

Faster Computation of magic monotones



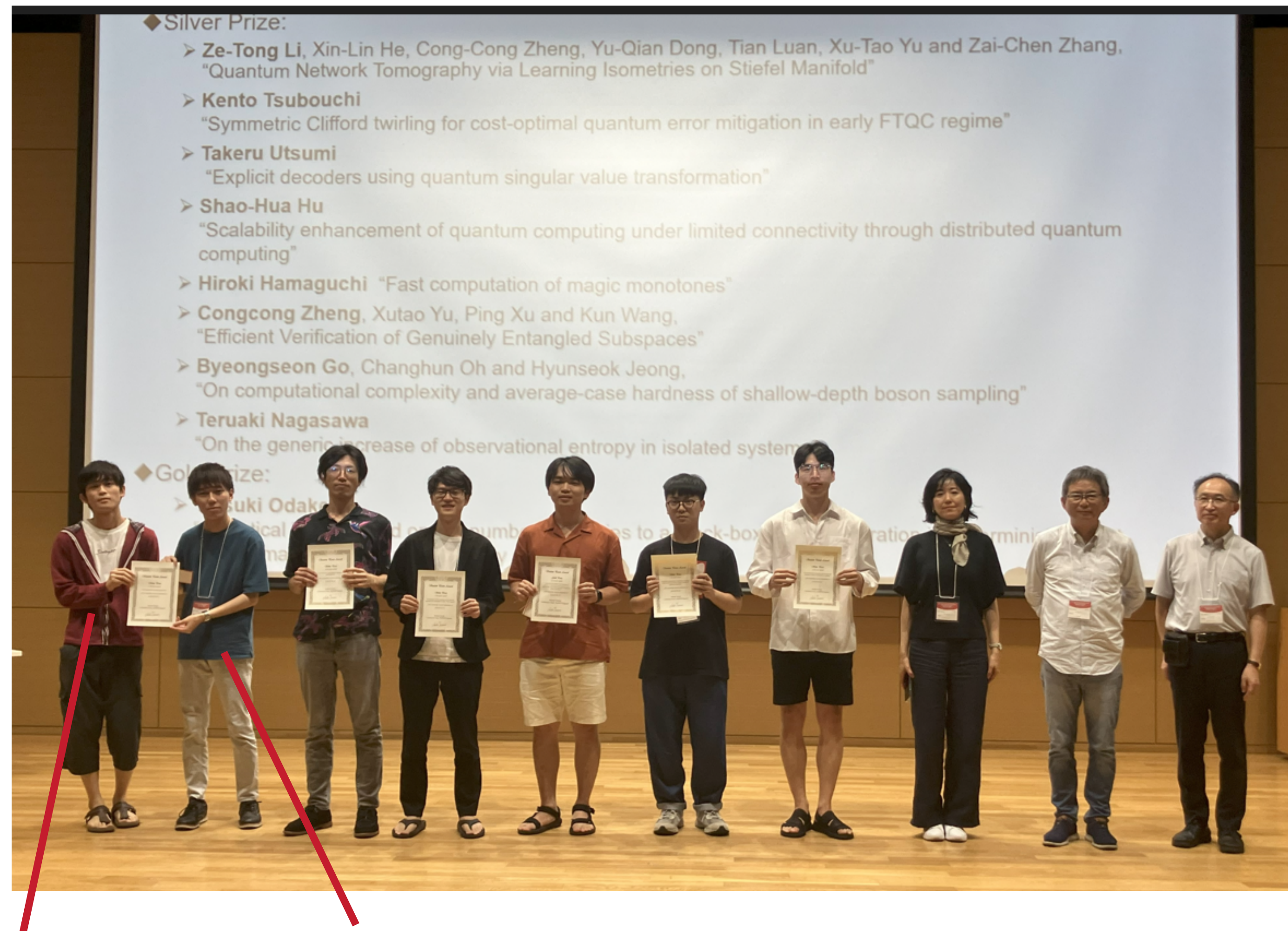
Based on [arXiv:2311.01362](#), [arXiv:2406.16673](#)

Nobuyuki Yoshioka
The University of Tokyo

2024.11.18



AQIS 2024 poster prize winners



Hiroki
Hamaguchi Kou
Hamada

Scaling for exact computation

Target	Application	Formulation	Subroutine time complexity		Memory	
			Naive	Ours	Naive	Ours
Robustness of magic [20]	Clifford+T sim. Circuit synthesis	LP	$\mathcal{O}(\mathcal{S}_n 2^n)$	$\mathcal{O}(\mathcal{S}_n n)$	$\mathcal{O}(\mathcal{S}_n 2^n)$	$\mathcal{O}(2^n)$
Stabilizer extent [24]	Clifford+T sim.	SOCP	$\mathcal{O}(\mathcal{S}_n 2^n n^2)$	$\mathcal{O}(\mathcal{S}_n)$	$\mathcal{O}(\mathcal{S}_n 2^n)$	$\mathcal{O}(2^n)$
Stabilizer fidelity [24]	Bound for RoM	Overlap calculation	$\mathcal{O}(\mathcal{S}_n 2^n n^2)$	$\mathcal{O}(\mathcal{S}_n)$	$\mathcal{O}(2^n)$	$\mathcal{O}(2^n)$
Pauli decomposition	Circuit simulation Noise analysis Quantum benchmark	Matrix-vector multiplication	$\mathcal{O}(16^n)$	$\mathcal{O}(4^n n)$	$\mathcal{O}(4^n)$	$\mathcal{O}(4^n)$

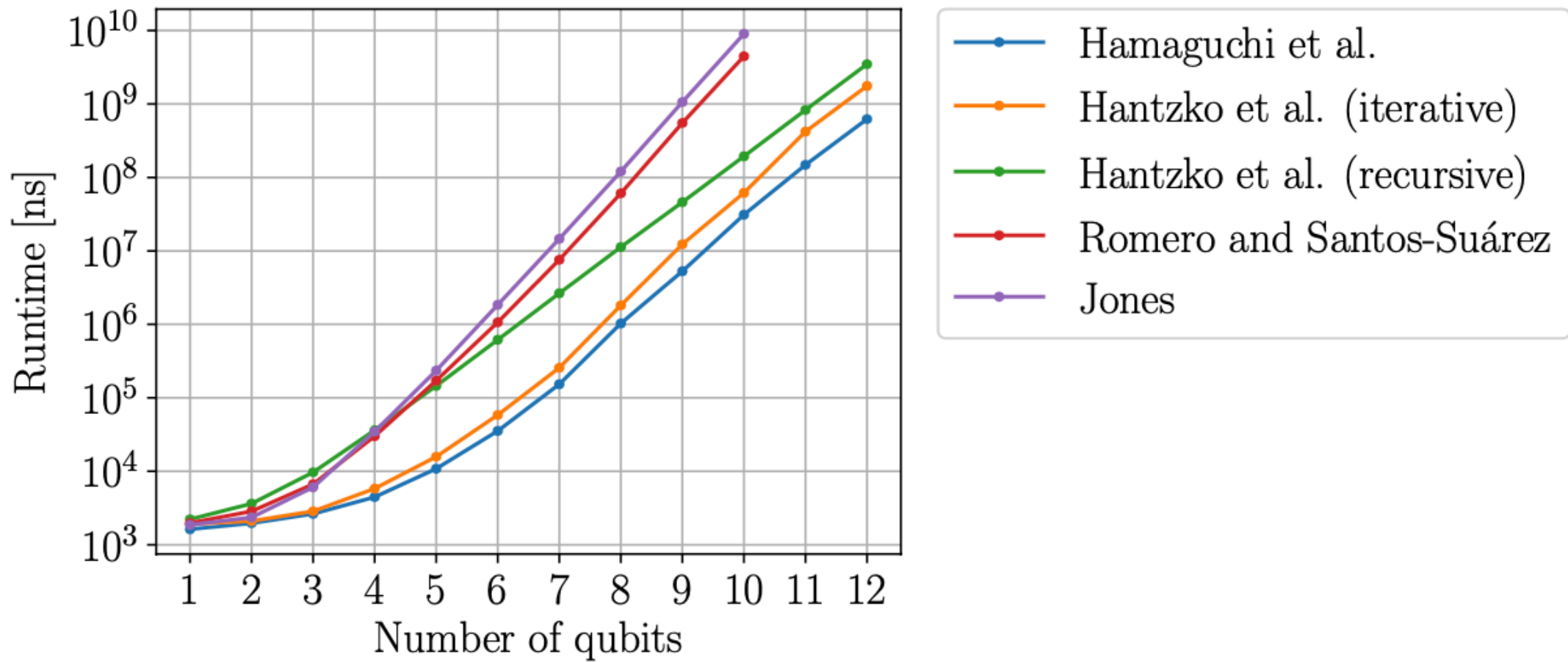
Scaling for exact computation

Target	Application	Formulation	Subroutine time complexity		Memory	
			Naive	Ours	Naive	Ours
Robustness of magic [20]	Clifford+T sim. Circuit synthesis	LP	$\mathcal{O}(\mathcal{S}_n 2^n)$	$\mathcal{O}(\mathcal{S}_n n)$	$\mathcal{O}(\mathcal{S}_n 2^n)$	$\mathcal{O}(2^n)$
Stabilizer extent [24]	Clifford+T sim.	SOCP	$\mathcal{O}(\mathcal{S}_n 2^n n^2)$	$\mathcal{O}(\mathcal{S}_n)$	$\mathcal{O}(\mathcal{S}_n 2^n)$	$\mathcal{O}(2^n)$
Stabilizer fidelity [24]	Bound for RoM	Overlap calculation	$\mathcal{O}(\mathcal{S}_n 2^n n^2)$	$\mathcal{O}(\mathcal{S}_n)$	$\mathcal{O}(2^n)$	$\mathcal{O}(2^n)$
Pauli decomposition	Circuit simulation Noise analysis Quantum benchmark	Matrix-vector multiplication	$\mathcal{O}(16^n)$	$\mathcal{O}(4^n n)$	$\mathcal{O}(4^n)$	$\mathcal{O}(4^n)$

