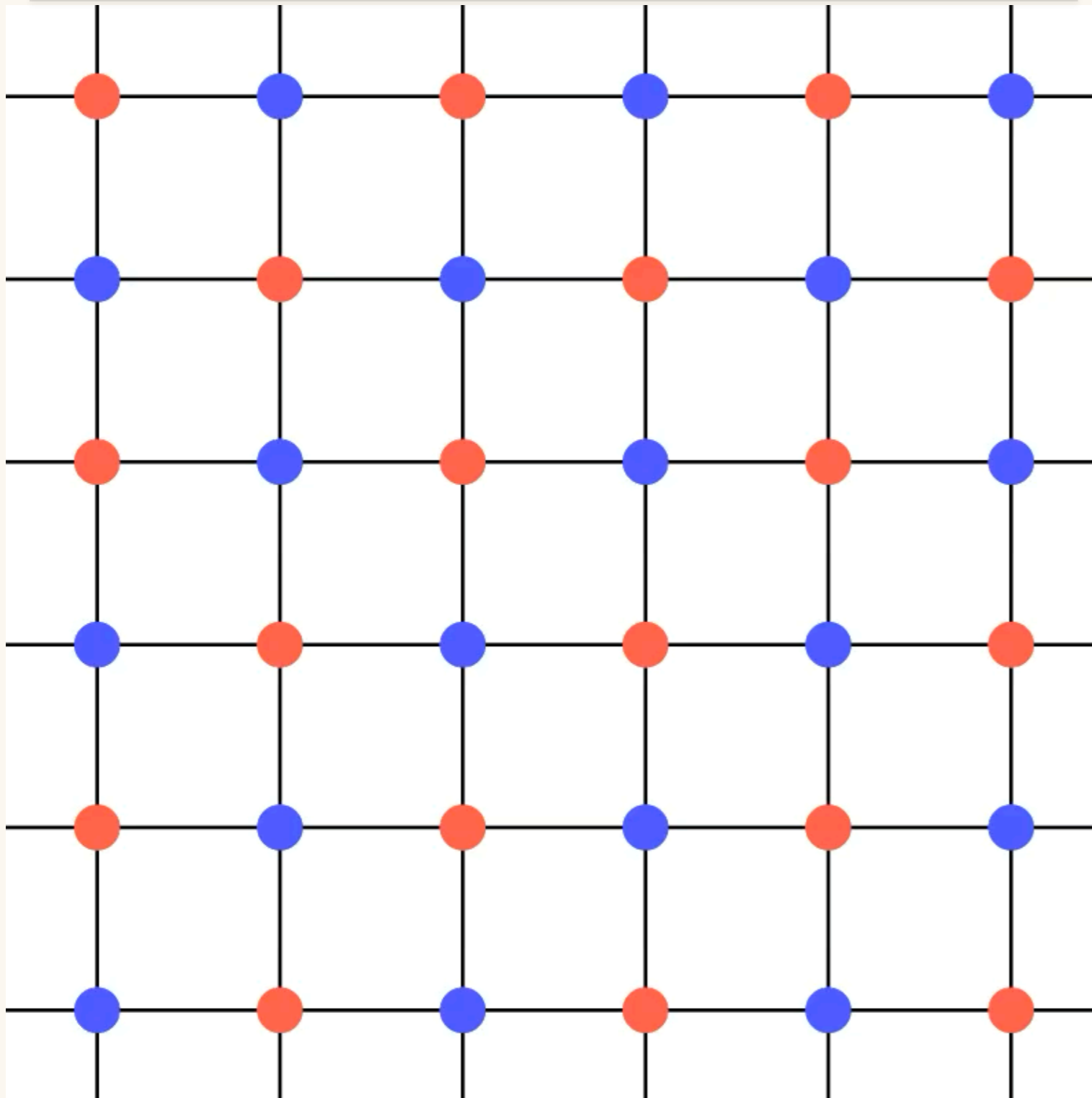
CTM-TRG

NNR-TNR

...etc.の組み合わせ

● Cause-graining by TRG

◇ 特異値分解を用いた近似的縮約



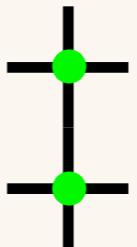
● 特異値分解と射影テンソル

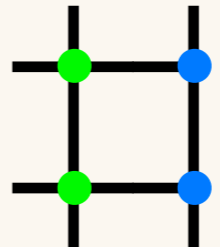
◇ 特異値分解による打ち切りは射影テンソルで表現可能。

$$T_{abcd} \simeq \sum_{k=1}^D A_{ab}^k \lambda^k B_{cd}^k \quad (A, B: (\text{打ち切り}) \text{ ユニタリ行列})$$

$$\sum_{c', d'} T_{abc'd'} B_{c'd'}^k \dagger B_{cd}^k = \sum_{k=1}^D A_{ab}^k \lambda^k B_{cd}^k \iff \text{Diagrammatic Form} = \text{Diagrammatic Form}$$

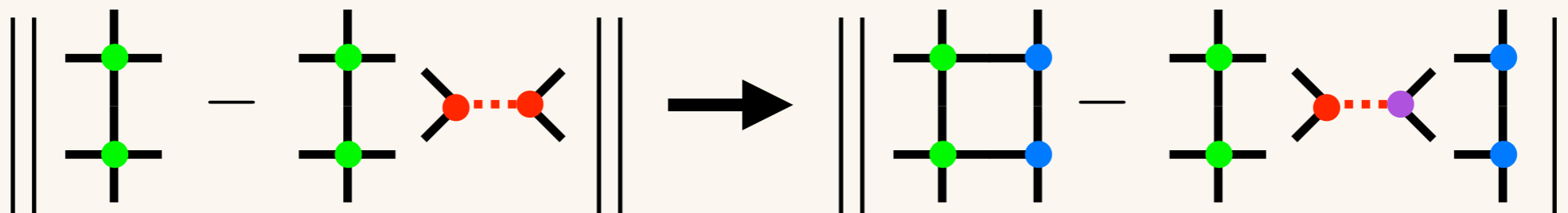
● Boundary-HOTRG

→ HOTRGでのIsometryは  に対して(HO)SVDを考えた。

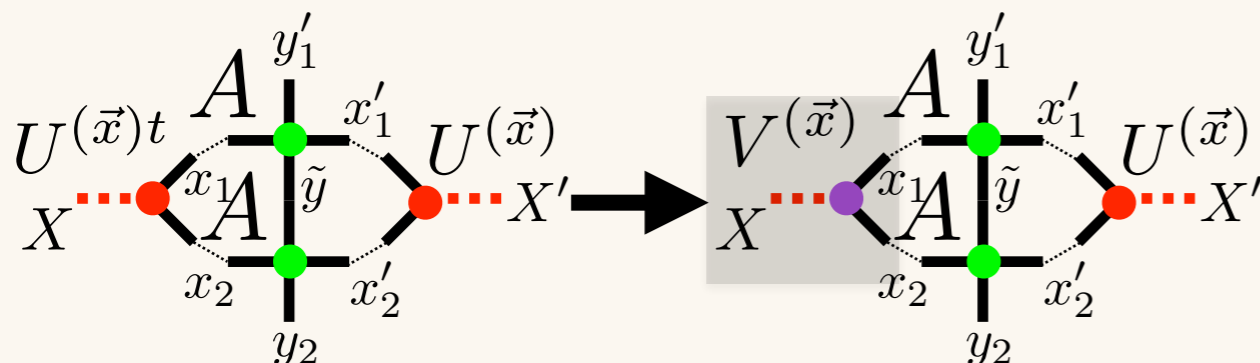
→ Boundary HOTRGでのIsometryは  に対してSVDを考える。

HOTRG

boundary HOTRG

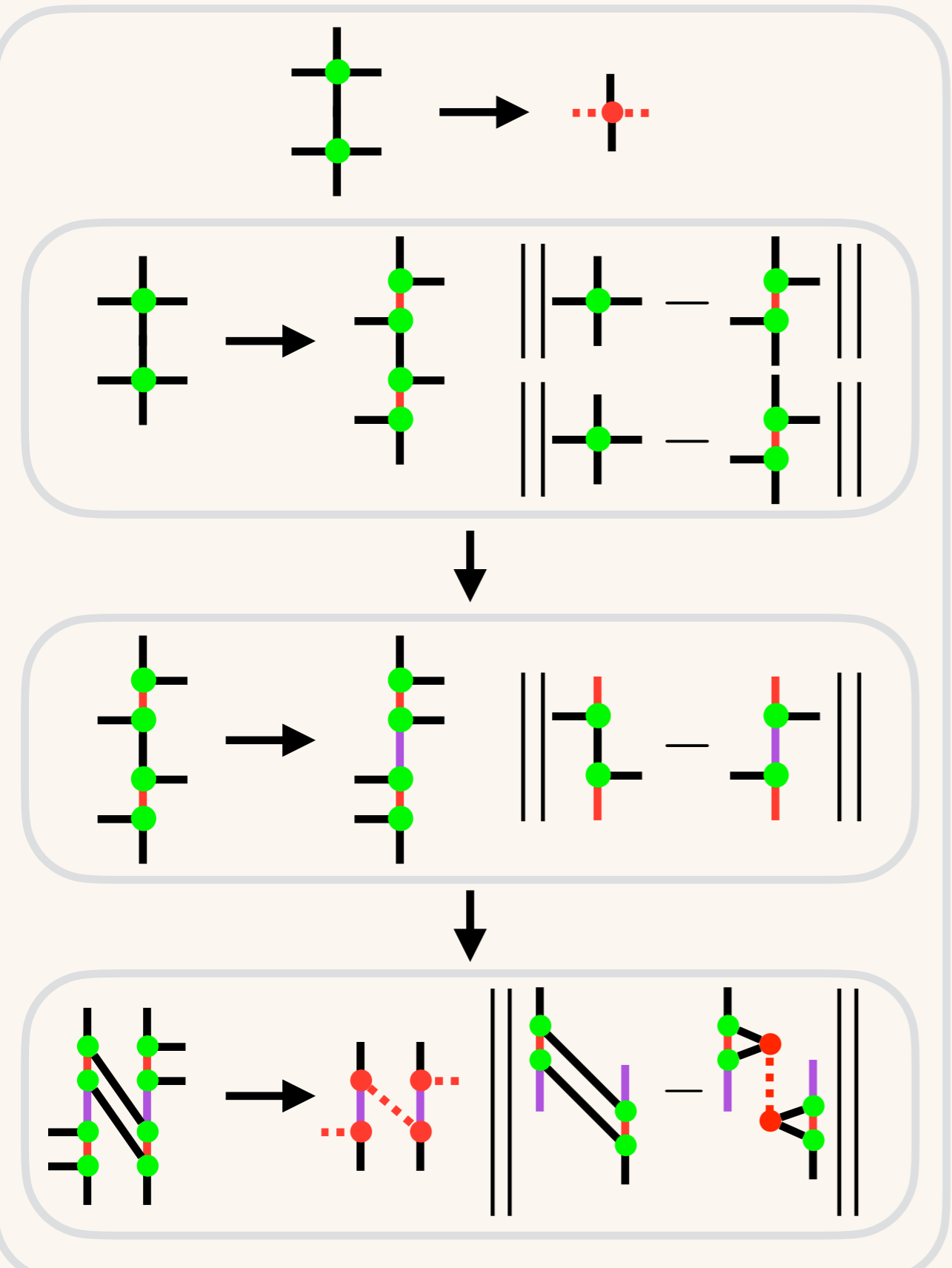


→ 隣からの寄与を取り入れるためにコスト関数が違う。



→ よりGlobalな近似を与える
計算量のオーダーを変えない
良い方法を提案。

● Anisotropic TRG (ATRG)



◇ 追加の分解でテンソルの
オーダーを落とす

オーダー $d+1$ まで分解

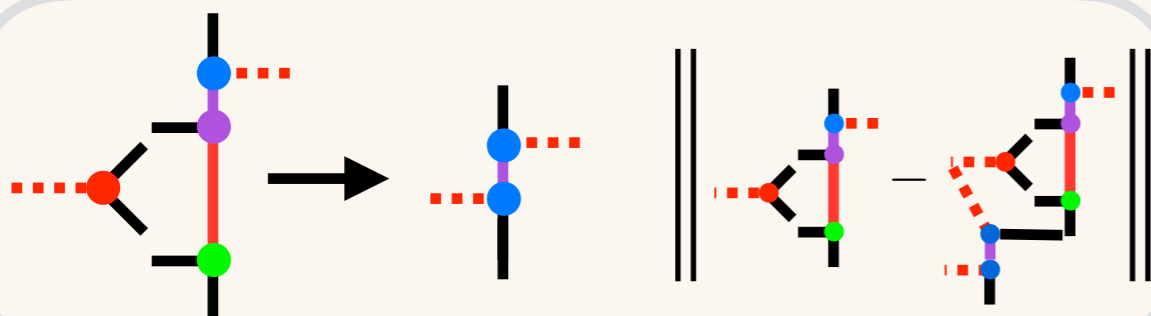
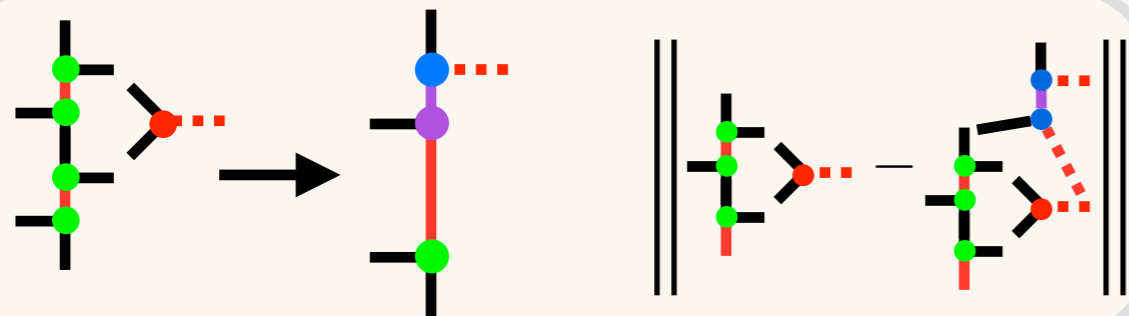
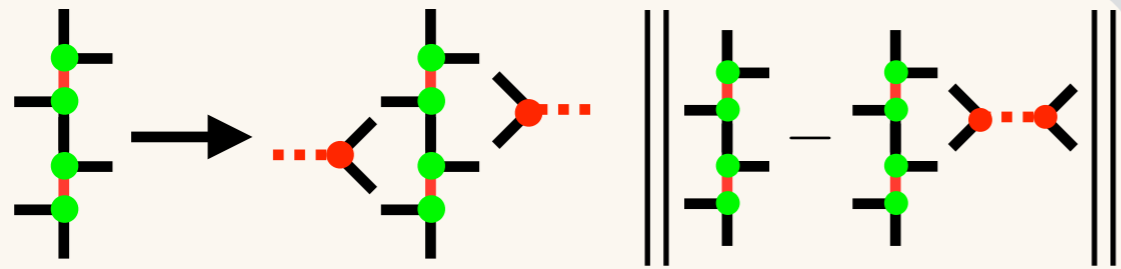
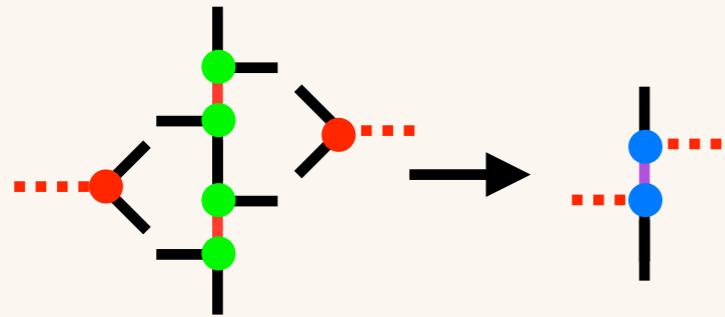
隣接したテンソルで縮約と
分解を考えていく。