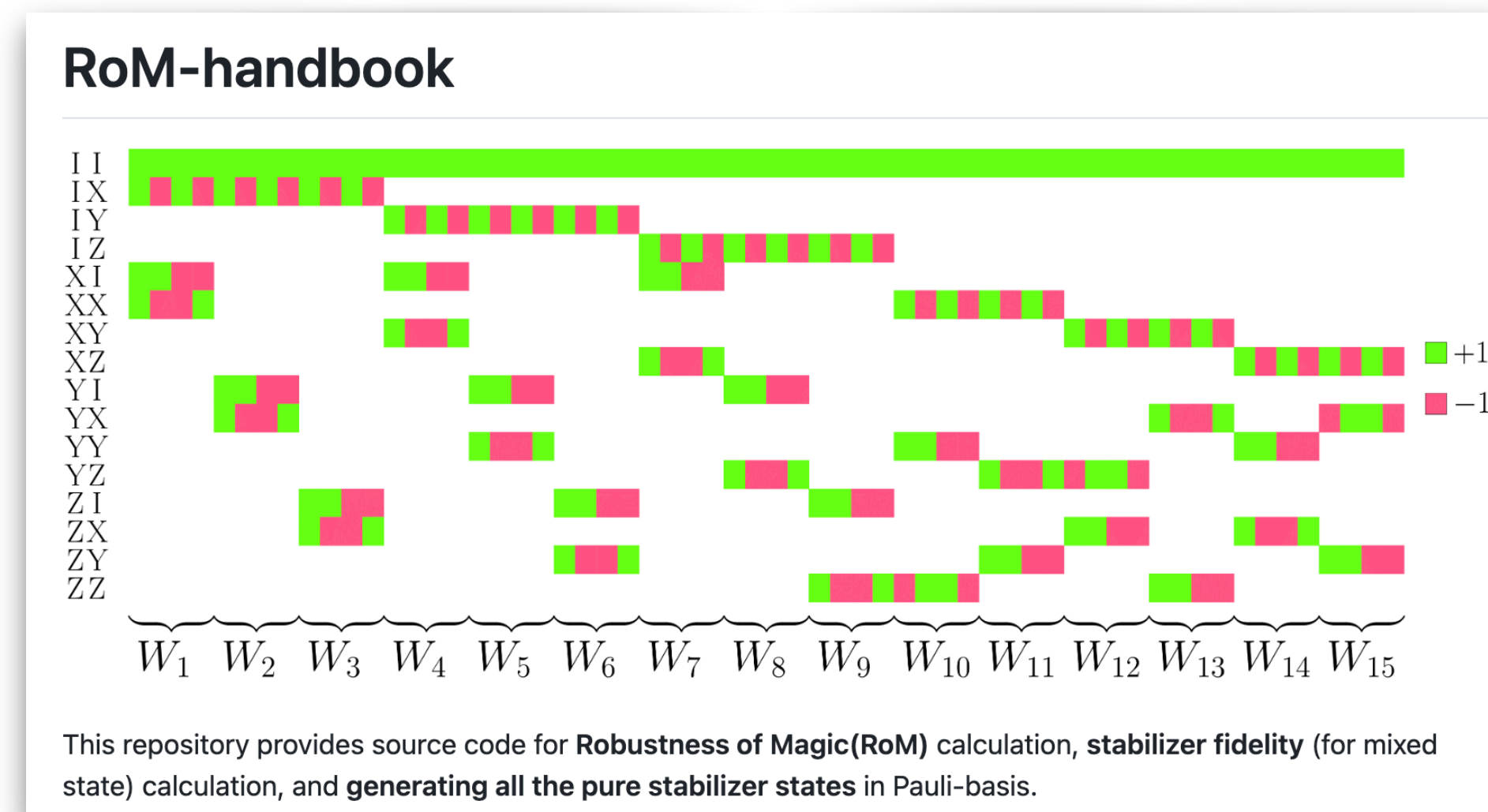
Run time and memory

qubit count n	5	6	7	8	9
states $ \mathcal{S}_n $	2.4×10^6	3.2×10^8	8.1×10^{10}	4.2×10^{13}	4.3×10^{16}
size of A_n^{RoM}	379 MiB	95 GiB	86 TiB	86 PiB	172 EiB
RoM naive time	2 min	×	×	×	×
our time	2.3 s	7.0 min	1.6 h	2.0 d	×
size of A_n^{SE}	1011 MiB	254 GiB	153 TiB	153 PiB	305 EiB
SE naive time	7.7 min	×	×	×	×
our time	1.5 s	3.8 s	12.9 s	8.8 min	19.2 h

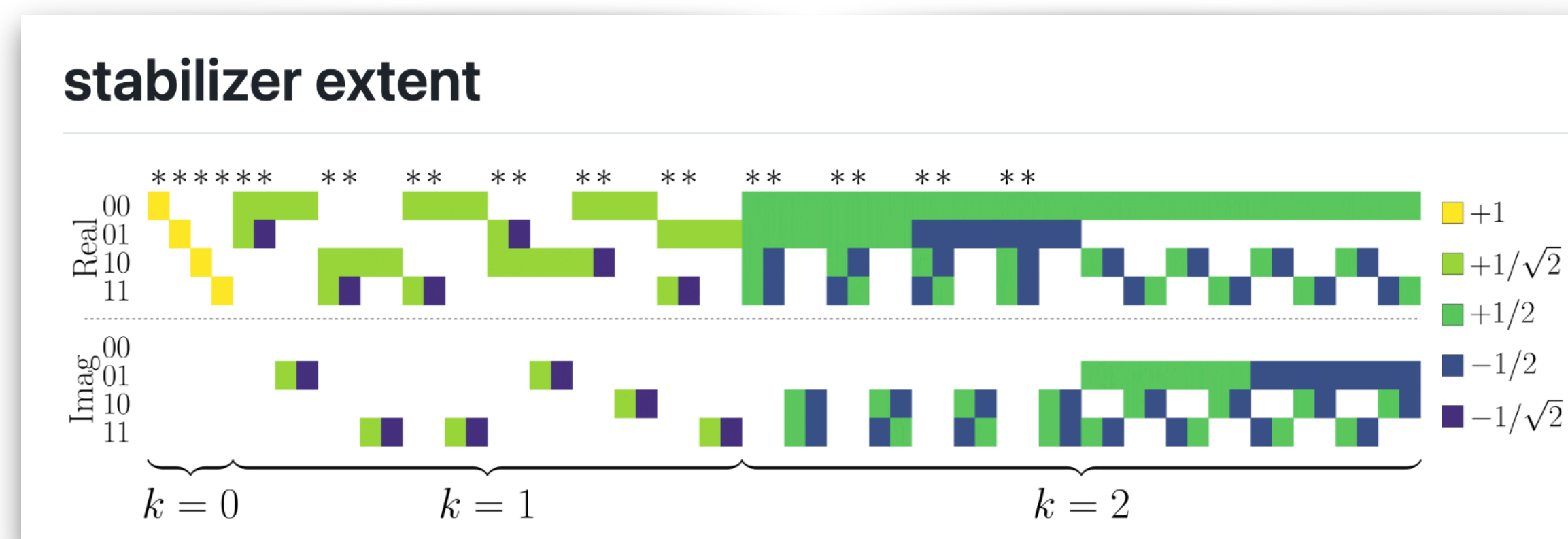
Pauli decomposition run time



RoM: <https://github.com/quantum-programming/RoM-handbook>



Stabilizer extent & fidelity: https://github.com/quantum-programming/stabilizer_extent



	Definition	Formulation
<div>Robustness of magic</div> <div>Howard&Campbell, PRL ('17)</div>	$\mathcal{R}(\rho) := \min_x \left\{ \ x\ _1 \left \rho = \sum_i x_i \sigma_i \right. \right\}$ $= \min_{x \in \mathbb{R}^{ \mathcal{S}_n }} \left\{ \ x\ _1 \left b = A^{\text{RoM}} x \right. \right\} \quad \begin{matrix} b_j = \text{Tr}(P_j \rho) \\ A_{j,i}^{\text{RoM}} = \text{Tr}(P_j \sigma_i) \end{matrix}$	Linear Program (LP)
<div>Stabilizer extent</div> <div>BBCCGH, Quantum ('19)</div>	$\xi(\psi\rangle) = \min_x \left\{ \ x\ _1 \left \psi\rangle = \sum_i x_i \phi_i\rangle \right. \right\}$ $= \min_{x \in \mathbb{C}^{ \mathcal{S}_n }} \left\{ \ x\ _1^2 \left b = A^{\text{SE}} x \right. \right\} \quad \begin{matrix} b_j = \langle j \psi \rangle \\ A_{j,i}^{\text{SE}} = \langle j \phi_i \rangle \end{matrix}$	Second-Order Cone Program (SOCP)

	Definition	Formulation
Robustness of magic Howard&Campbell, PRL ('17)	$\mathcal{R}(\rho) := \min_x \left\{ \ x\ _1 \mid \rho = \sum_i x_i \sigma_i \right\}$ $= \min_{x \in \mathbb{R}^{ S_n }} \left\{ \ x\ _1 \mid b = A^{\text{RoM}} x \right\} \quad \begin{array}{l} b_j = \text{Tr}(P_j \rho) \\ A_{j,i}^{\text{RoM}} = \text{Tr}(P_j \sigma_i) \end{array}$	Linear Program (LP)
Stabilizer extent BBCCGH, Quantum ('19)	$\xi(\psi\rangle) = \min_x \left\{ \ x\ _1 \mid \psi\rangle = \sum_i x_i \phi_i\rangle \right\}$ $= \min_{x \in \mathbb{C}^{ S_n }} \left\{ \ x\ _1^2 \mid b = A^{\text{SE}} x \right\} \quad \begin{array}{l} b_j = \langle j \psi \rangle \\ A_{j,i}^{\text{SE}} = \langle j \phi_i \rangle \end{array}$	Second-Order Cone Program (SOCP)

Common property: Convex optimization that considers full set of Stabilizer states

Pros: Poly-time solution w.r.t. problem size

Cons: Stabilizer state set scales as $|S_n| = 2^{O(n^2)}$

Q1. The solution expected to be “sparse.”

How to systematically predict contributing bases?

➡ Observation: overlap between the target

RoM: Large *and* small overlaps

→ Compute all the overlaps efficiently

SE : Large overlaps

→ Use branch-and-bound method

Q1. The solution expected to be “sparse.”

How to systematically predict contributing bases?

➔ Observation: overlap between the target

RoM: Large *and* small overlaps

→ Compute all the overlaps efficiently

SE : Large overlaps

→ Use branch-and-bound method

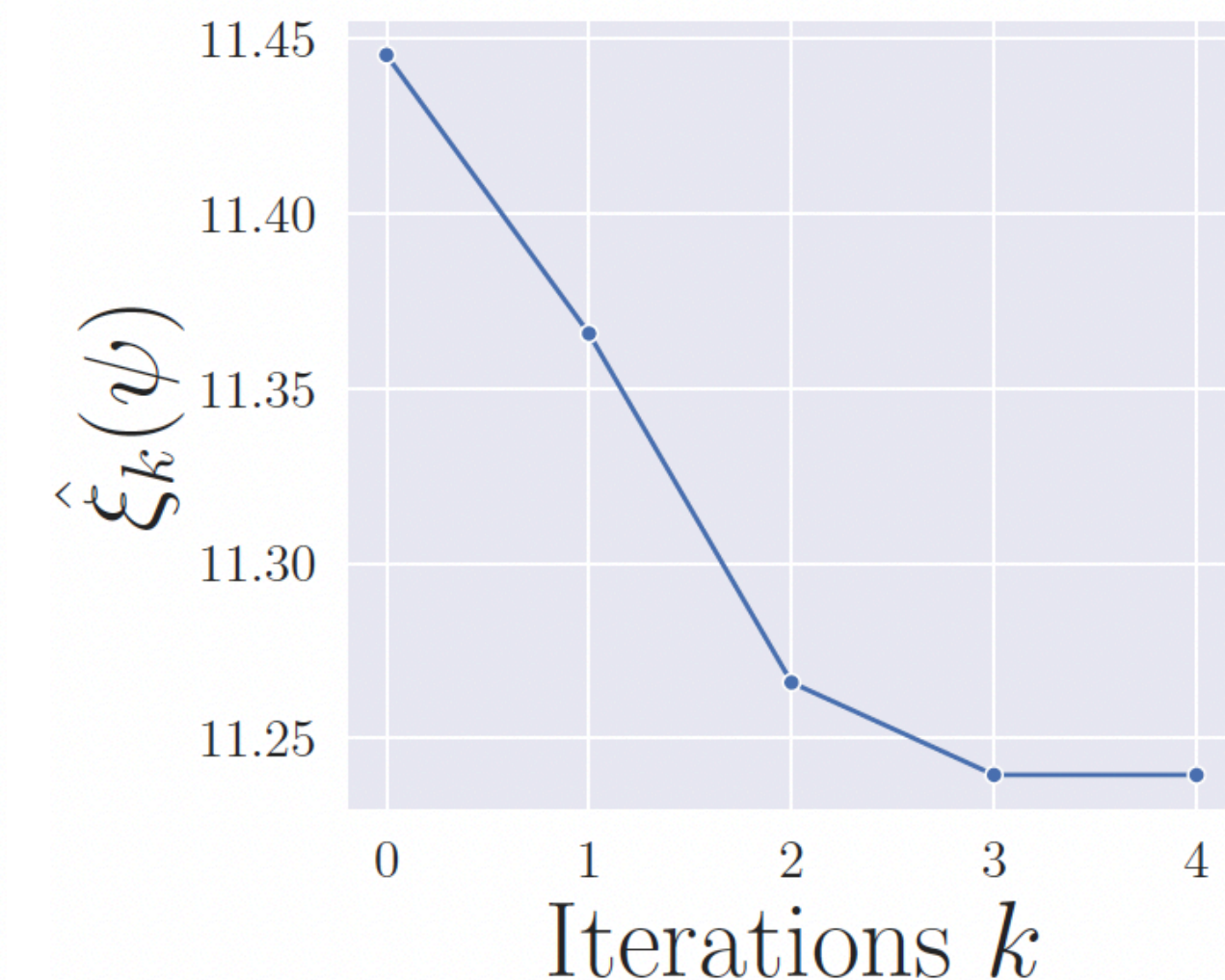
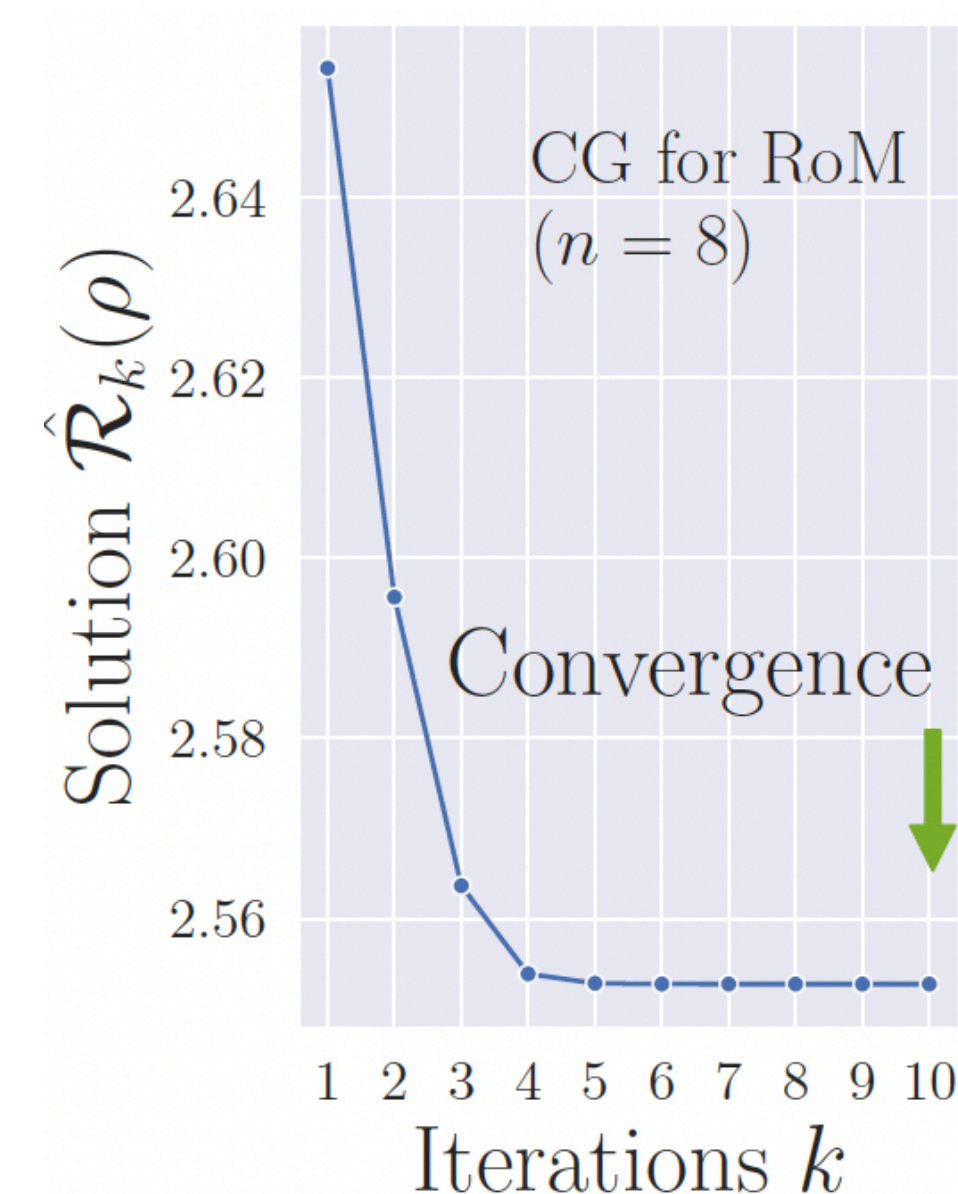
Q2. How to improve approx. solution?