複数のコスト関数を順番に
最適化していく。

◇ 追加分解は計算量を削減

$$O(D^{4\text{dim}-1}) \rightarrow O(D^{2\text{dim}+1})$$

● Triad RG



◇ 任意のネットは3本足
(Triad)テンソルで表現できる。
オーダー 3を基本に考える。

計算途中でも大きなオーダー
にならないように。

複数のコスト関数を順番に
最適化していく。

◇ 追加分解は計算量を削減

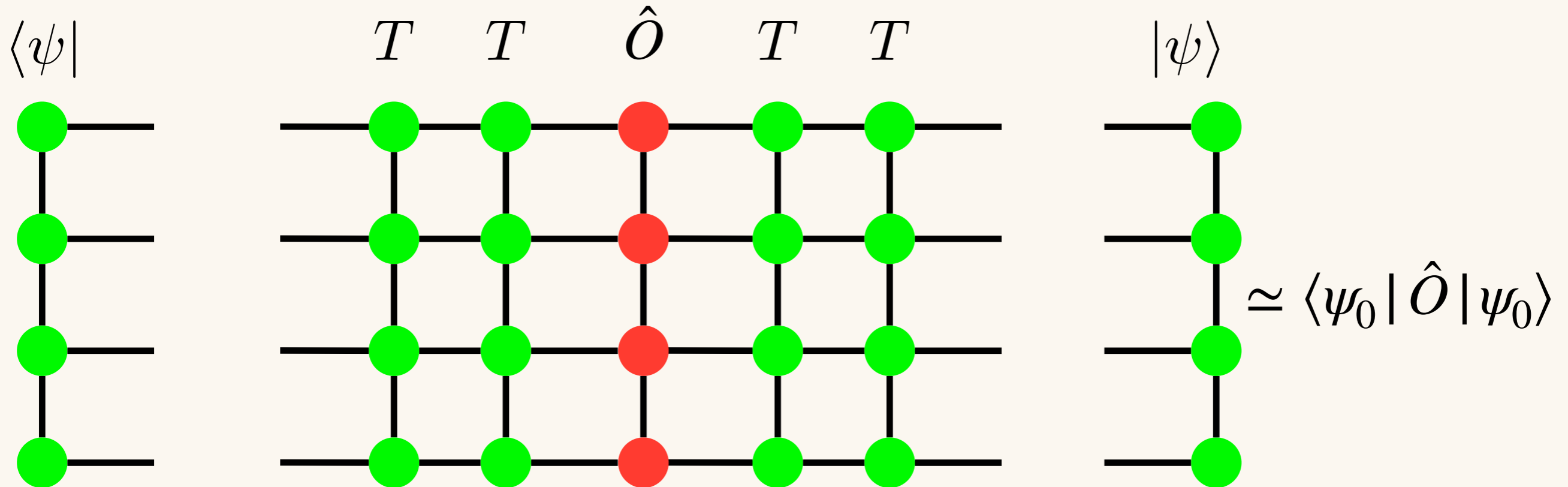
$$O(D^{4\text{dim}-1}) \rightarrow O(D^{\text{dim}+3})$$

● DMRGとTRG

T:微小(虚)時間発展

$|\psi\rangle$:初期状態

- ◇ 単純なDMRGは下のネットワークの縮約。



→ DMRGでは変分法を使うことが多い。

→ 特異値分解の組み合わせ、TRGで近似縮約してもいい。

TRGはラグランジアンを必要としない。

(DMRGもハミルトニアンとは無関係に考えることも可能)

Kicked Ising model on the Quantum computer

IBM's paper [Kim, Y et al. *Nature* 618, 500–505 (2023, June)]

Article

Evidence for the utility of quantum computing before fault tolerance

<https://doi.org/10.1038/s41586-023-06096-3>

Received: 24 February 2023

Youngseok Kim^{1,6}, Andrew Eddins^{2,6}, Sajant Anand³, Ken Xuan Wei¹, Ewout van den Berg¹, Sami Rosenblatt¹, Hasan Nayfeh¹, Yantao Wu^{3,4}, Michael Zaletel^{3,5}, Kristan Temme¹ & Abhinav Kandala¹

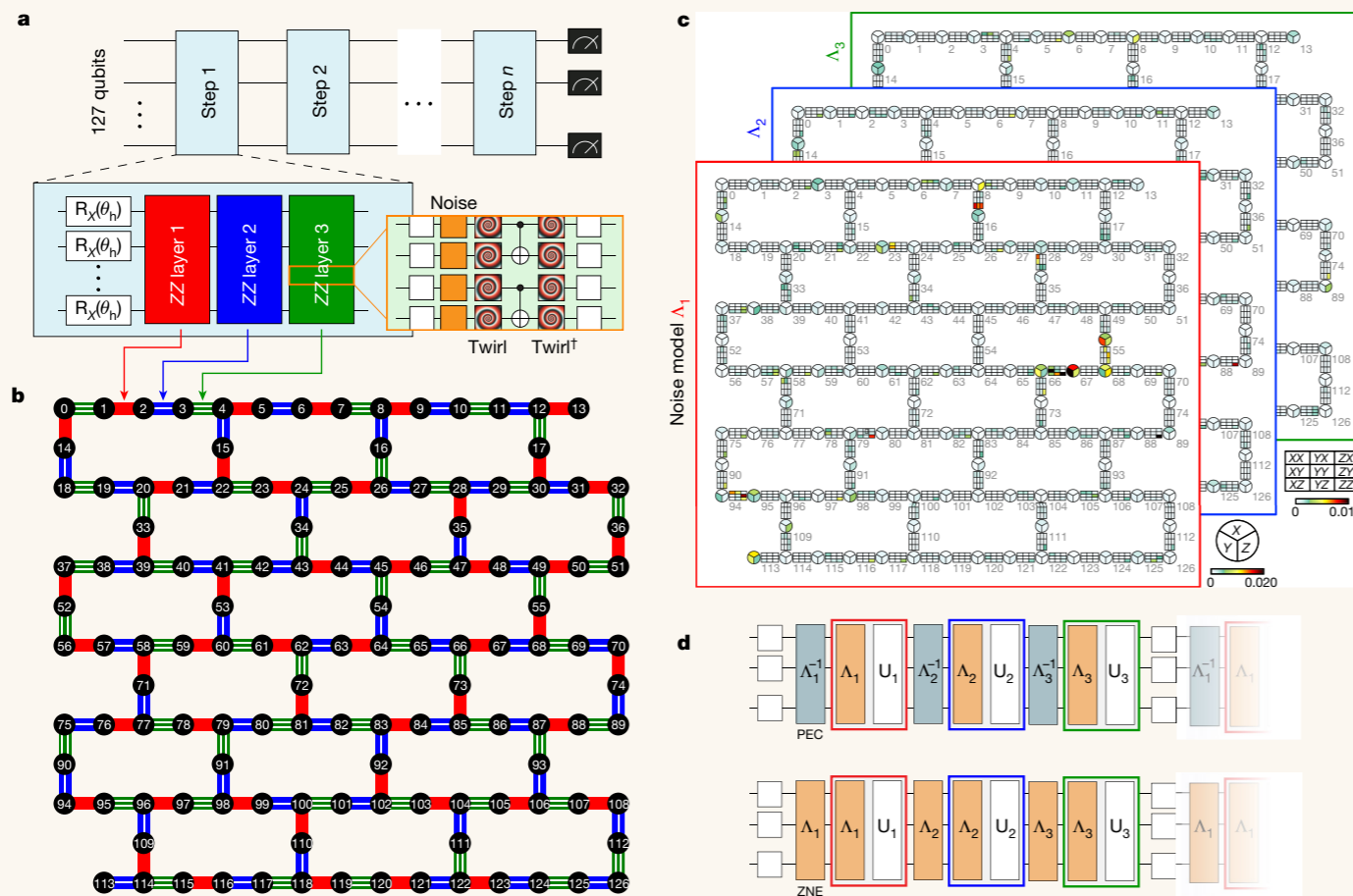


Fig. 1 | Noise characterization and scaling for 127-qubit Trotterized time-evolution circuits. **a**, Each Trotter step of the Ising simulation includes single-qubit X and two-qubit ZZ rotations. Random Pauli gates are inserted to twirl (spirals) and controllably scale the noise of each CNOT layer. The dagger indicates conjugation by the ideal layer. **b**, Three depth-1 layers of CNOT gates suffice to realize interactions between all neighbour pairs on `ibm_kyiv`.

c, Characterization experiments efficiently learn the local Pauli error rates $\lambda_{l,i}$ (colour scales) comprising the overall Pauli channel Λ_l associated with the l th twirled CNOT layer. (Figure expanded in Supplementary Information IV.A). **d**, Pauli errors inserted at proportional rates can be used to either cancel (PEC) or amplify (ZNE) the intrinsic noise.

◇ Transverse Ising (2+1 dim)
(Kicked Ising)

$$H = -J \sum_{\langle i,j \rangle} Z_i Z_j + h \sum_i X_i,$$

$$e^{-iH_{ZZ}\delta t} = \prod_{\langle i,j \rangle} e^{iJ\delta t Z_i Z_j} = \prod_{\langle i,j \rangle} R_{Z_i Z_j}(-2J\delta t)$$

$$e^{-iH_X\delta t} = \prod_i e^{-ih\delta t X_i} = \prod_i R_{X_i}(2h\delta t),$$

IBM's paper

Quantum

Classical

○ Unmitigated ● Mitigated — MPS ($\chi = 1,024$; 127 qubits) — isoTNS ($\chi = 12$; 127 qubits) — Exact

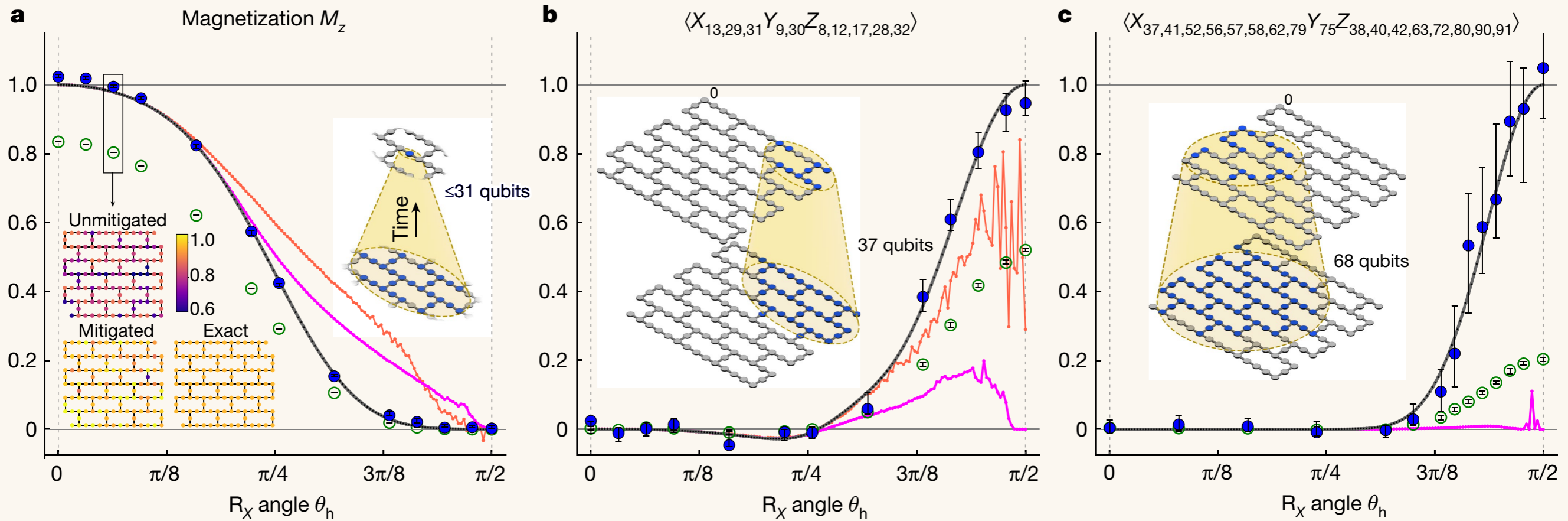


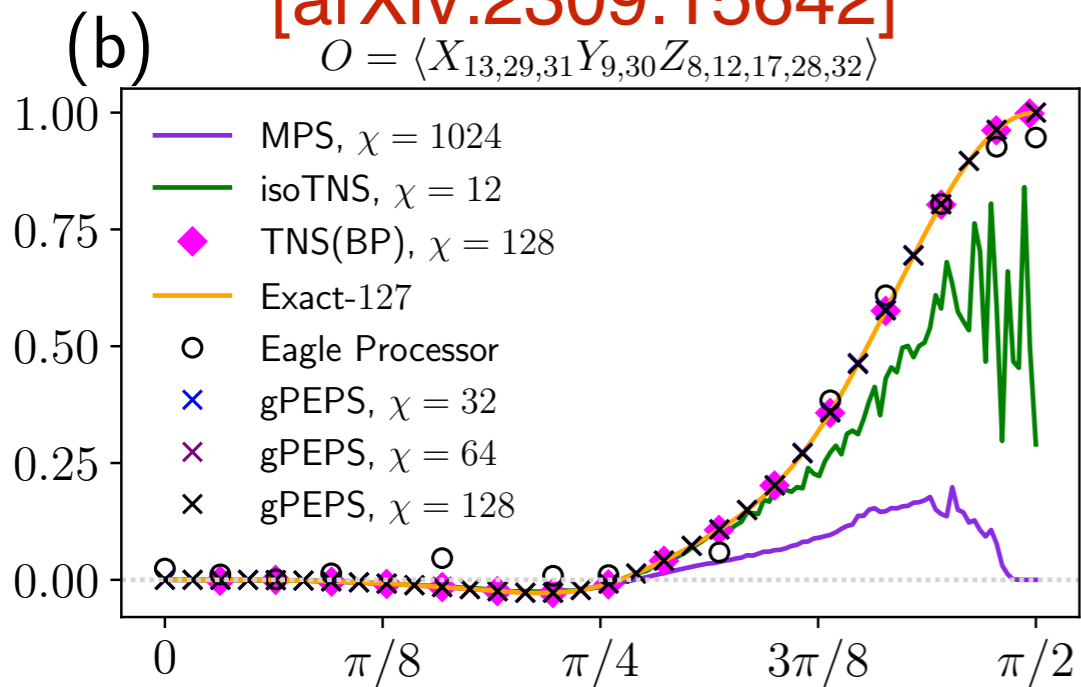
Fig. 3 | Classically verifiable expectation values from 127-qubit, depth-15 Clifford and non-Clifford circuits. Expectation value estimates for θ_h sweeps at a fixed depth of five Trotter steps for the circuit in Fig. 1a. The considered circuits are non-Clifford except at $\theta_h = 0, \pi/2$. Light-cone and depth reductions of respective circuits enable exact classical simulation of the observables for all θ_h . For all three plotted quantities (panel titles), mitigated experimental results (blue) closely track the exact behaviour (grey). In all panels, error bars indicate 68% confidence intervals obtained by means of percentile bootstrap. The weight-10 and weight-17 observables in **b** and **c** are stabilizers of the circuit at $\theta_h = \pi/2$ with respective eigenvalues +1 and -1; all values in **c** have been negated for visual simplicity. The lower inset in **a** depicts variation of $\langle Z_q \rangle$ at $\theta_h = 0.2$ across the device before and after mitigation and compares with exact results.

Upper insets in all panels illustrate causal light cones, indicating in blue the final qubits measured (top) and the nominal set of initial qubits that can influence the state of the final qubits (bottom). M_z also depends on 126 other cones besides the example shown. Although in all panels exact results are obtained from simulations of only causal qubits, we include tensor network simulations of all 127 qubits (MPS, isoTNS) to help gauge the domain of validity for those techniques, as discussed in the main text. isoTNS results for the weight-17 operator in **c** are not accessible with current methods (see Supplementary Information VI). All experiments were carried out for $G = 1, 1.2, 1.6$ and extrapolated as in Supplementary Information II.B. For each G , we generated 1,800–2,000 random circuit instances for **a** and **b** and 2,500–3,000 instances for **c**.

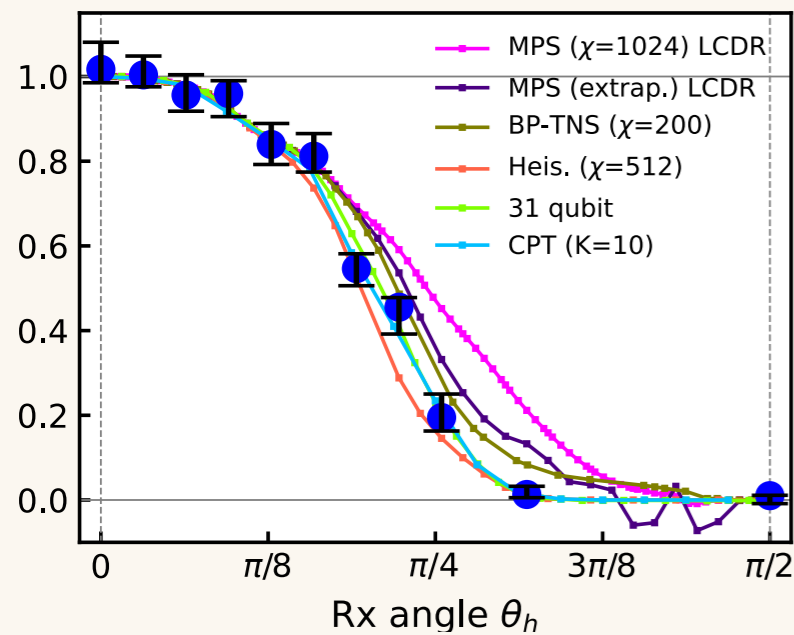
→ Quantum supremacy!?

[arXiv:2309.15642]

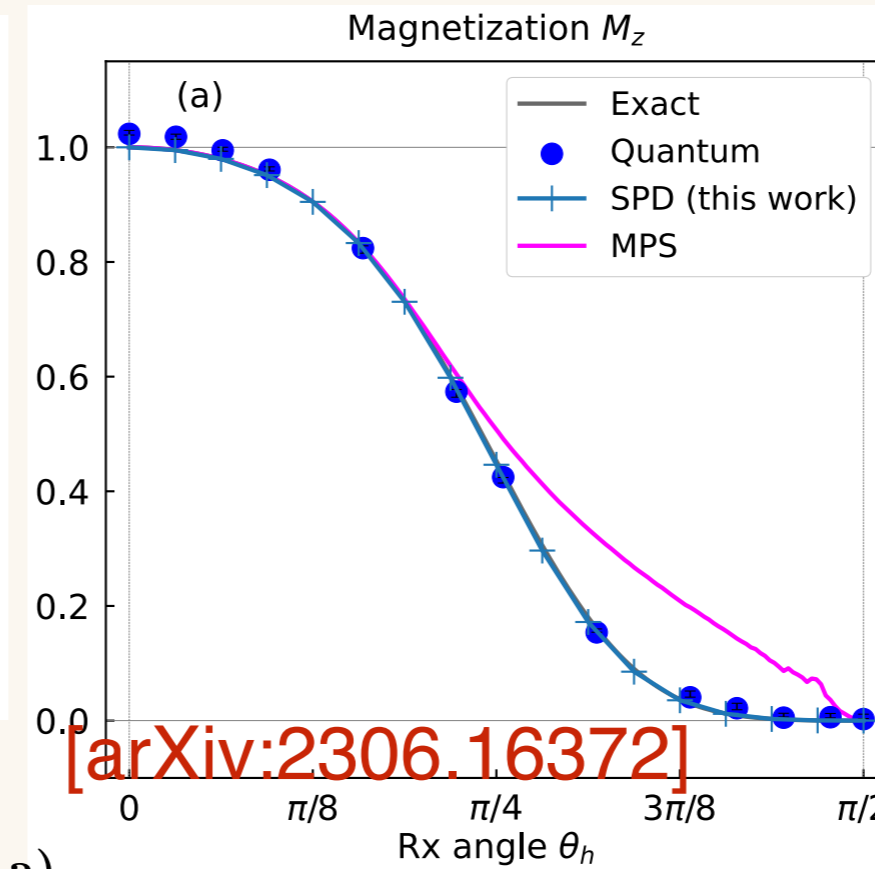
$$O = \langle X_{13,29,31} Y_{9,30} Z_{8,12,17,28,32} \rangle$$



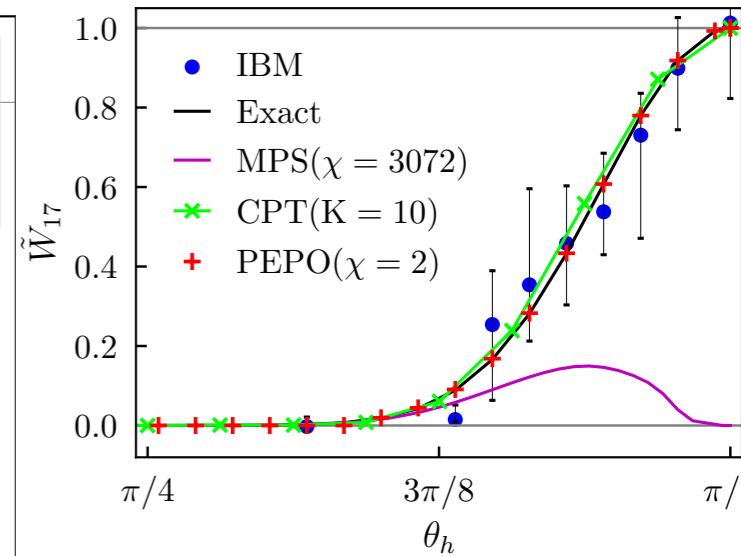
[arXiv:2306.17839]



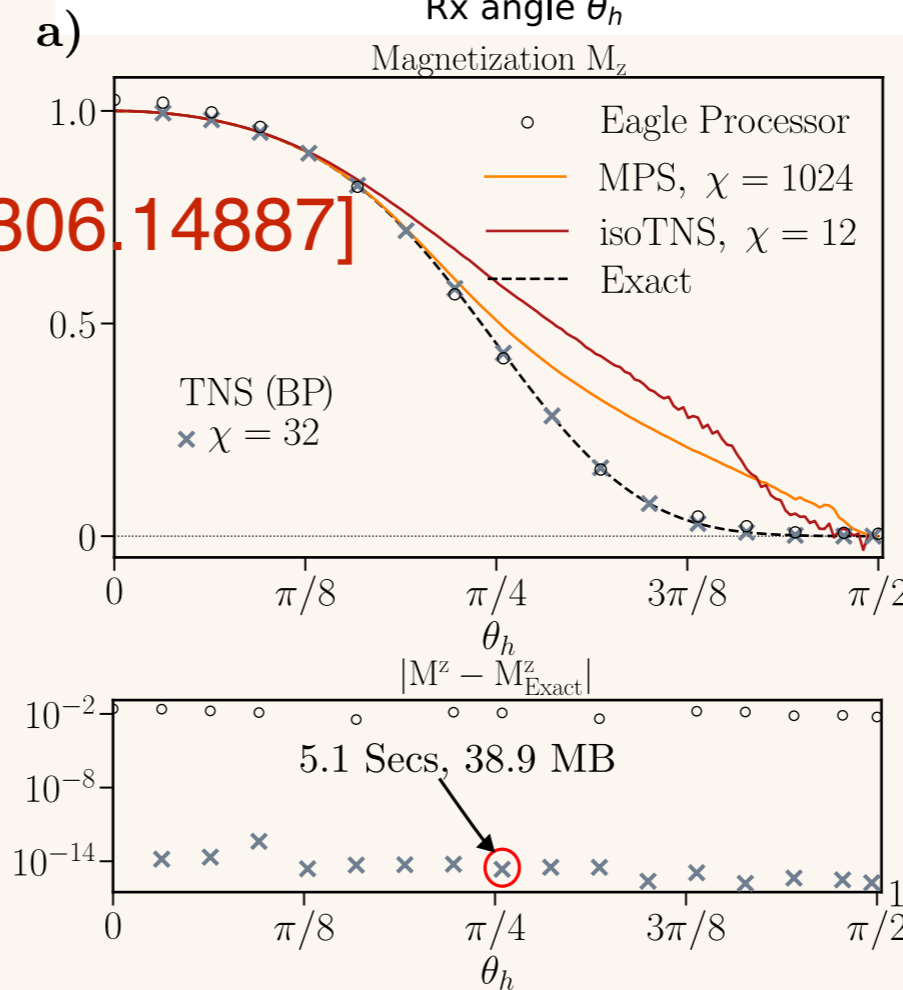
[arXiv:2306.14887]

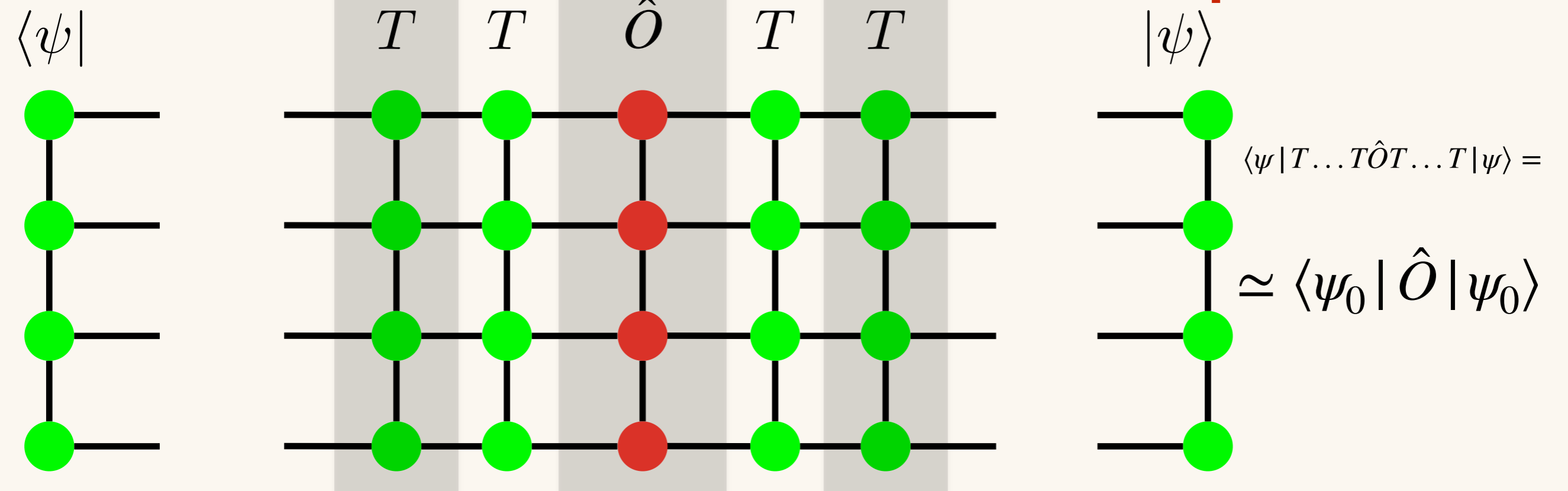


[arXiv:2306.16372]

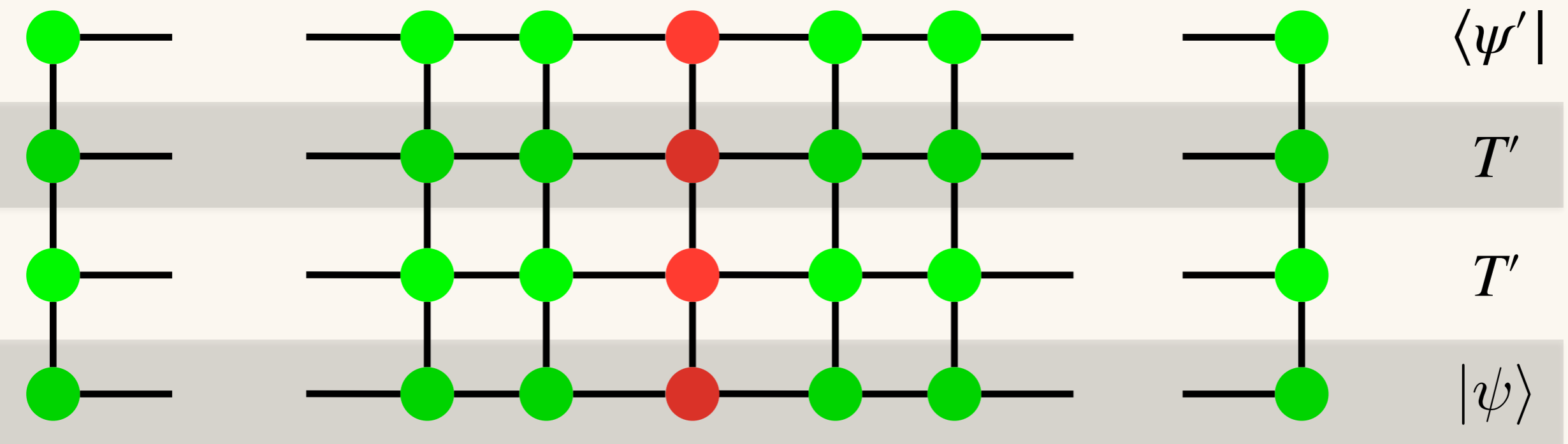


[arXiv:2308.03082]





下で計算するだけで
 精度が向上することがある。



$= \langle \psi' | T' T' | \psi' \rangle \simeq \langle \psi_0 | \hat{O} | \psi_0 \rangle$

初期テンソルの構成方法

● 二次元イジングモデルのテンソルネットワーク表現

◇ どうやってテンソルネットワーク表現を構成するのか？

(e.g.): 二次元イジング(分配関数)

$$Z = \sum_{\sigma} \prod_{x,y} e^{\sigma_{x,y}\sigma_{x+1,y} + \sigma_{x,y}\sigma_{x,y+1}} = \sum_{\sigma} \prod_{x,y} K_{\sigma_{x,y}\sigma_{x+1,y}\sigma_{x,y+1}}$$

Boltzmann
factor

→ テンソル $K_{\sigma_{x,y}\sigma_{x+1,y}\sigma_{x,y+1}}$ はネットワークを構成してはいない。

(添字 $\sigma_{x,y}$ が三つの違うテンソルに含まれている)

$$K_{\sigma_{x,y}\sigma_{x+1,y}\sigma_{x,y+1}}, K_{\sigma_{x-1,y}\sigma_{x,y}\sigma_{x-1,y+1}}, K_{\sigma_{x,y-1}\sigma_{x+1,y-1}\sigma_{x,y}}$$



◇ 従来法: 関数(テイラー)展開を使う ($\sigma^2 = 1$)

● 二次元イジング模型のテンソルネットワーク表現

[H.H.Zhao et al. arXiv:1002.1405]

[Z.Y.Xie et al. arXiv:1201.1144] [Y.Liu et al. arXiv:1307.6543]

◇ 従来法は ? : (Taylor) expansion. and $\sigma^2 = 1$

$$e^{\beta\sigma_{x,y}\sigma_{x+1,y}} = \sum_{l_{x,y}=0}^1 (\cosh(\beta))^{1-l_{x,y}} (\sigma_{x,y}\sigma_{x+1,y}\sinh(\beta))^{l_{x,y}} = \sum_{l_{x,y}=0}^1 W_{\sigma_{x,y}l_{x,y}} W_{\sigma_{x+1,y}l_{x,y}}$$

x-direction: $\sigma \rightarrow l$

y-direction: $\sigma \rightarrow m$



$$W = \begin{pmatrix} \sqrt{\cosh(\beta)}, \sqrt{\sinh(\beta)} \\ \sqrt{\cosh(\beta)}, -\sqrt{\sinh(\beta)} \end{pmatrix}$$

$$Z = \sum_{\sigma} \prod_{x,y} e^{\sigma_{x,y}\sigma_{x+1,y} + \sigma_{x,y}\sigma_{x,y+1}}$$