And ensure optimality at the end?

➔ Column Generation (CG) technique.

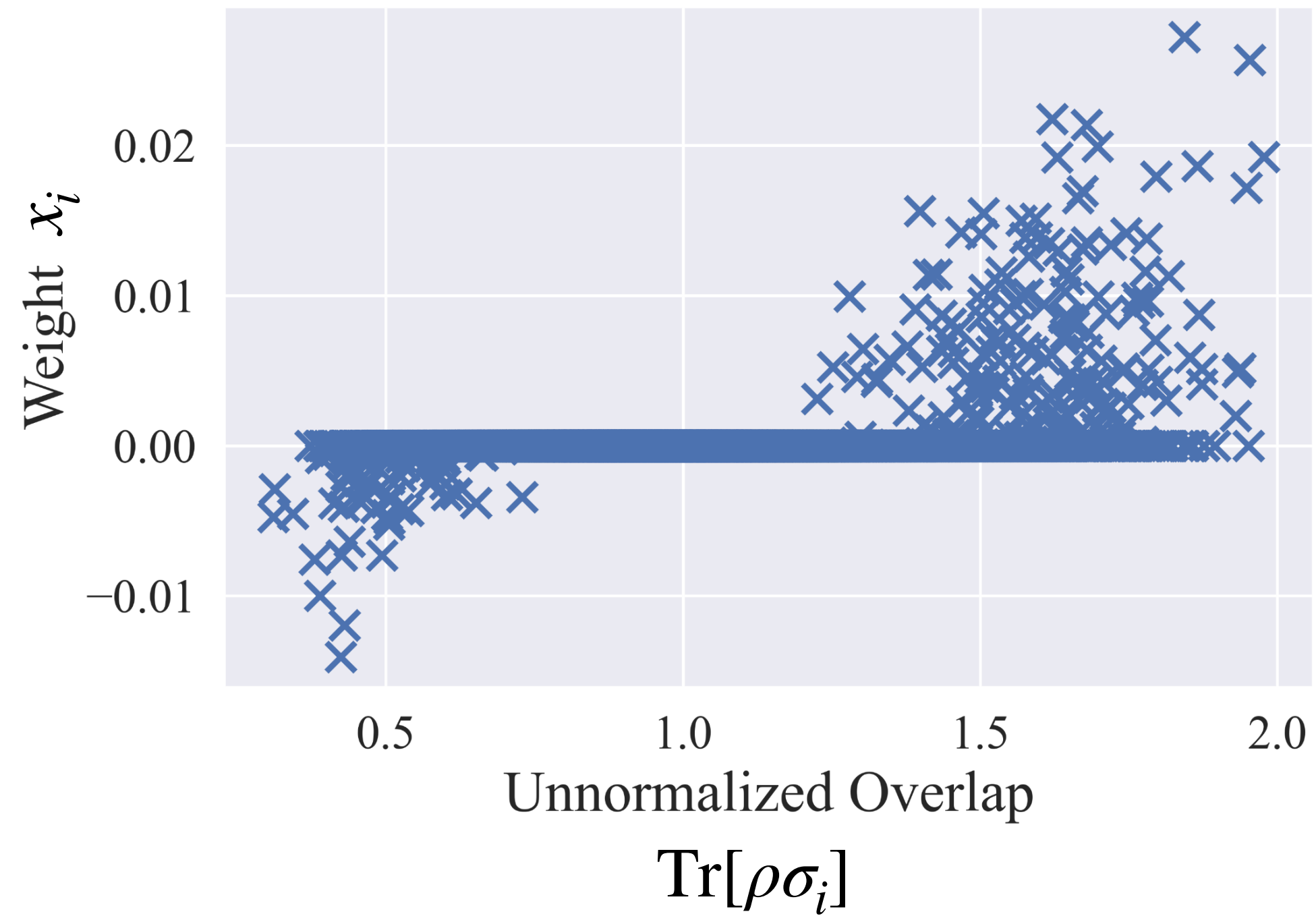
Step 1: Initial guess based on overlaps

Step 2: Iteratively update the guess by CG



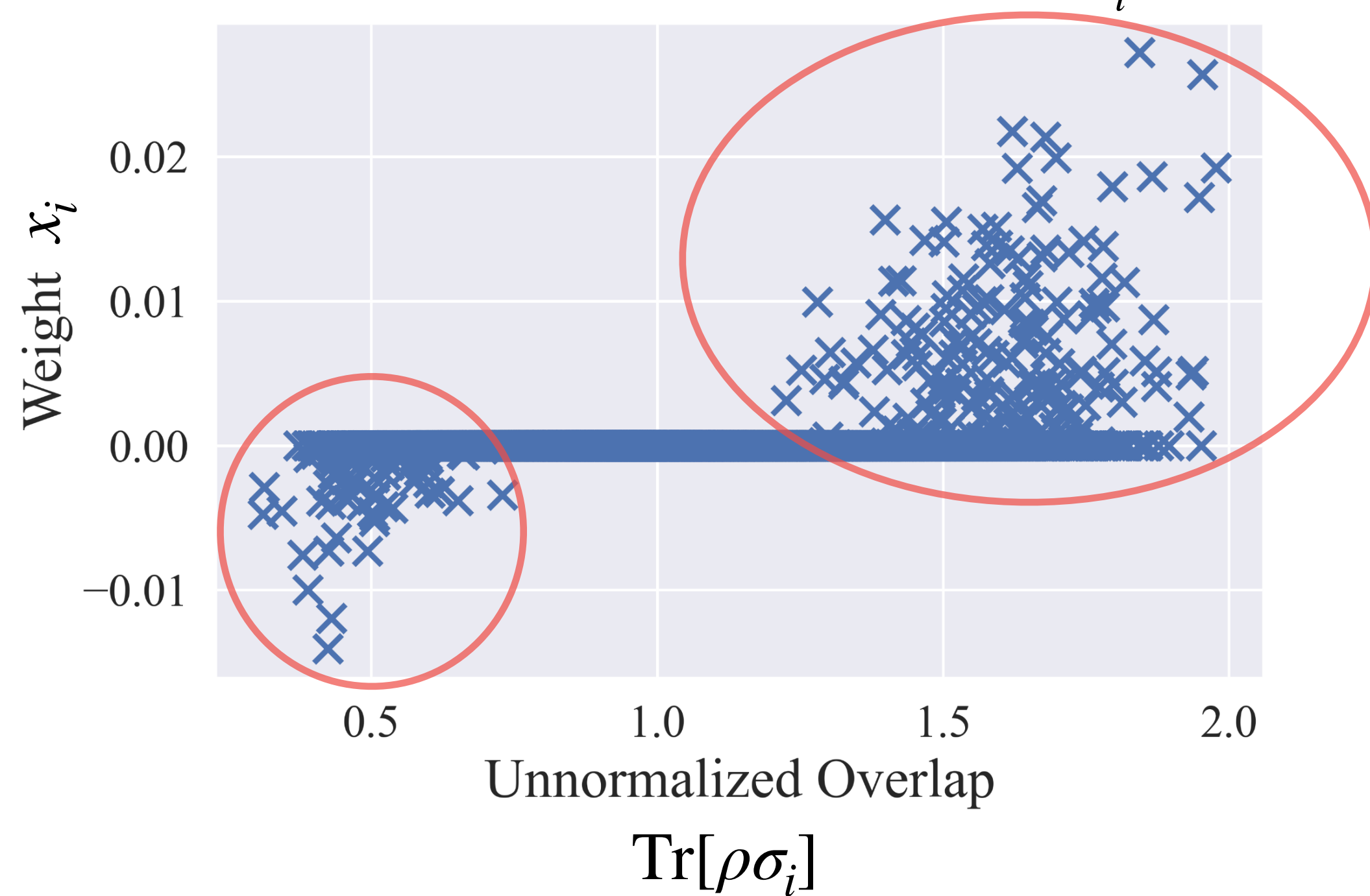
Weight x_i and stabilizer overlaps $\text{Tr}[\rho\sigma_i]$