σ の和を先に取り切る.

$$= \sum_{\sigma, l, m} \prod_{x,y} W_{\sigma_{x,y}l_{x,y}} W_{\sigma_{x,y}l_{x-1,y}} W_{\sigma_{x,y}m_{x,y}} W_{\sigma_{x,y}m_{x,y-1}} = \sum_{l, m} \prod_{x,y} K_{l_{x,y}l_{x-1,y}m_{x,y}m_{x,y-1}}^{(\text{exp})}$$

→ テンソル $K_{l_{x,y}l_{x-1,y}m_{x,y}m_{x,y-1}}^{(\text{exp})}$ がネットワークを構成する。

(添字 $l_{x,y}$ と $m_{x,y}$ が二つのテンソルにだけ含まれる)

$$K_{l_{x,y}l_{x-1,y}m_{x,y}m_{x,y-1}}^{(\text{exp})}, K_{l_{x+1,y}l_{x,y}m_{x+1,y}m_{x+1,y-1}}^{(\text{exp})}$$

$$\sigma_{x,y} \rightarrow \{l_{x,y}, m_{x,y}\}$$

● 展開による構成手法

- ◇ 条件(e.g. $\sigma^2 = 1$)がない時、展開による構成は？

$$F(\sigma_{x,y}\sigma_{x+1,y}) = \sum_{k_{x,y}=0}^{\infty} C^{(k_{x,y})}(\sigma_{x,y}\sigma_{x+1,y})^{k_{x,y}}$$

$$\sum_{\sigma,k} \prod_{x,y=1}^N (\sigma_{x,y}\sigma_{x+1,y})^{k_{x,y}} = \sum_{\sigma,k} \prod_{x,y=1}^N (\sigma_{x,y}^{k_{x,y}} \sigma_{x,y}^{k_{x-1,y}}) = \sum_k \prod_{x,y=1}^N \left(\sum_{\sigma} \sigma_{x,y} \right)^{k_{x,y}+k_{x-1,y}} = \sum_k \prod_{x,y=1}^N f(k_{x,y}, k_{x-1,y})$$

→ 変数変換: $\sigma_{x,y} \rightarrow k_{x,y}$

→ 少なくとも $k_{x,y}$ の有限性が明示的じゃなくなる。

→ そもそも $e^{\beta\sigma_{x,y}\sigma_{x+1,y}}$ より複雑な関数な時は、展開も複雑化

→ どの展開を使うかも自由（直行関数展開なども使える）

- ◇ 目的や条件に応じて適切な展開を考えないと、表現すら得られないことも（得るのに手間がかかる）。

● 単位行列による構成方法

[K. Nakayama, M.Schneider arXiv:2407.14226]

◇ ポイント: 展開を使う場合は $\sigma^2 = 1$ のような条件を利用

→ 問題: ボルツマン因子が複雑化していった時、

ネットワーク表現を機械的に構成できると嬉しい。

◇ 提案手法: 単位行列による添字のシフト

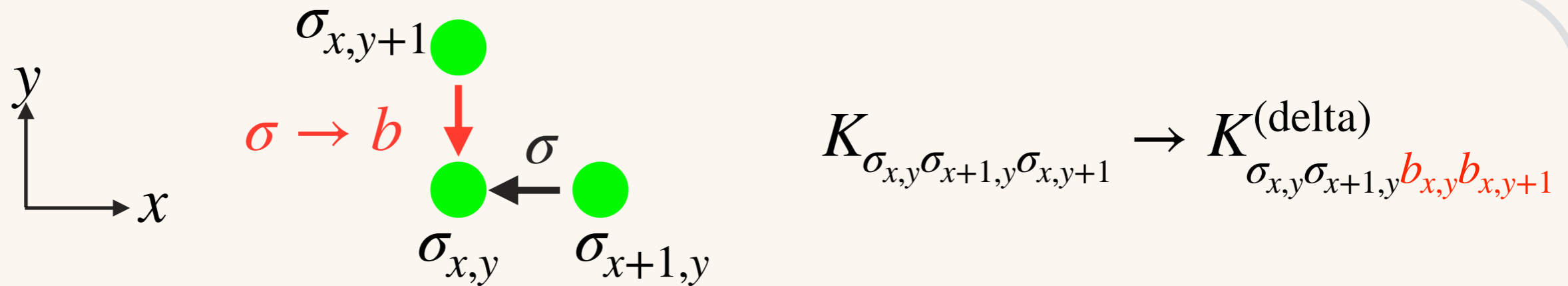
(e.g.): 二次元イジング模型 (periodic b.c.)

$$\begin{aligned} Z &= \sum_{\sigma} \prod_{x,y} K_{\sigma_{x,y}\sigma_{x+1,y}\sigma_{x,y+1}} = \sum_{\sigma,b} \prod_{x,y} K_{\sigma_{x,y}\sigma_{x+1,y}b_{x,y+1}} \delta_{b_{x,y+1}\sigma_{x,y+1}} = \sum_{\sigma,b} \prod_{x,y} K_{\sigma_{x,y}\sigma_{x+1,y}b_{x,y+1}} \delta_{b_{x,y}\sigma_{x,y}} \\ &= \sum_{\sigma,b} \prod_{x,y} K^{(\text{delta})}_{\sigma_{x,y}\sigma_{x+1,y}b_{x,y}b_{x,y+1}} \end{aligned}$$

◆ index shift ($y + 1 \rightarrow y$) by δ

→ ネットワークを構成している $K^{(\text{delta})}_{\sigma_{x,y}\sigma_{x+1,y}b_{x,y}b_{x,y+1}} \equiv K_{\sigma_{x,y}\sigma_{x+1,y}b_{x,y+1}} \times \delta_{b_{x,y}\sigma_{x,y}}$

● 模式図



(1): 点 \leftrightarrow 元々の添字 $\{\sigma_{x,y}, \sigma_{x+1,y}, \sigma_{x,y+1}\}$

(2): すべての点を矢印で繋ぐ

(3): 矢印 (σ 以外) \leftrightarrow 単位行列 δ による添字シフト.



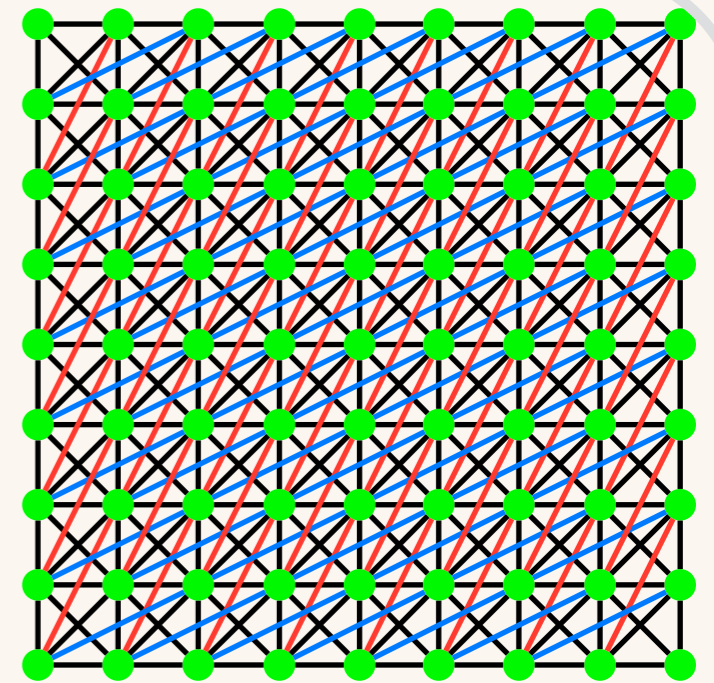
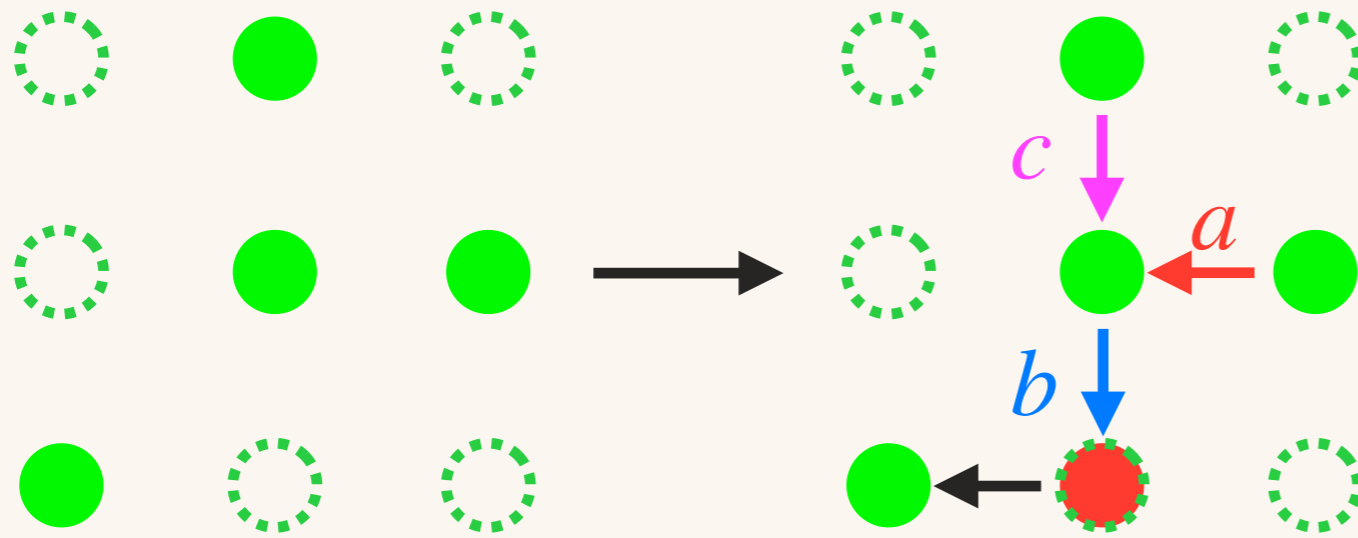
矢印 \leftrightarrow 新しい添字 $\{\sigma, a\}$.

→ この構成方法は、

条件やボルツマン因子の詳細の情報を一切必要としない。

$$\text{(e.g. } \sigma^2 = 1, K_{\sigma_{x,y}\sigma_{x+1,y}\sigma_{x,y+1}} = e^{\sigma_{x,y}\sigma_{x+1,y} + \sigma_{x,y}\sigma_{x,y+1}} \text{)}_{33}$$

● 一般化の一例: J1-J2+ α



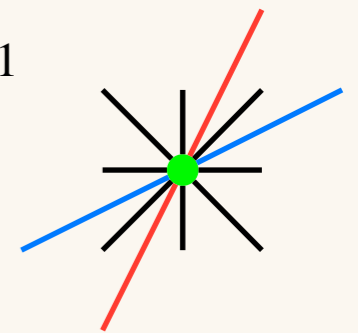
$$K_{\sigma_{x,y}\sigma_{x+1,y+1}\sigma_{x+2,y+1}\sigma_{x+1,y+2}} \rightarrow K''_{[\sigma a]_{x,y}[\sigma a]_{x+1,y}[bc]_{x,y}[bc]_{x,y+1}}$$

$$e^{h(\sigma_{x,y}+\sigma_{x+1,y+1}+\sigma_{x+2,y+1}+\sigma_{x+1,y+2})}$$

$$e^{J_1(\sigma_{x+1,y+1}\sigma_{x+2,y+1}+\sigma_{x+1,y+1}\sigma_{x+1,y+2})} e^{J_2(\sigma_{x,y}\sigma_{x+1,y+1}+\sigma_{x+2,y+1}\sigma_{x+1,y+2})} e^{g_1\sigma_{x,y}\sigma_{x+2,y+1}+g_2\sigma_{x,y}\sigma_{x+1,y+2}}$$

$$e^{t_1\sigma_{x,y}\sigma_{x+2,y+1}\sigma_{x+1,y+2}+t_2\sigma_{x,y}\sigma_{x+1,y+1}\sigma_{x+2,y+1}+t_3\sigma_{x,y}\sigma_{x+1,y+1}\sigma_{x+1,y+2}+t_4\sigma_{x+1,y+1}\sigma_{x+2,y+1}\sigma_{x+1,y+2}}$$

$$e^{e_1\sigma_{x,y}\sigma_{x+1,y+1}\sigma_{x+2,y+1}\sigma_{x+1,y+2}}$$



→ 追加した赤点はSteiner点と呼ぶ。

→ この模式図はRectilinear Steiner木問題になっている。

(一般化最小木問題。似た問題で巡回セールスマン問題が有名)。

→ K'' は $4 \times 4 \times 4 \times 4$ だけの添字で表現される。

● J1-J2+α: Frustrated system with 12 parameters.

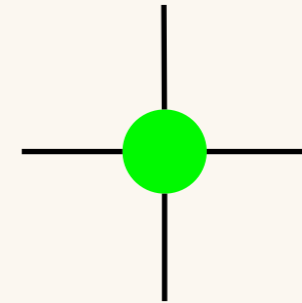
1-spin

$$e^{h(\sigma_{x,y} + \sigma_{x+1,y+1} + \sigma_{x+2,y+1} + \sigma_{x+1,y+2})}$$



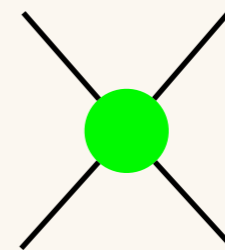
2-spin (J1)

$$e^{J_1^{(x)} \sigma_{x+1,y+1} \sigma_{x+2,y+1} + J_1^{(y)} \sigma_{x+1,y+1} \sigma_{x+1,y+2}}$$



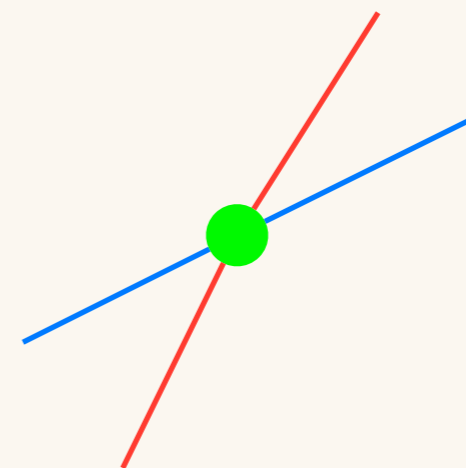
2-spin (J2)

$$e^{J_2^{(a)} \sigma_{x,y} \sigma_{x+1,y+1} + J_2^{(b)} \sigma_{x+2,y+1} \sigma_{x+1,y+2}}$$



2-spin (g1,g2)

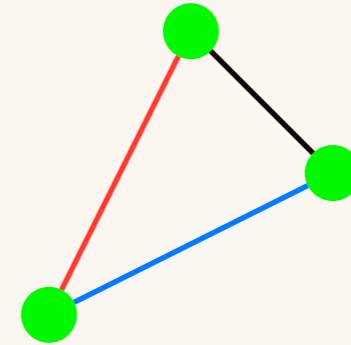
$$e^{g_1 \sigma_{x,y} \sigma_{x+2,y+1} + g_2 \sigma_{x,y} \sigma_{x+1,y+2}}$$



● J1-J2+ α : Frustrated system with 12 parameters.

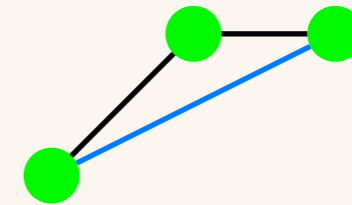
3-spin t1

$$e^{t_1 \sigma_{x,y} \sigma_{x+2,y+1} \sigma_{x+1,y+2}}$$



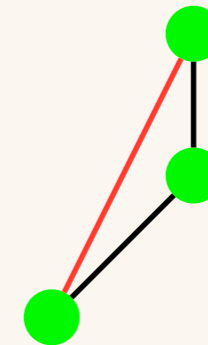
3-spin t2

$$e^{t_2 \sigma_{x,y} \sigma_{x+1,y+1} \sigma_{x+2,y+1}}$$



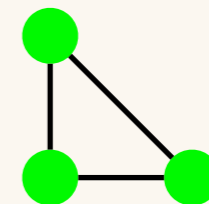
3-spin t3

$$e^{t_3 \sigma_{x,y} \sigma_{x+1,y+1} \sigma_{x+1,y+2}}$$



3-spin t4

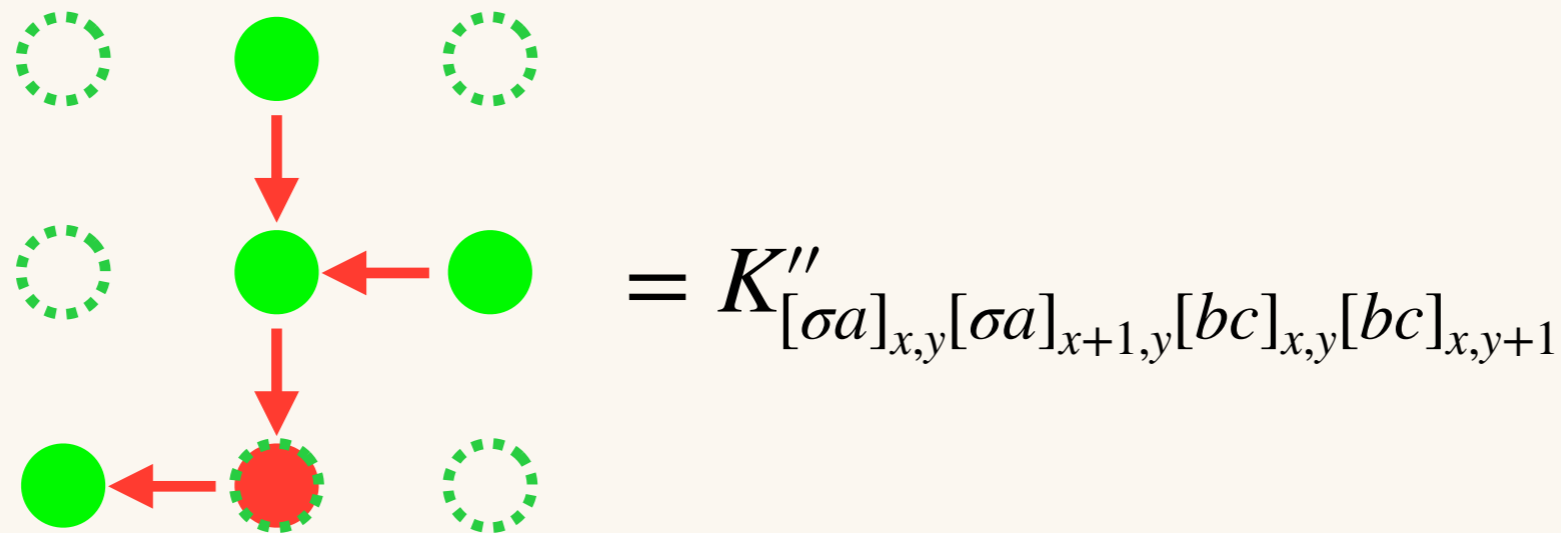
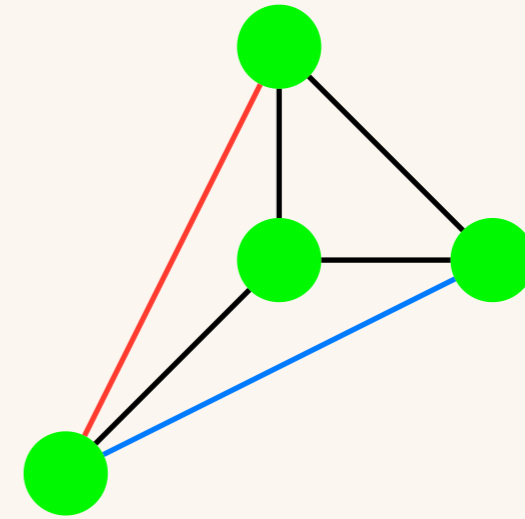
$$e^{t_4 \sigma_{x+1,y+1} \sigma_{x+2,y+1} \sigma_{x+1,y+2}}$$



● J1-J2+ α : Frustrated system with 12 parameters.

4-spin

$$e^{e_1 \sigma_{x,y} \sigma_{x+1,y+1} \sigma_{x+2,y+1} \sigma_{x+1,y+2}}$$



◇ 相互作用の種類 (パラメータの数):

$$\left(\sum_{k=2}^4 {}_4C_k \right) + 1 = 2^4 - 4 = 12$$

→ K'' は $4 \times 4 \times 4 \times 4$ だけの添字で表現される。³⁷

NOTE:Steiner木問題

● Note: Steiner木問題.

◇ Steiner木問題:

最短の線分（木）の集合で全ての点を繋ぐ（最小全域木）

+ 新しい点を加えてもいい

◇ Rectilinear Steiner木問題: (NP-complete)

格子上的Steiner木問題

[M. Hanan, SIAM Appl. Math, 14, 2, p255, (1966)]

... 格子上的にのみ点、リンク上にのみ線分。

→ テンソルの添字の大きさが線分の長さに関連 $D_{ini}^{(\text{Road length})}$

→ 長距離相互作用では添字の大きさが大きくなっていく。

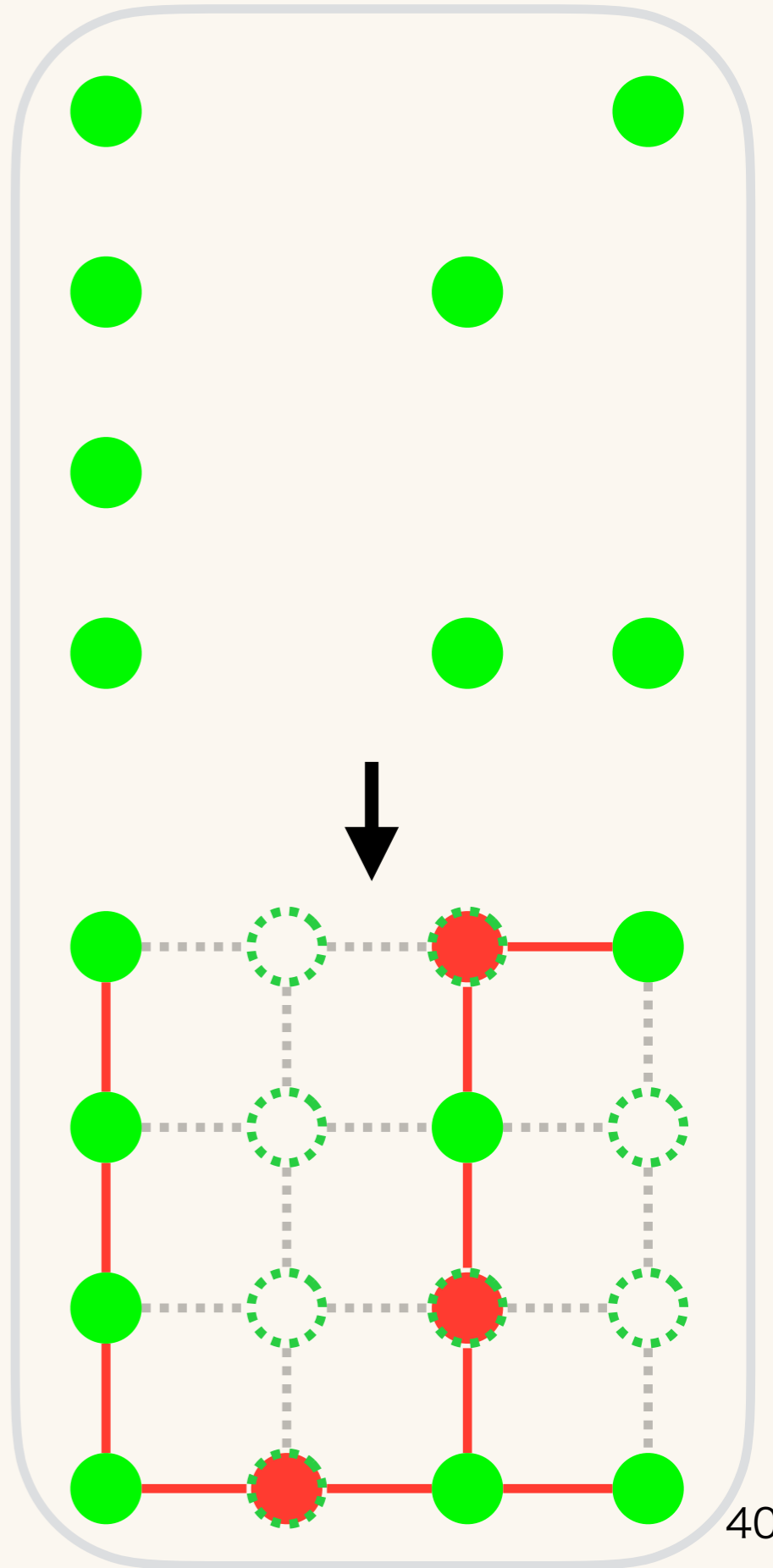
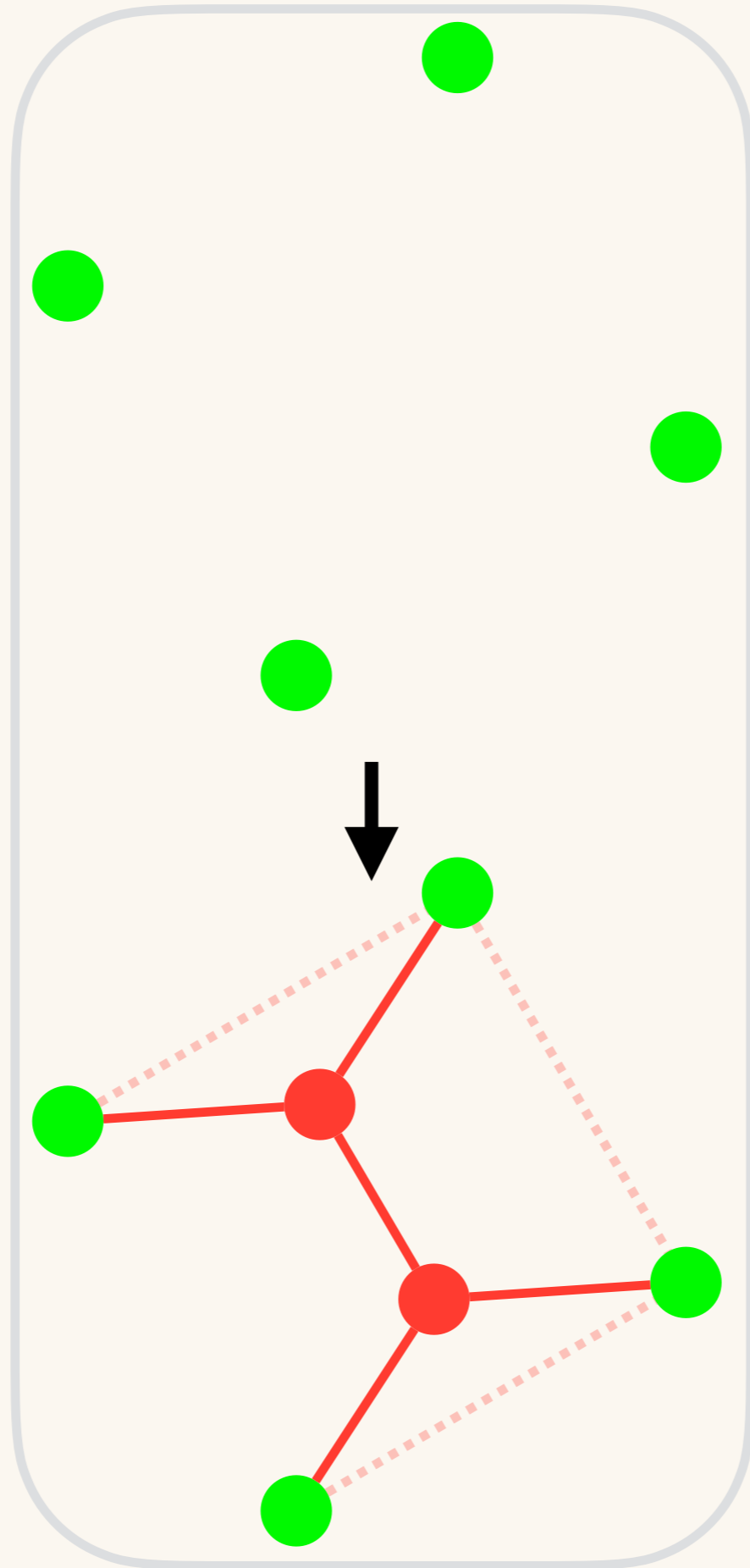
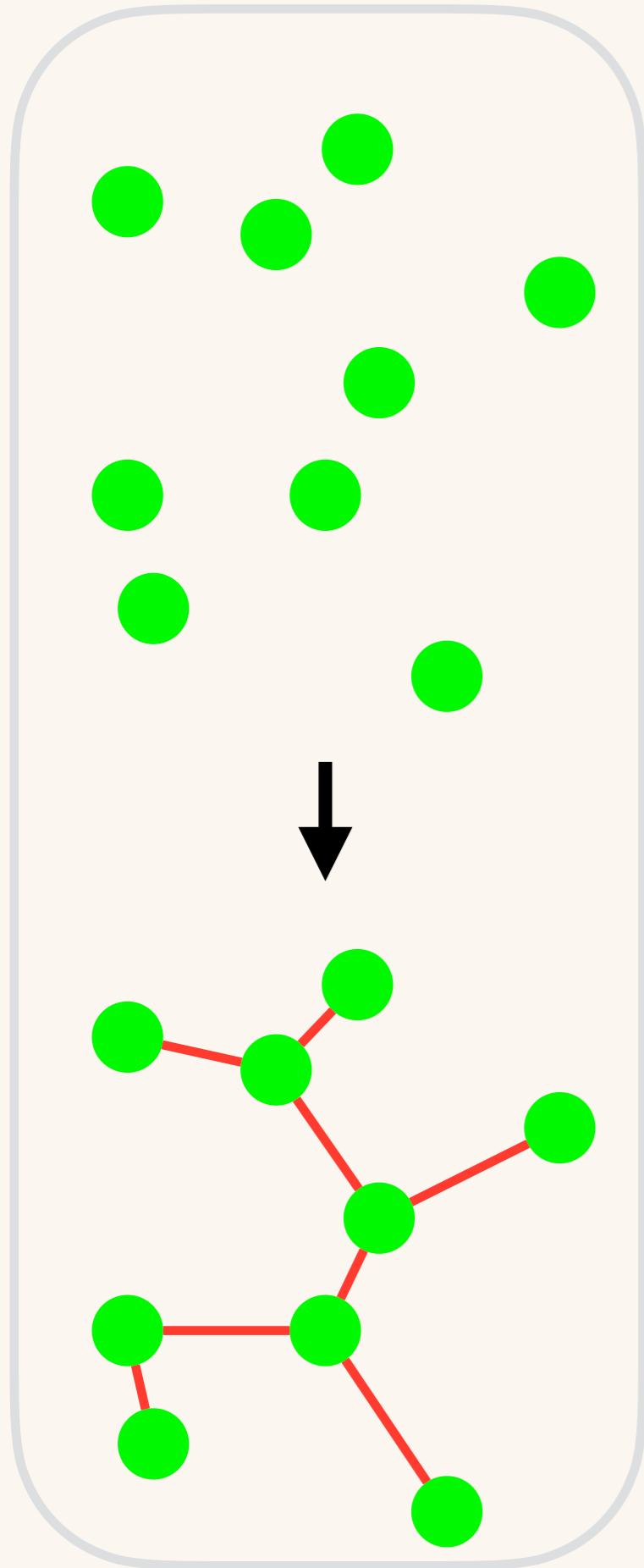
→ この分類だとゲージ作用（プラケット作用）なども