4-qubit random mixed state, $\rho = \sum_i x_i \sigma_i$



Weight x_i and stabilizer overlaps $\text{Tr}[\rho\sigma_i]$

4-qubit random mixed state, $\rho = \sum_i x_i \sigma_i$

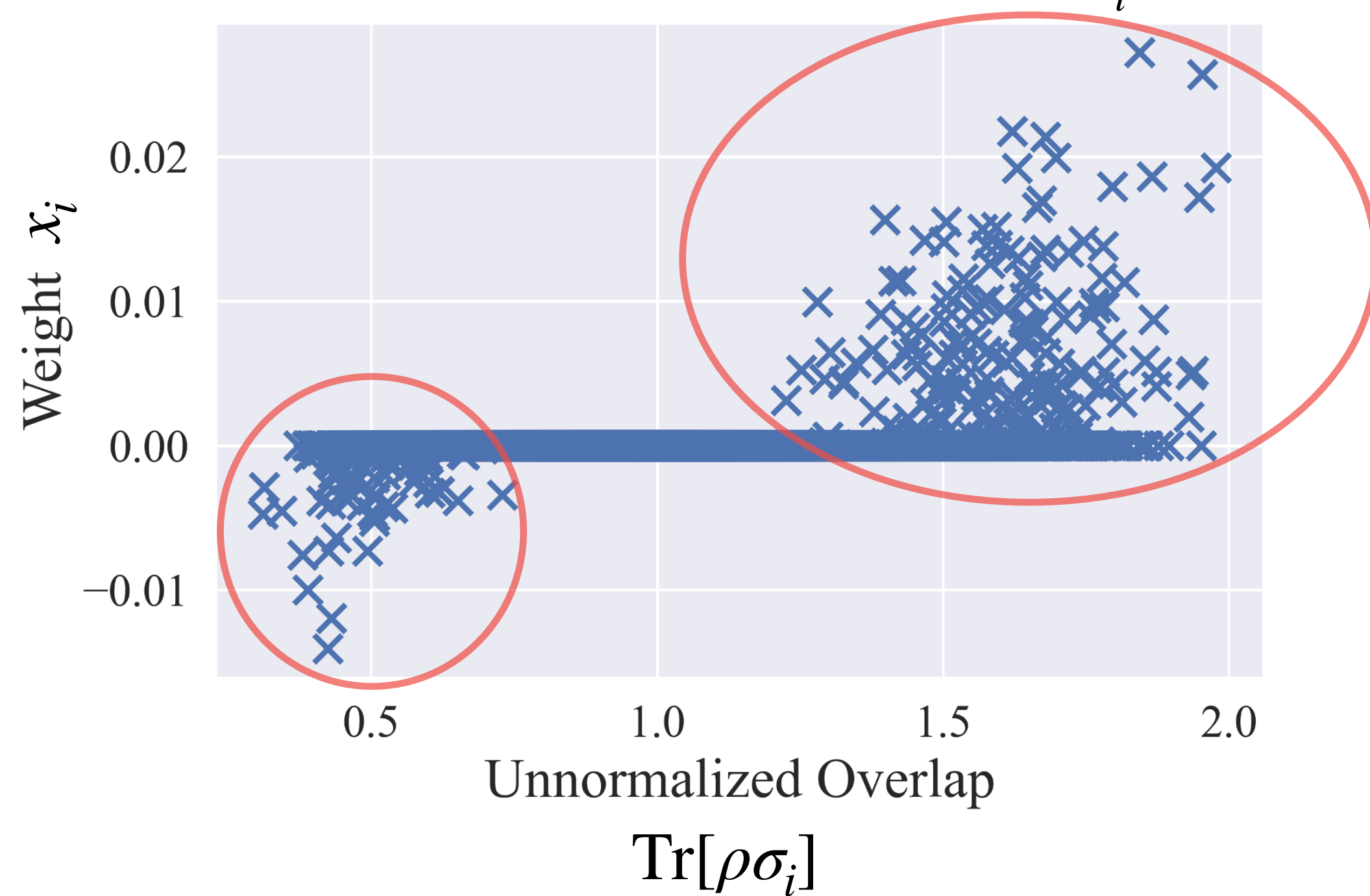


- Observation : large and small overlapping states contribute to RoM

Overlaps are good metric for dimension reduction

Weight x_i and stabilizer overlaps $\text{Tr}[\rho\sigma_i]$

4-qubit random mixed state, $\rho = \sum_i x_i \sigma_i$



- Observation : large and small overlapping states contribute to RoM

Overlaps are good metric for dimension reduction

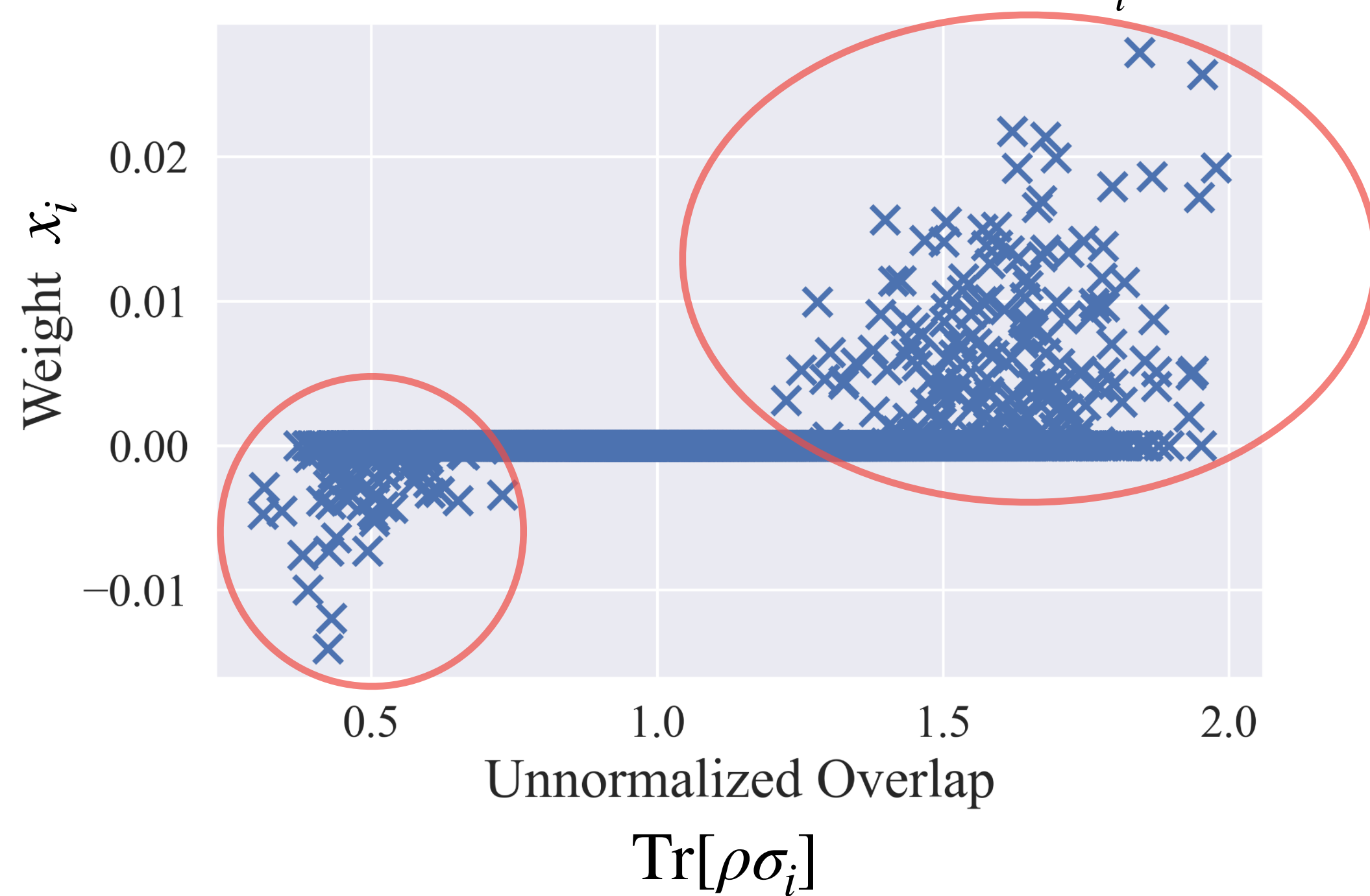
- Since we have

$$(\text{Overlap}) = \text{Tr}[\rho\sigma_i] = \sum_{j,k} \text{Tr}[b_j P_j A_{i,k}^{\text{RoM}} P_i] = (A^{\text{RoM}} b)_i$$

we need $A^{\text{RoM}} b$ for all overlap calculation

Weight x_i and stabilizer overlaps $\text{Tr}[\rho\sigma_i]$

4-qubit random mixed state, $\rho = \sum_i x_i \sigma_i$



- Observation : large and small overlapping states contribute to RoM

Overlaps are good metric for dimension reduction

- Since we have

$$(\text{Overlap}) = \text{Tr}[\rho\sigma_i] = \sum_{j,k} \text{Tr}[b_j P_j A_{i,k}^{\text{RoM}} P_i] = (A^{\text{RoM}} b)_i$$