最近接相互作用ではないが、この程度なら添字は小さい。

最小全域木

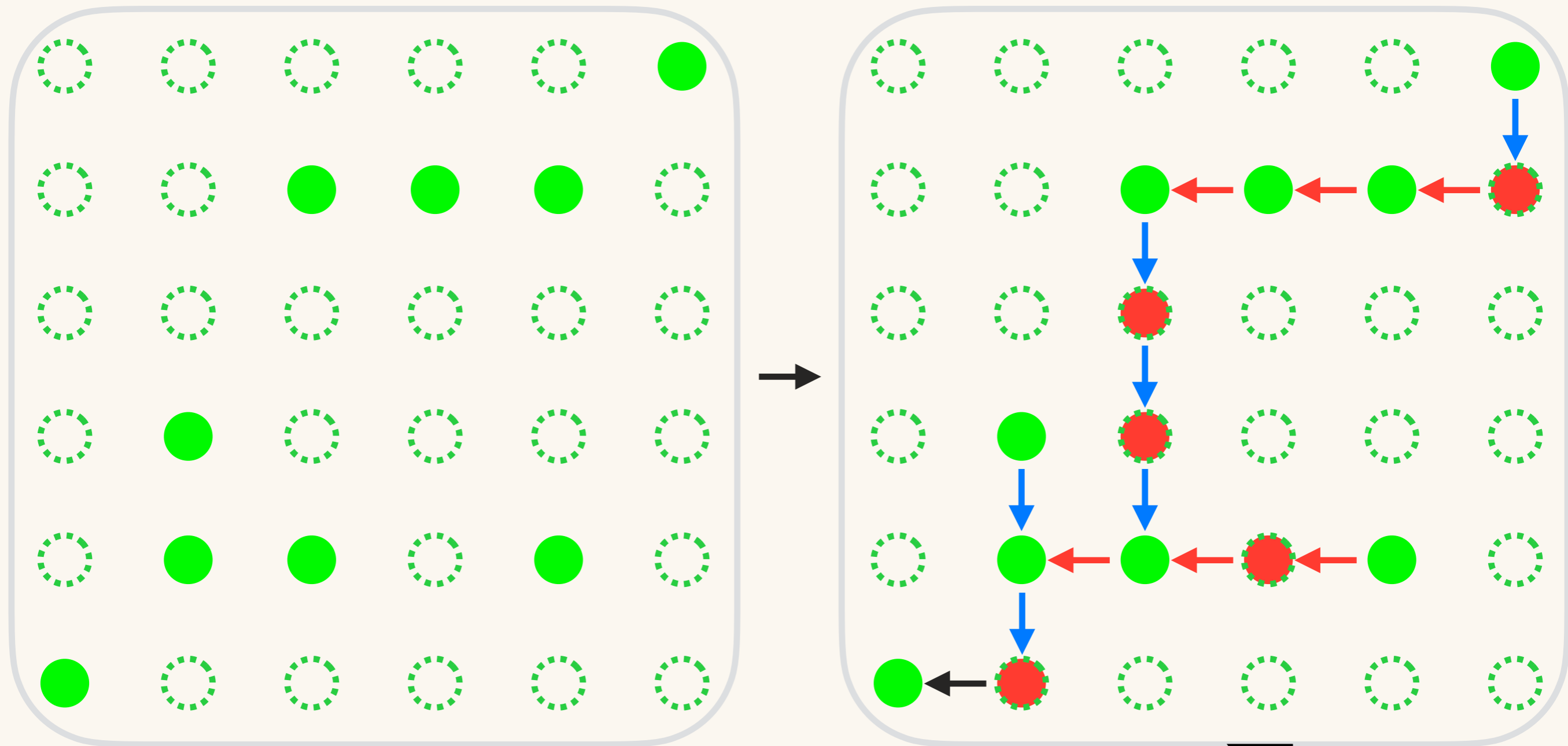
Steiner木

Rectilinear Steiner



● Note: (Rectilinear) Steiner木問題

- ◇ Rectilinear Steiner tree problem: Find shortest line segment on the lattice between dots with additional dots.

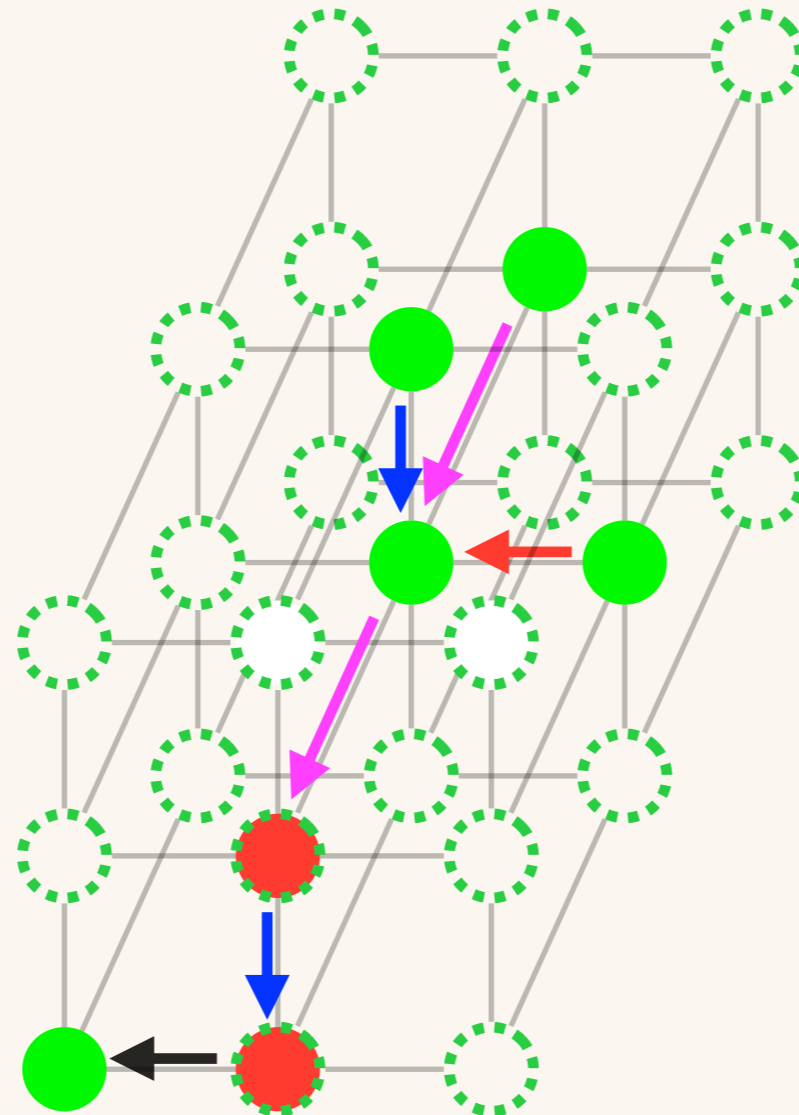


→ Num. of interaction (parameters) order $\sim O(\sum 9C_k = 2^9)$

→ K'' has $2^6 \times 2^6 \times 2^7 \times 2^7$ indices.

● (e.g.) J1-J2+α in 3dim

◇ 三次元でも要領は同じ



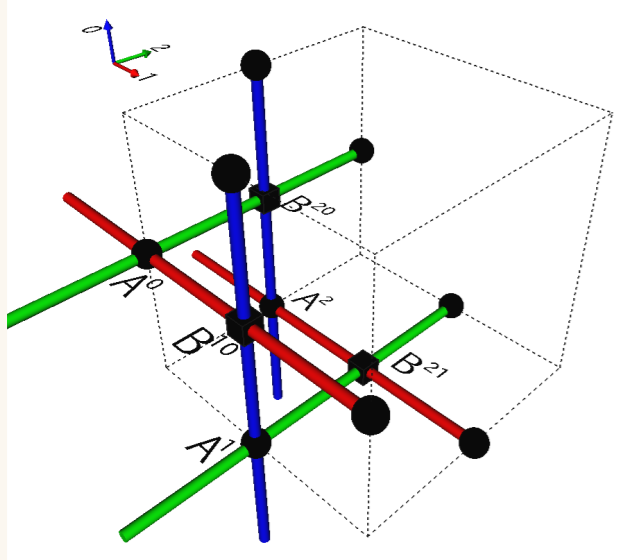
$$K^{(3d)} \sigma_{x,y,z} \sigma_{x+1,y+1,z+1} \sigma_{x+2,y+1,z+1} \sigma_{x+1,y+2,z+1} \sigma_{x+1,y+1,z+2}$$

$$\rightarrow T^{(3d)} \sigma_{x,y,z} \sigma_{x+1,y,z} a_{x,y,z} a_{x+1,y,z} b_{x,y,z} b_{x,y+1,z} c_{x,y,z} c_{x,y+1,z} d_{x,y,z} d_{x,y,z+1} e_{x,y,z} e_{x,y,z+1}$$

● 三次元 Z_2 ゲージ理論のテンソル表現

[Y.Liu et al. arXiv:1307.6543] [Y.Kuramashi, Y.Yoshimura, arXiv:1808.08025]

◇ Common method: (Taylor) expansion. and $\sigma^2 = 1$



$$Z = 2^{-3V} \sum_{\sigma} \prod_{n, \mu > \nu} e^{-\beta \sigma_{n, \mu} \sigma_{n+\hat{\mu}, \nu} \sigma_{n+\hat{\nu}, \mu} \sigma_{n, \nu}}$$

$$e^{-\beta \sigma_{n, \mu} \sigma_{n+\hat{\mu}, \nu} \sigma_{n+\hat{\nu}, \mu} \sigma_{n, \nu}} = \cosh \beta \sum_{p=0}^1 (\tanh \beta)^p (\sigma_{n, \mu} \sigma_{n+\hat{\mu}, \nu} \sigma_{n+\hat{\nu}, \mu} \sigma_{n, \nu})^p$$

$$A_{pqrs} = \text{mod}(1 + p + q + r + s, 2)$$

$$B_{pqrs} = (\tanh \beta)^{(p+q+r+s)/4} \delta_{pq} \delta_{pr} \delta_{rs}$$

$$Z = \sum_{g, h, i, j, k, l} \prod_n T_{[gh]_{x,y,z} [gh]_{x+1,y,z} [ij]_{x,y,z} [ij]_{x,y+1,z} [kl]_{x,y,z} [kl]_{x,y,z+1}}^{(\text{exp})}$$

$$T_{[xX][x'X'] [yY][y'Y'] [zZ][z'Z']}^{(\text{exp})} = (\cosh \beta)^3 \sum_{a, b, c, d, e, f} A_{cyZe} A_{fzxb} A_{dYXa} B_{bx'y'c} B_{aX'Z'e} B_{fz'Y'd}$$

◇ 一般にはどんな展開でもいい。

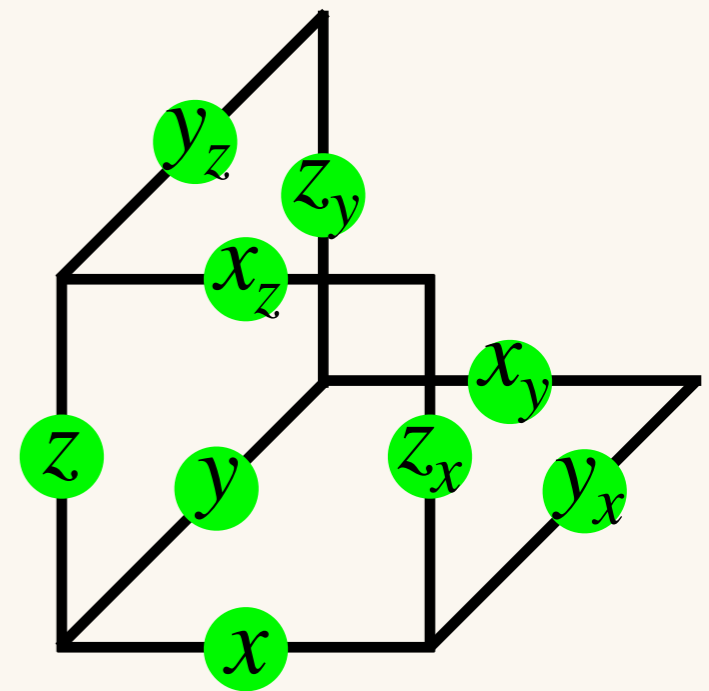
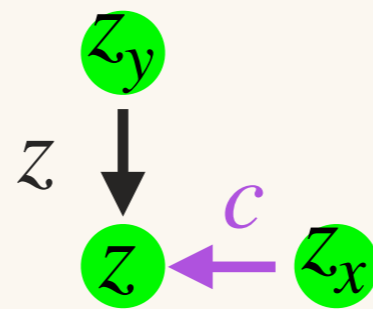
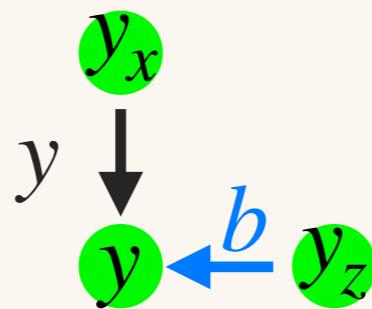
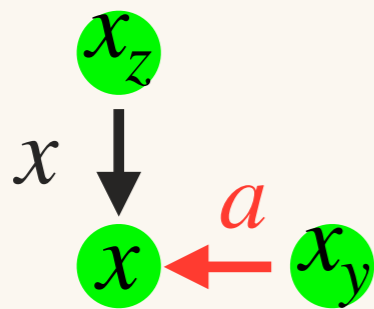
(Character for gauge theory, Orthogonal function, Taylor...) 43

● 三次元 Z_2 ゲージ理論のテンソル表現

◇ 添字のシフトを単位行列で。

$$Z = 2^{-3V} \sum_{\sigma} \prod_{n, \mu > \nu} e^{-\beta \sigma_{n, \mu} \sigma_{n+\hat{\mu}, \nu} \sigma_{n+\hat{\nu}, \mu} \sigma_{n, \nu}} = \sum_{\sigma} \prod_{n, \mu > \nu} e^{-\beta (x x_{\hat{y}} y y_{\hat{x}} + x x_{\hat{z}} z z_{\hat{x}} + y y_{\hat{z}} z z_{\hat{y}})} / 8$$

→ x, y, z 方向の添字は互いに独立。

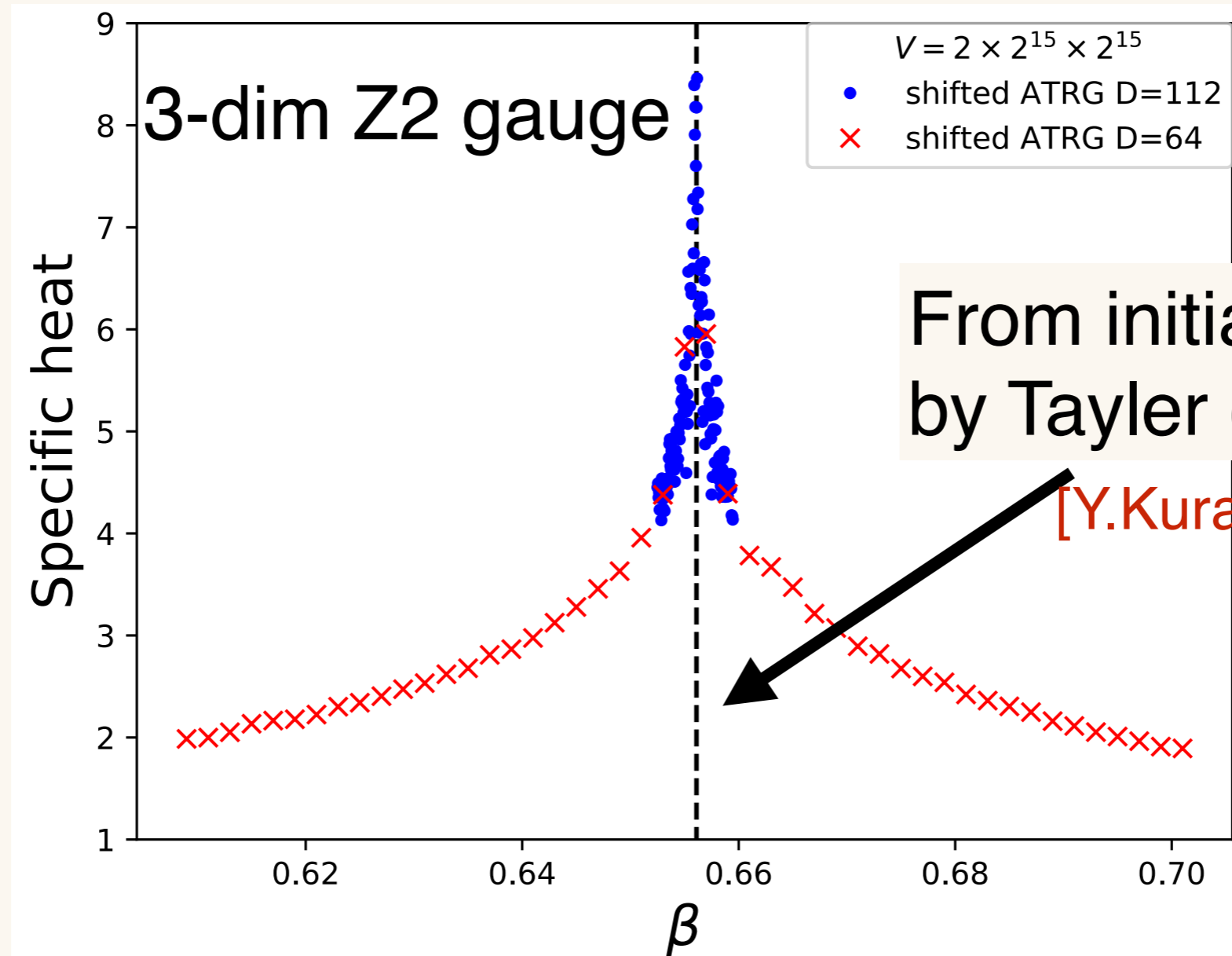


$$K_{x x_y x_z y y_z y_x z z_x z_y} \rightarrow K''_{[x b] [x b]_z [y c] [y c]_x [z a] [z a]_y}$$

→ K'' has only $4 \times 4 \times 4 \times 4 \times 4 \times 4$ indices.

● 三次元 Z_2 ゲージ理論の相転移

- ◇ Numerical calc. by (modified) ATRG + impurity method



From initial tensor
by Taylor expansion

[Y. Kuramashi et al.
arXiv:1808.08025]

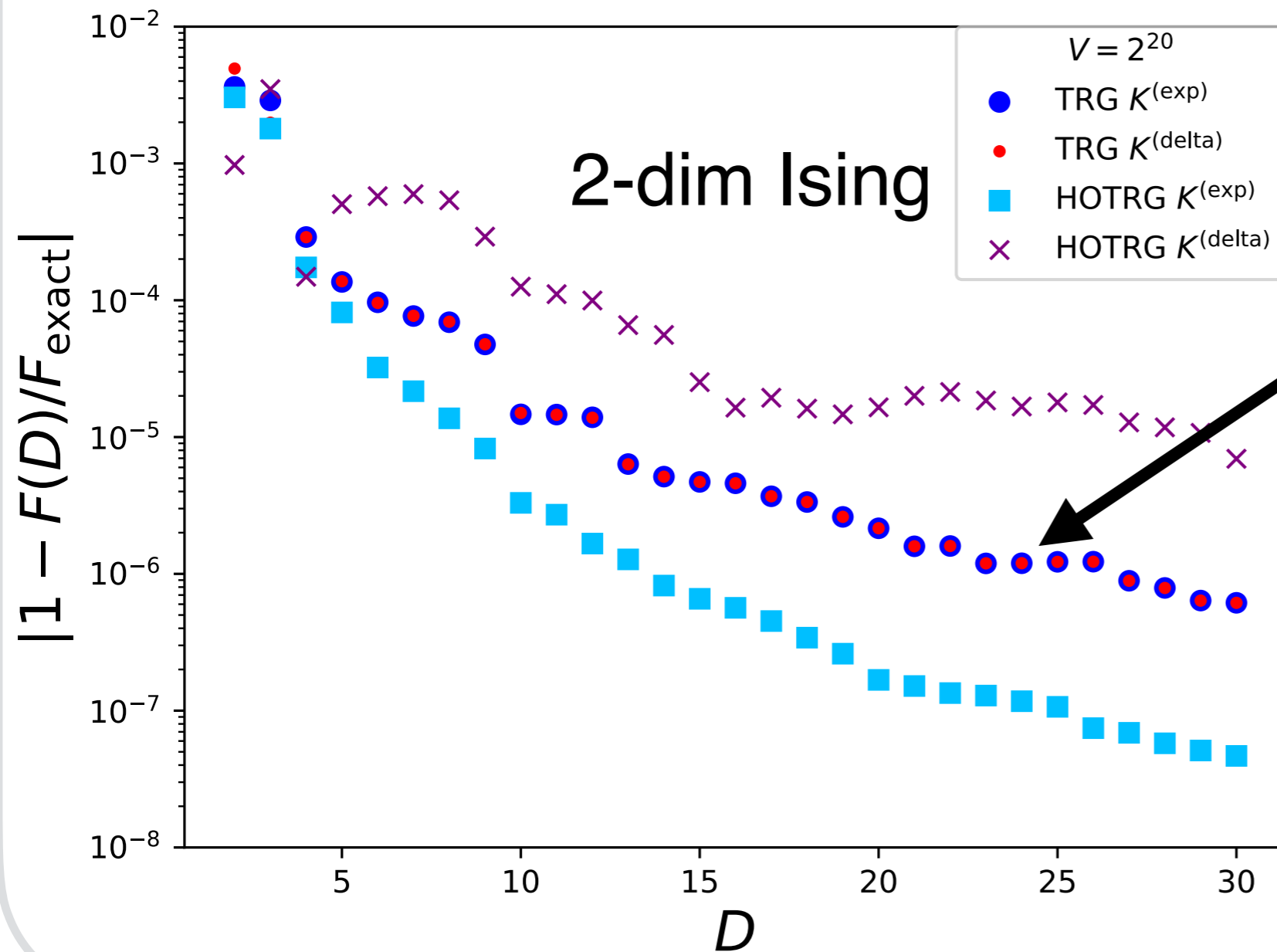
→ Our method produces correct result for critical temperature.

初期テンソル依存性

● Initial tensor dependence

◇ 簡単にネットワークを見つけられるようになった！

→ でも精度が犠牲になってたりしないか？



TRGの精度:
初期テンソルに
依存しない。

HOTRGの精度:
初期テンソルに
依存する。

→ この依存性を消す方法が経験的に見つけれられる。

● Boundary TRG

[S. Iino et al. arXiv:1905.02351]

[K. N., M. Schneider arXiv:2407.14226]

- ◇ Original HOTRG: U か V をノルムの大きさで選ぶ。

$$C_U = \left\| \begin{array}{c} K \\ \text{---} \\ K \\ \text{---} \\ | \end{array} \right. - \left. \begin{array}{c} U^\dagger(\text{HOTRG}) \\ \text{---} \\ U(\text{HOTRG}) \\ \text{---} \\ | \end{array} \right\|^2 \quad C_V = \left\| \begin{array}{c} K \\ \text{---} \\ K \\ \text{---} \\ | \end{array} \right. - \left. \begin{array}{c} V(\text{HOTRG}) \\ \text{---} \\ V^\dagger(\text{HOTRG}) \\ \text{---} \\ | \end{array} \right\|^2$$

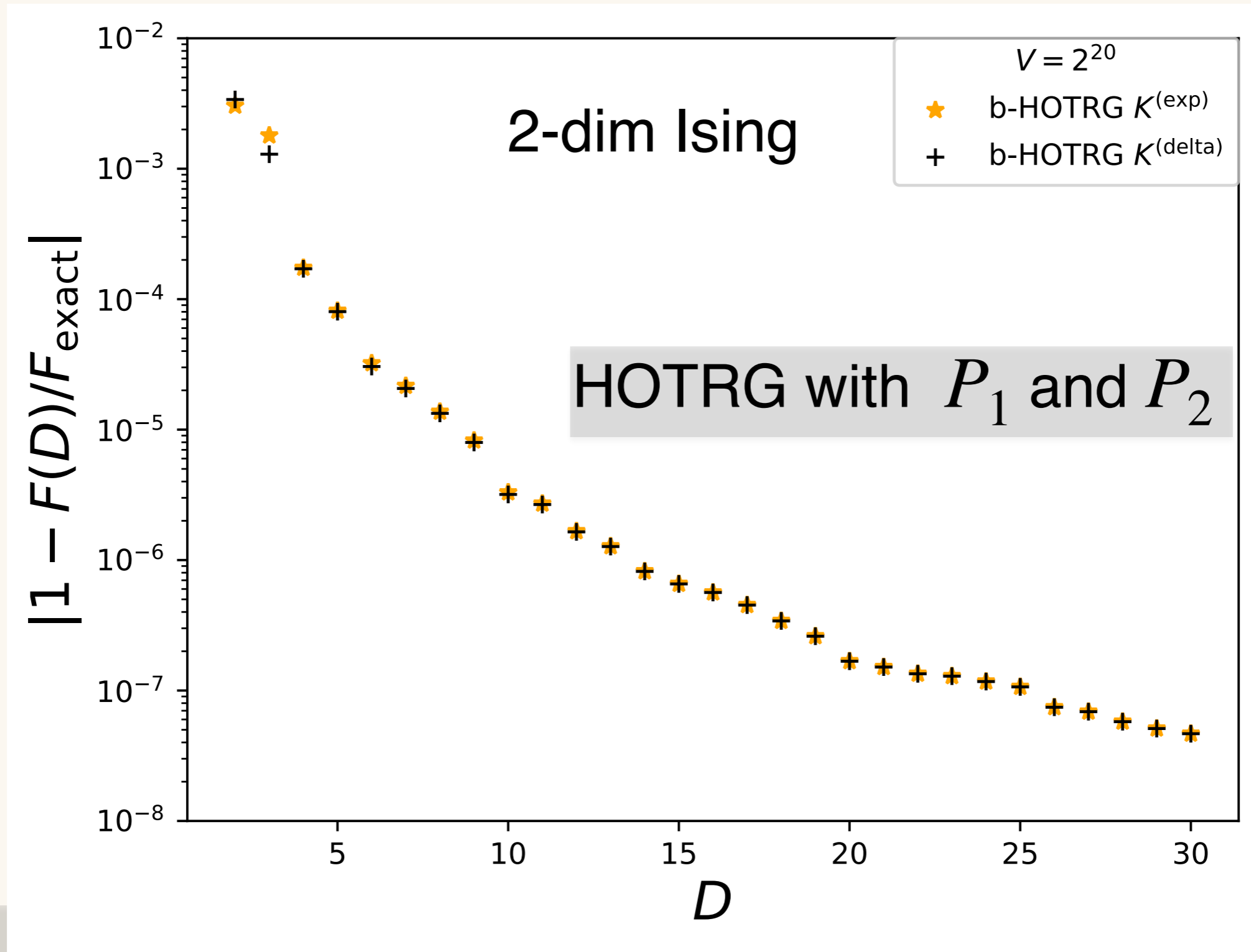
- ◇ HOTRG with boundary TRG:
 P_1, P_2 を U と V から作る。

$$C_{P_1, P_2} = \left\| \begin{array}{c} K \\ \text{---} \\ K \\ \text{---} \\ | \end{array} \right. - \left. \begin{array}{c} P_1^{(\text{bHOTRG})} \\ \text{---} \\ P_2^{(\text{bHOTRG})} \\ \text{---} \\ | \end{array} \right\|^2$$