we need $A^{\text{RoM}} b$ for all overlap calculation

- Naive cost : Time $O(|S_n| 2^n)$, memory $O(|S_n| 2^n)$

Our cost : Time $O(|S_n| n)$, memory $O(2^n)$

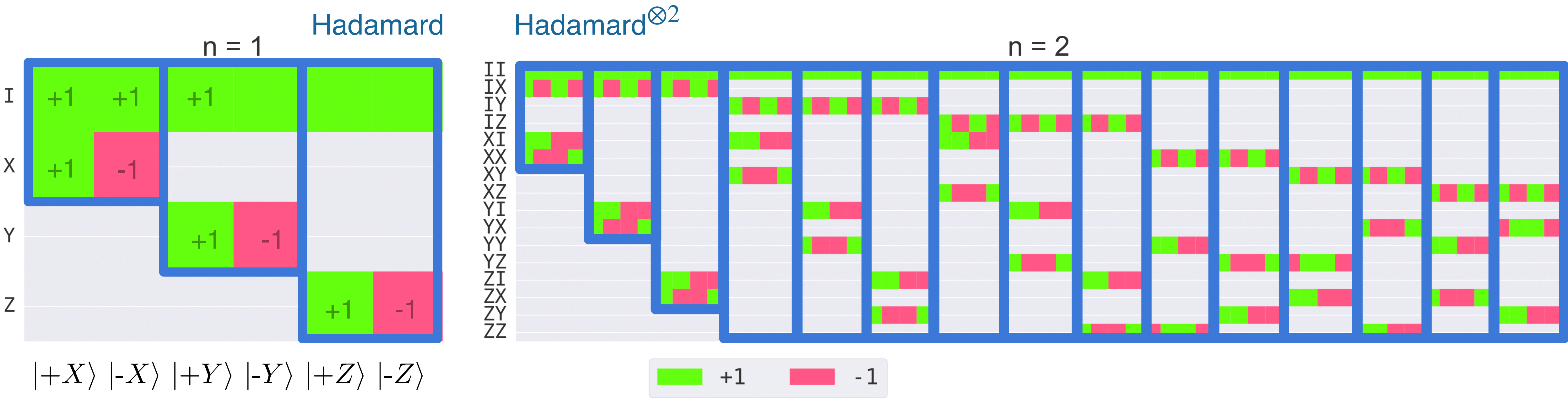
n = 1

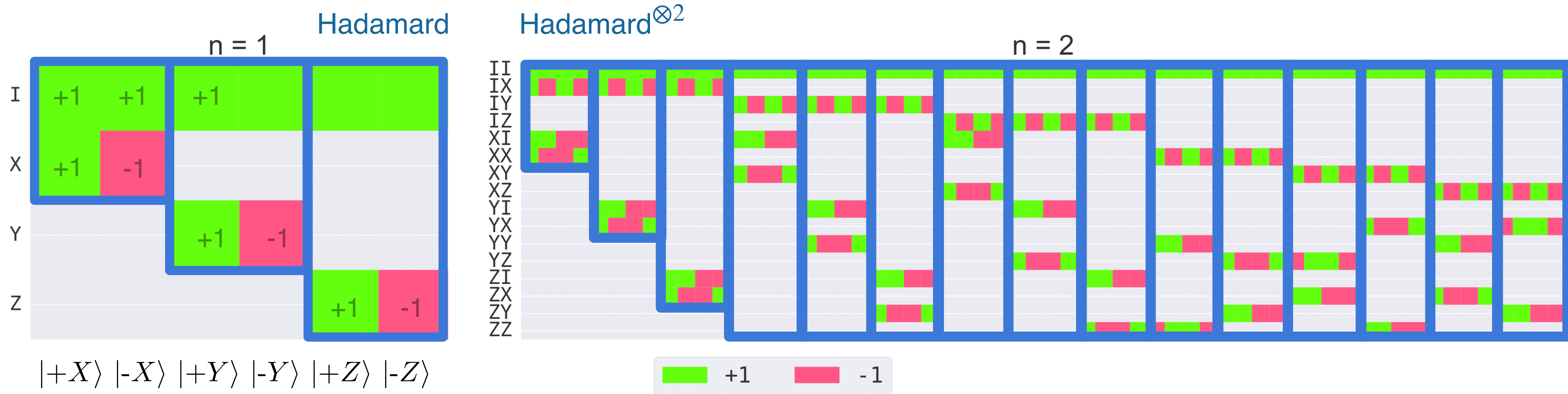
I	+1	+1	+1			
X	+1	-1				
Y			+1	-1		
Z					+1	-1
	$ +X\rangle$	$ -X\rangle$	$ +Y\rangle$	$ -Y\rangle$	$ +Z\rangle$	$ -Z\rangle$

Hadamard

n = 1

I	+1	+1	+1		
X	+1	-1			
Y		+1	-1		
Z			+1	-1	
	$ +X\rangle$	$ -X\rangle$	$ +Y\rangle$	$ -Y\rangle$	$ +Z\rangle$ $ -Z\rangle$





Core decomposition technique

A matrix is concatenation of Hadamard matrix $H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{\otimes n}$ with sparsification & sign change

Matrix multiplication by Fast Walsh-Hadamard Transformation (FWHT)

Matrix-vector multiplication of $M^{\otimes n}$ can be done by $O(n2^n)$, instead of naive $O(4^n)$

Example in $\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{\otimes 3}$









000:	1
001:	0
010:	1
011:	1
100:	0
101:	1
110:	1
111:	0

Matrix multiplication by Fast Walsh-Hadamard Transformation (FWHT)

Matrix-vector multiplication of $M^{\otimes n}$ can be done by $O(n2^n)$, instead of naive $O(4^n)$

Example in $\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{\otimes 3}$

















1st qubit

000:	1		$1+0 = 1$
001:	0		$-0+1 = 1$
010:	1		$1+1 = 2$
011:	1		$-1+1 = 0$
100:	0		$0+1 = 1$
101:	1		$-1+0 = -1$
110:	1		$1+0 = 1$
111:	0		$-0+1 = 1$

Matrix multiplication by Fast Walsh-Hadamard Transformation (FWHT)

Matrix-vector multiplication of $M^{\otimes n}$ can be done by $O(n2^n)$, instead of naive $O(4^n)$

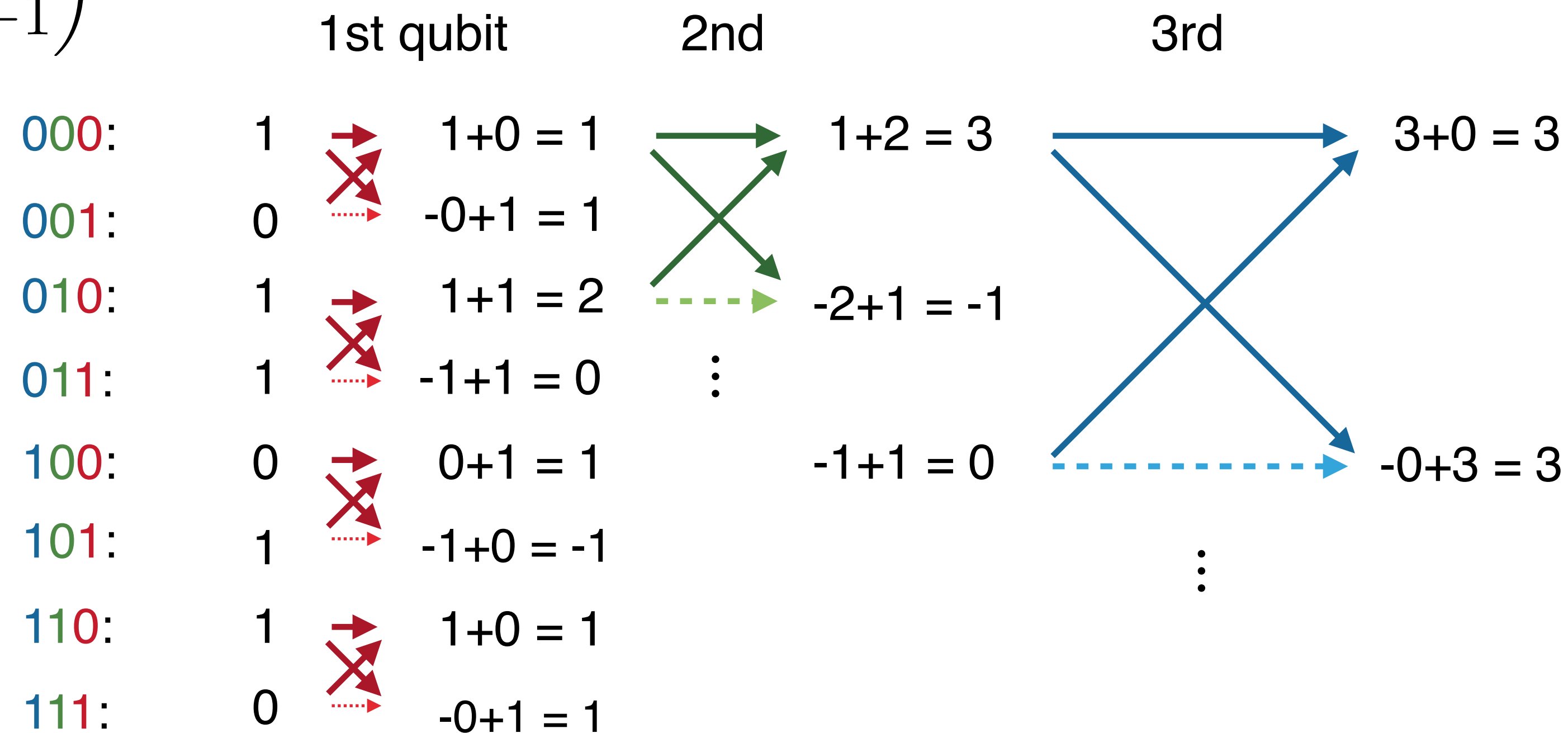
Example in $\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{\otimes 3}$

	1st qubit	2nd
000:	1  1+0 = 1	 1+2 = 3
001:	0  -0+1 = 1	
010:	1  1+1 = 2	
011:	1  -1+1 = 0	
100:	0  0+1 = 1	
101:	1  -1+0 = -1	
110:	1  1+0 = 1	
111:	0  -0+1 = 1	

Matrix multiplication by Fast Walsh-Hadamard Transformation (FWHT)

Matrix-vector multiplication of $M^{\otimes n}$ can be done by $O(n2^n)$, instead of naive $O(4^n)$

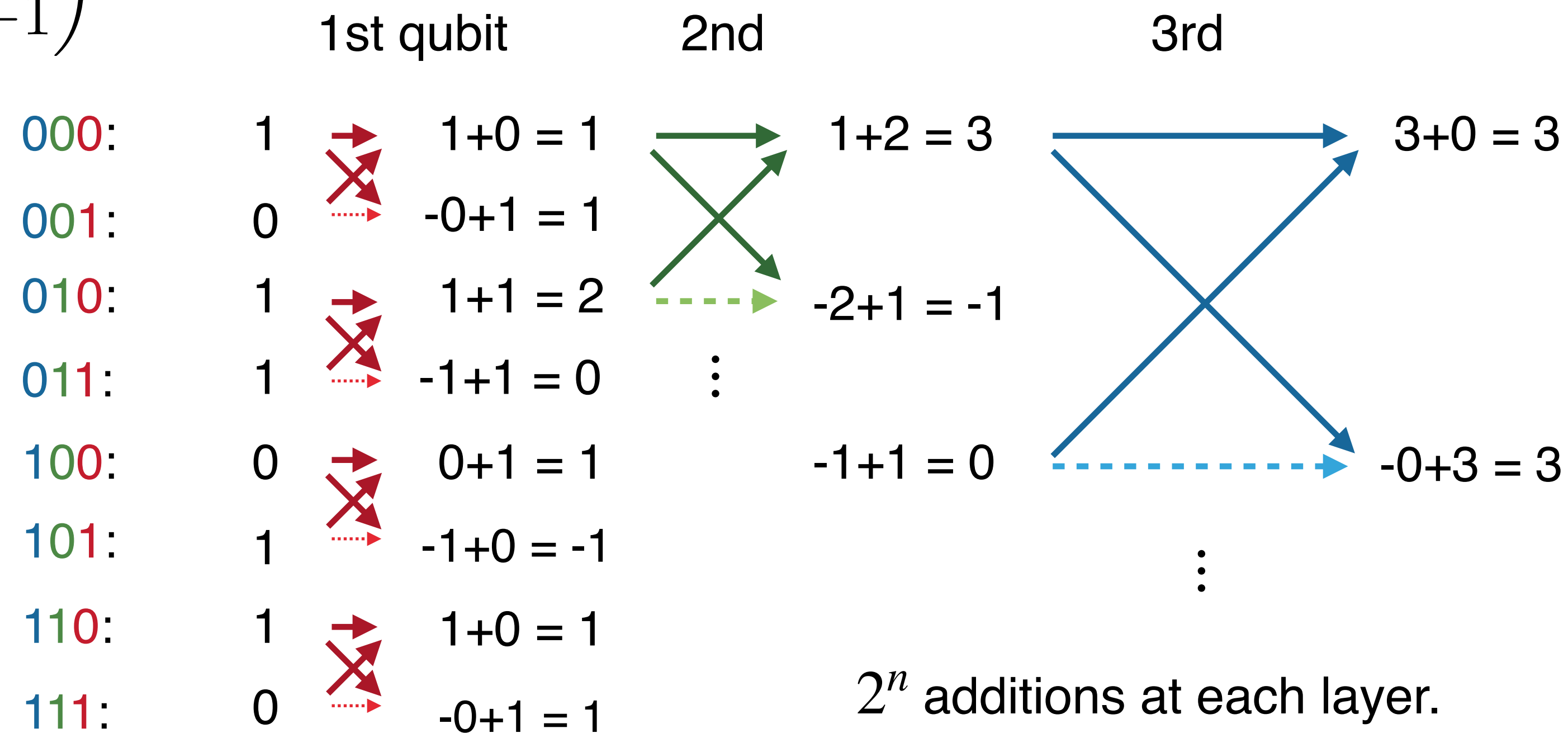
Example in $\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{\otimes 3}$



Matrix multiplication by Fast Walsh-Hadamard Transformation (FWHT)

Matrix-vector multiplication of $M^{\otimes n}$ can be done by $O(n2^n)$, instead of naive $O(4^n)$

Example in $\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{\otimes 3}$



2^n additions at each layer.

Time complexity : $O(n2^n)$

Space complexity : $O(2^n)$

Core decomposition technique

A matrix is concatenation of Hadamard matrix $H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{\otimes n}$ with sparsification & sign change

Matrix multiplication by Fast Walsh-Hadamard Transformation (FWHT)

Matrix-vector multiplication of $M^{\otimes n}$ can be done by $O(n2^n)$, instead of naive $O(4^n)$

Core decomposition technique

A matrix is concatenation of Hadamard matrix $H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{\otimes n}$ with sparsification & sign change

Matrix multiplication by Fast Walsh-Hadamard Transformation (FWHT)

Matrix-vector multiplication of $M^{\otimes n}$ can be done by $O(n2^n)$, instead of naive $O(4^n)$

➡ Since there are $|S_n|/2^n$ blocks in total, the total time complexity is $O(n |S_n|)$
(naive : $O(2^n |S_n|)$)

Pauli Decomposition

Input : N -qubit quantum state ρ

Output : Coefficients $\{c_i\}$ such that $\rho = \sum_{P_i \in \mathcal{P}_N} c_i P_i$

Pauli Decomposition

Input : N -qubit quantum state ρ

Output : Coefficients $\{c_i\}$ such that $\rho = \sum_{P_i \in \mathcal{P}_N} c_i P_i$

Naive way

Calculate $c_i = \text{Tr}[\rho P_i]/2^N$ for every Pauli

Cost : $O(32^N)$

Pauli Decomposition

Input : N -qubit quantum state ρ

Output : Coefficients $\{c_i\}$ such that $\rho = \sum_{P_i \in \mathcal{P}_N} c_i P_i$

Naive way

Calculate $c_i = \text{Tr}[\rho P_i]/2^N$ for every Pauli

Cost : $O(32^N)$

Faster way Jones ('24)

Use of gray code to suppress computation

Cost : $O(8^N)$

Pauli Decomposition

Input : N -qubit quantum state ρ

Output : Coefficients $\{c_i\}$ such that $\rho = \sum_{P_i \in \mathcal{P}_N} c_i P_i$

Naive way

Calculate $c_i = \text{Tr}[\rho P_i]/2^N$ for every Pauli

Cost : $O(32^N)$

Faster way Jones ('24)

Use of gray code to suppress computation

Cost : $O(8^N)$

Even Faster way

Hamaguchi, Hamada, **NY** ('23)

Hantzko et al. ('23)

Step 1: $\vec{\rho} = \begin{bmatrix} \vdots \\ \rho_{ij} \\ \vdots \end{bmatrix}_{ij}$

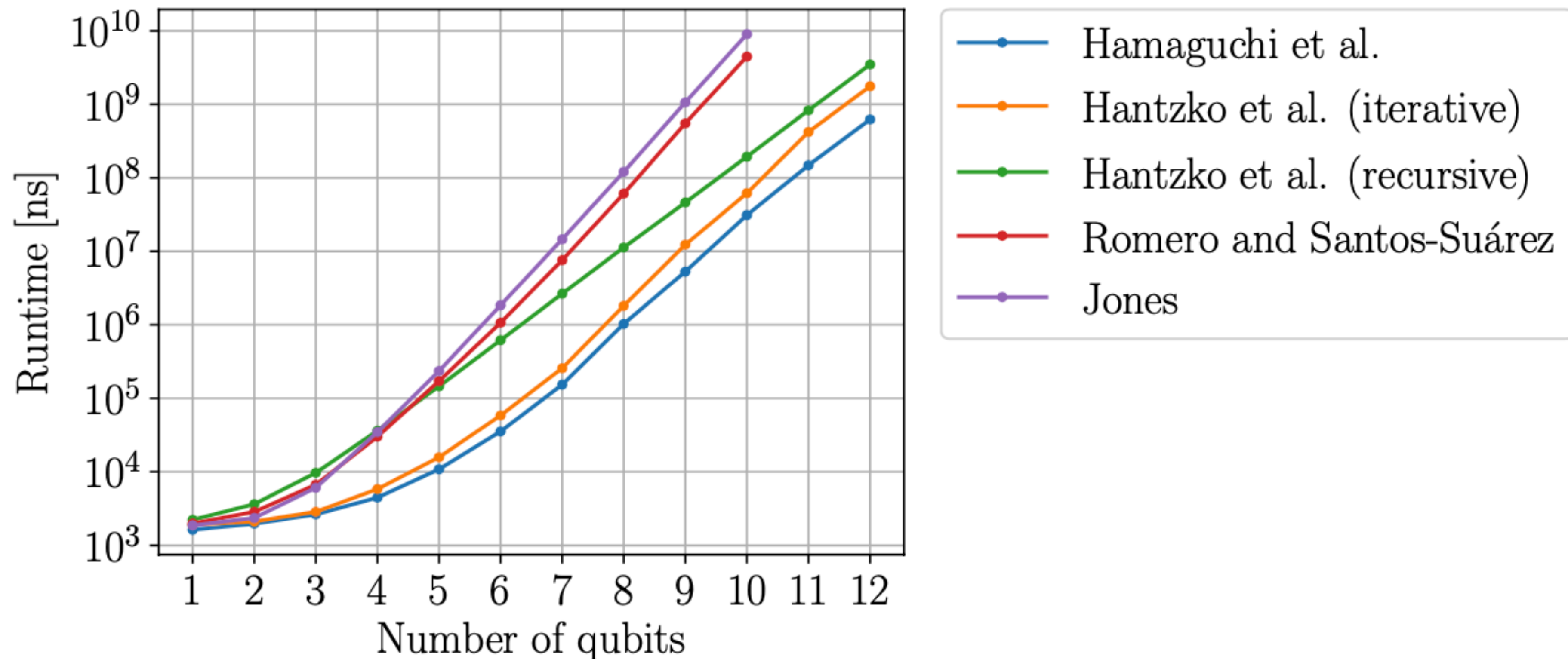
Step 2: $\vec{c} = M^{\otimes N} \vec{\rho}$ where $M := \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & i & -i & 0 \\ 1 & 0 & 0 & -1 \end{pmatrix}$