→ U か V を選ぶ方法から P_1 と P_2 を構成することは、

他のTRGの手法でも問題なく可能。 48

● Boundary HOTRG



→ HOTRG with P_1 and P_2 は初期テンソルに依存しない。

Boundary TRG for ATRG and MDTRG

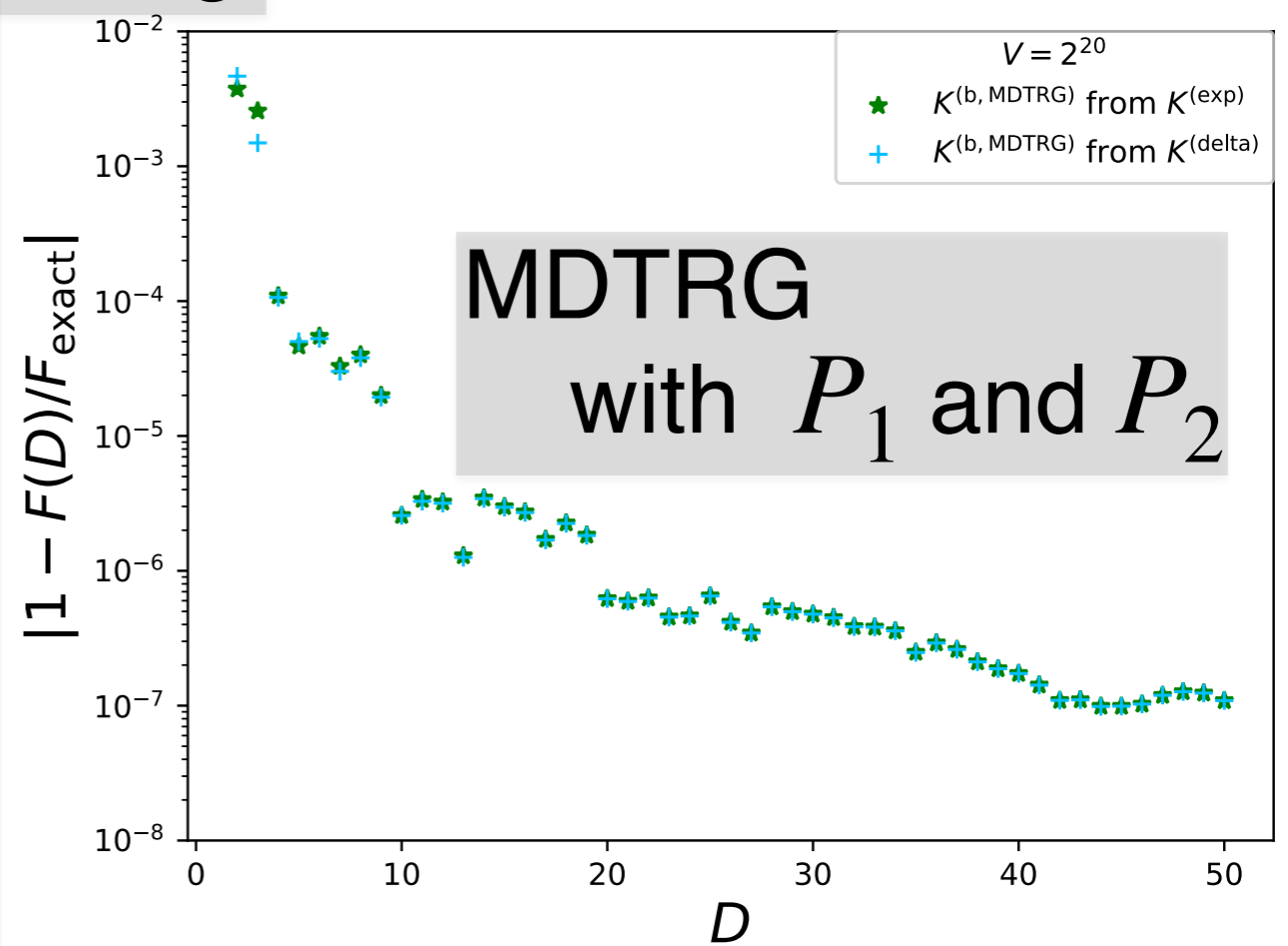
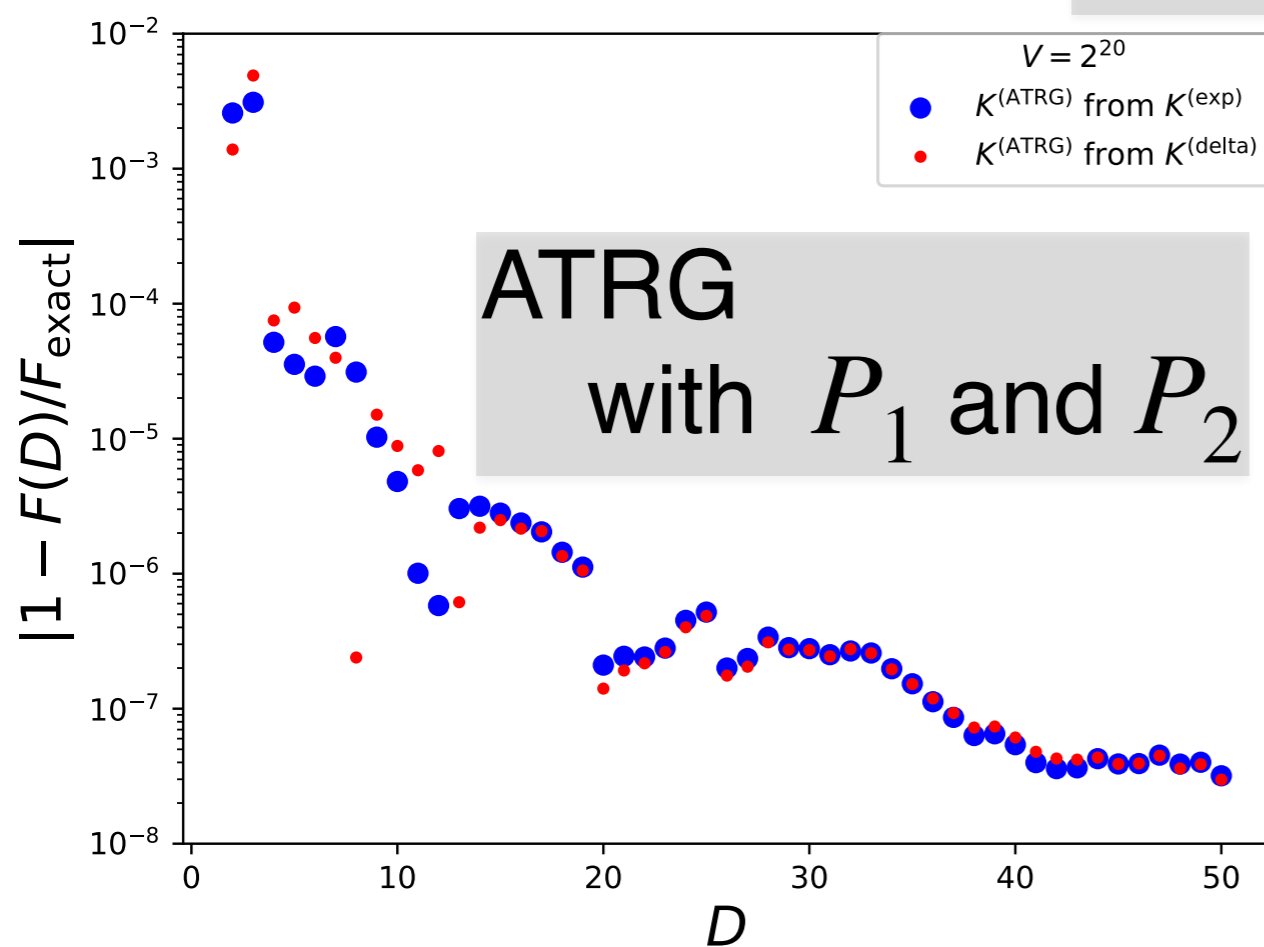
[D. Adachi et al. arXiv:1906.02007]

[D. Kadoh, K.N. arXiv:1912.02414][K.N. arXiv:2307.14191]

- ◇ Many method (ATRG, TriadTRG, MDTRG) can use isometry (U or V) or P_1 and P_2 .

[K. N., M.Schneider arXiv:2407.14226]

2-dim Ising



→ TRG methods with P_1 and P_2 are initial tensor independent.

→ Our construction can also produce compatible result.

● まとめ

[main]

- ◇ ボルツマン因子表現があれば、簡単にテンソルネットワーク表現に変換できる。
- ◇ 初期テンソル依存性はboundary TRG法で消せる。

[details]

- ◇ ネットワーク構成方法はSteiner木問題になっている。
- ◇ 二次元や三次元でテストした。
- ◇ 依存性を消す方法は拡張性が高い。

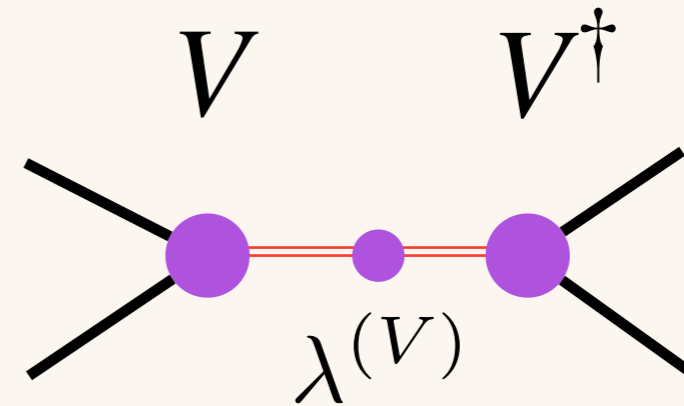
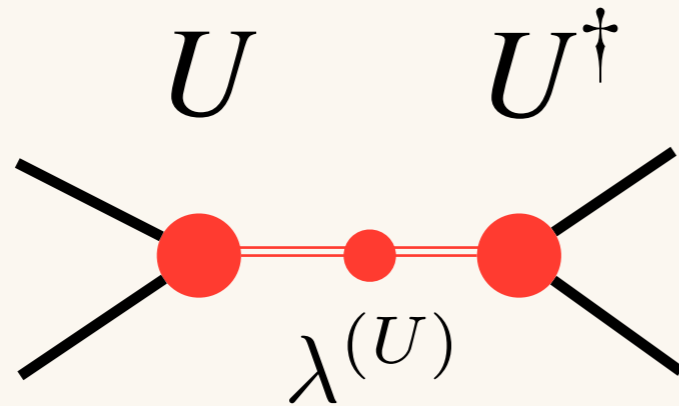
→ この方法が、簡便かつ不利益の少ない選択になり得る。

backup

● boundary HOTRG

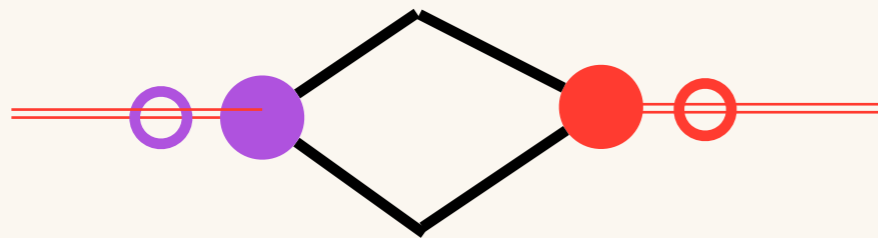
[S. Iino et al 1905.02351]

- ◇ まずは U, V を **厳密に** (打ち切りなしで) 計算する。



- ◇ U, V を両方巻き込みつつ SVD, 打ち切り近似

$$\sqrt{\lambda^{(V)}} V^\dagger U \sqrt{\lambda^{(U)}} = R_1 R_2 \simeq U^{(R)} \Lambda^{(R)} V^{(R)}$$



\simeq



(ルートを取るのは、HOTRGは MM^\dagger のように固有値が二乗されているから。)

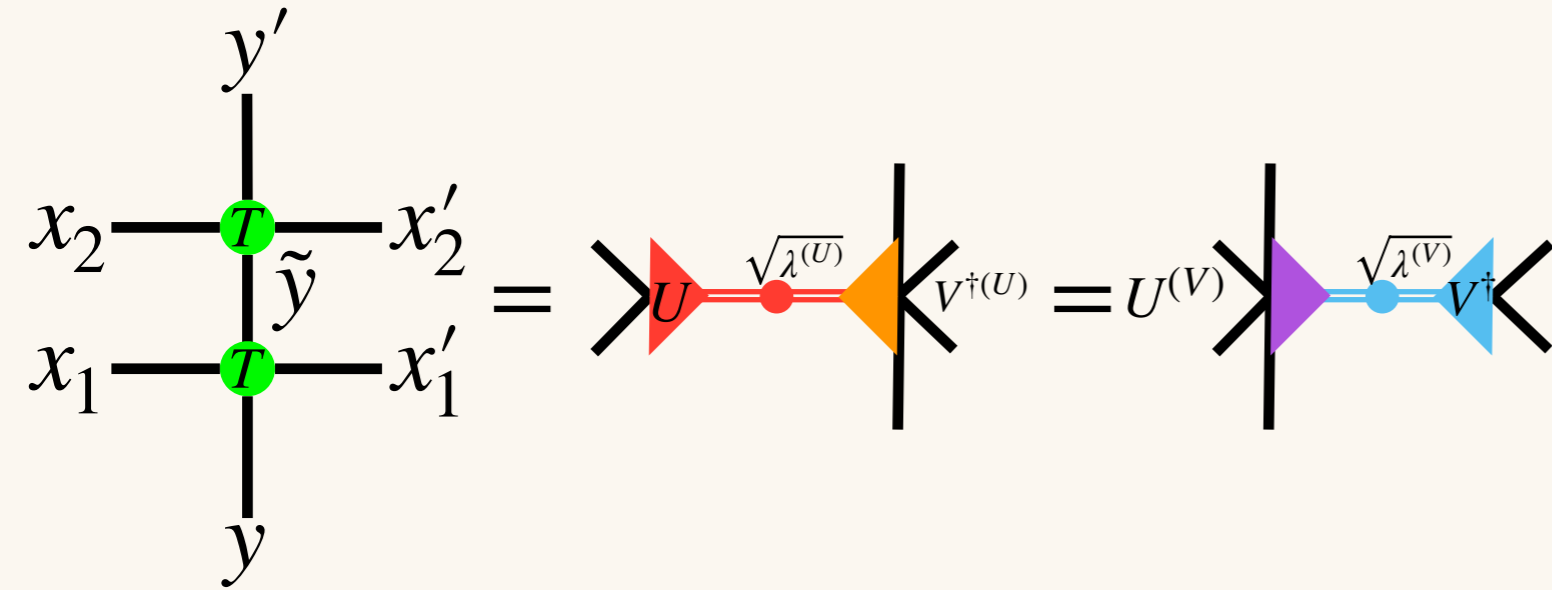
- ◇ P_1, P_2 を定義

$$P_1 = U \sqrt{\lambda^{(U)}} V^{(R)} / \sqrt{\Lambda^{(R)}} = R_2 V^{(R)} / \sqrt{\Lambda^{(R)}}$$

$$P_2 = \left(V \sqrt{\lambda^{(V)}} U^{(R)} / \sqrt{\Lambda^{(R)}} \right)^\dagger = \left(R_1^\dagger U^{(R)} / \sqrt{\Lambda^{(R)}} \right)^\dagger$$

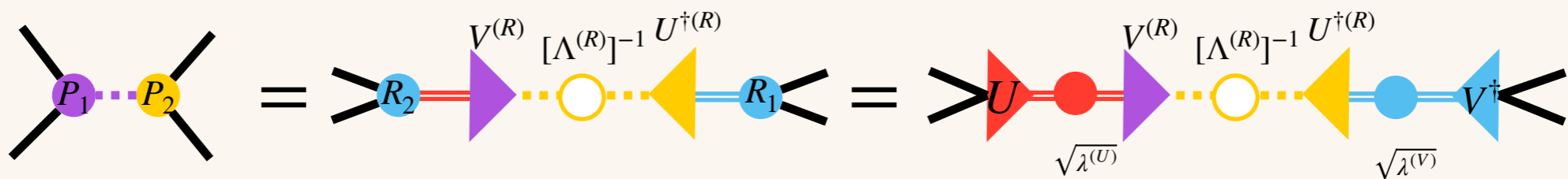
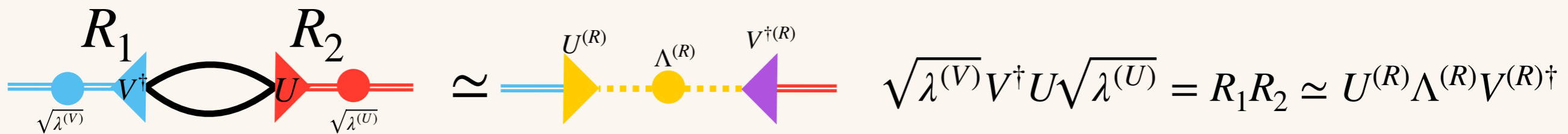
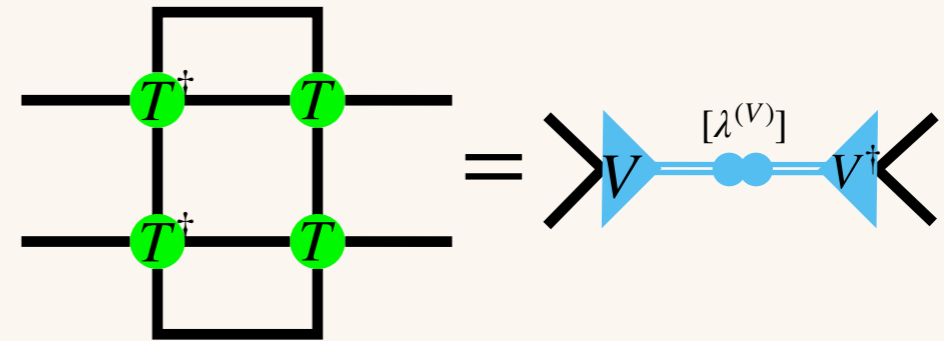
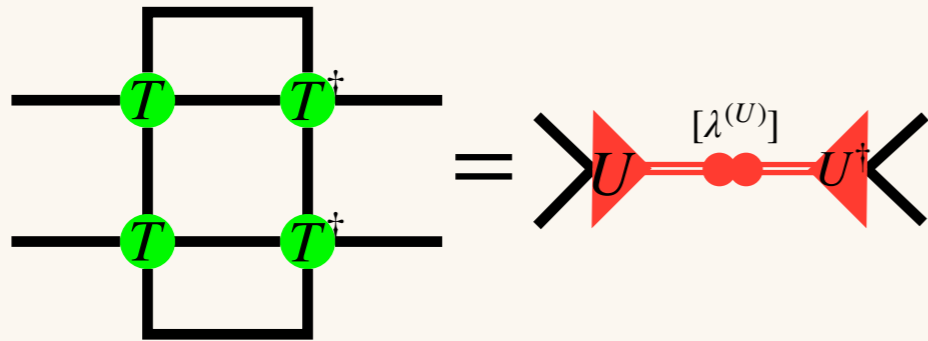
● boundary HOTRG

[S. Iino et al 1905.02351]



$$TT = U\sqrt{\lambda^{(U)}}V^{(U)\dagger} = U^{(V)}\sqrt{\lambda^{(V)}}V^{\dagger}$$

$$= R_2V^{(U)\dagger} = U^{(V)}R_1$$



$$P_1 = U\sqrt{\lambda^{(U)}}V^{(R)}/\sqrt{\Lambda^{(R)}} = R_2V^{(R)}/\sqrt{\Lambda^{(R)}}$$

$$P_2 = \left(V\sqrt{\lambda^{(V)}}U^{(R)}/\sqrt{\Lambda^{(R)}} \right)^{\dagger} = \left(R_1^{\dagger}U^{(R)}/\sqrt{\Lambda^{(R)}} \right)^{\dagger}$$