Cost : $O(N4^N)$

Pauli Decomposition

Input : N -qubit quantum state ρ

Output : Coefficients $\{c_i\}$ such that $\rho = \sum_{P_i \in \mathcal{P}_N} c_i P_i$

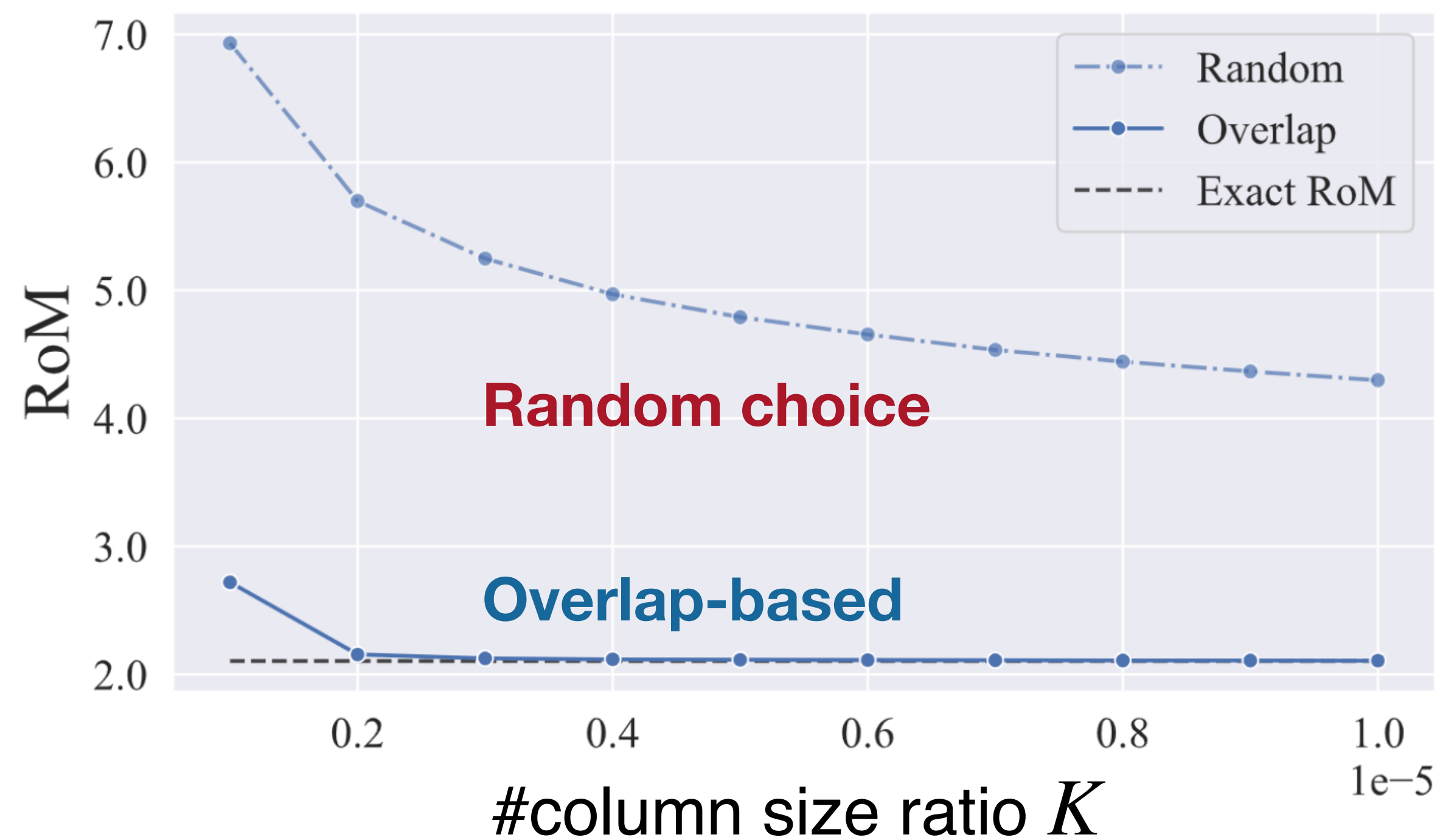


Q2. How to systematically improve approx. solution?

And ensure optimality?

Naive idea: increase the column set according to overlap

RoM, $n = 7$ qubit random mixed state

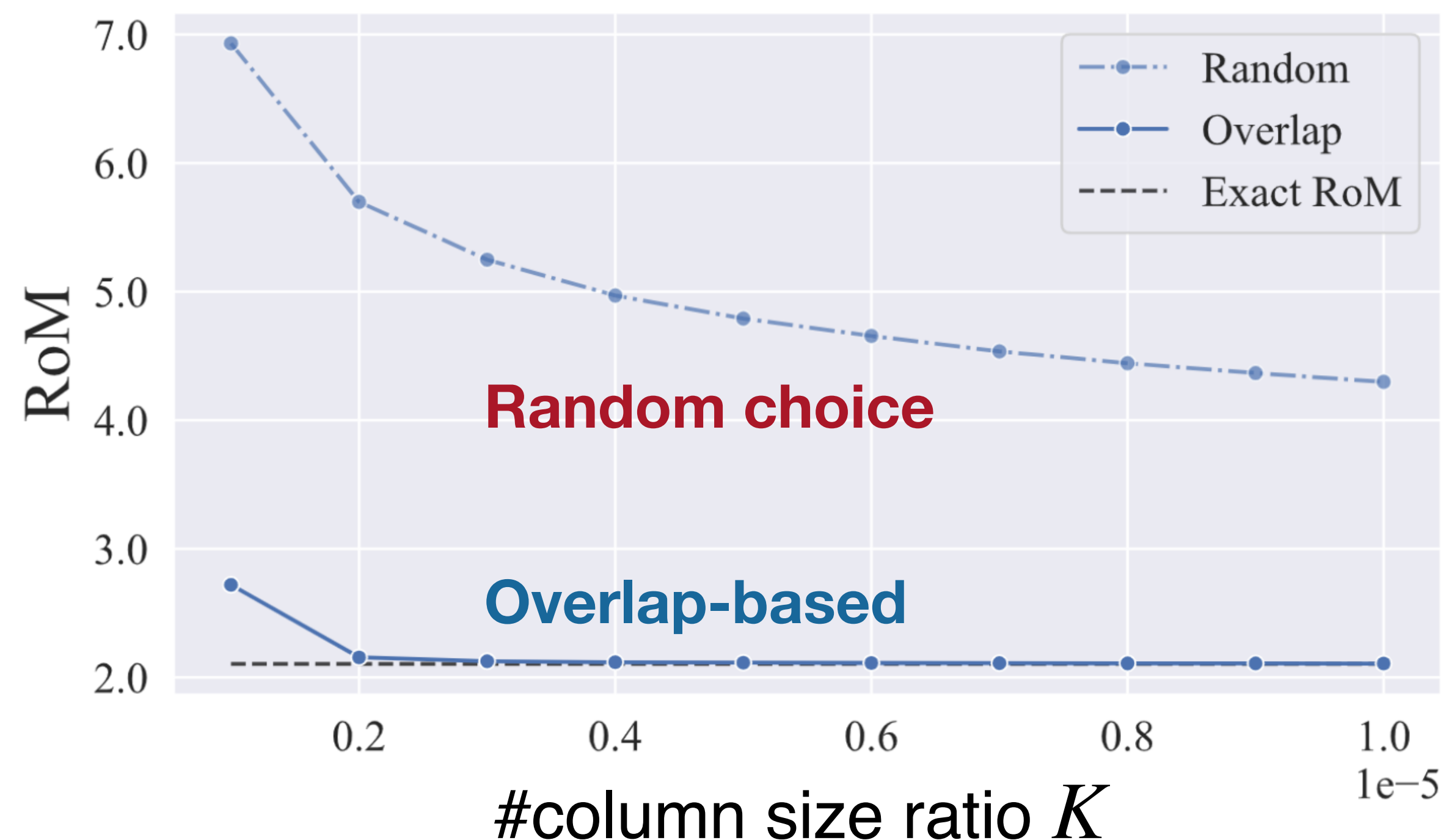


Q2. How to systematically improve approx. solution?

And ensure optimality?

Naive idea: increase the column set according to overlap

RoM, $n = 7$ qubit random mixed state



But we still have problems:

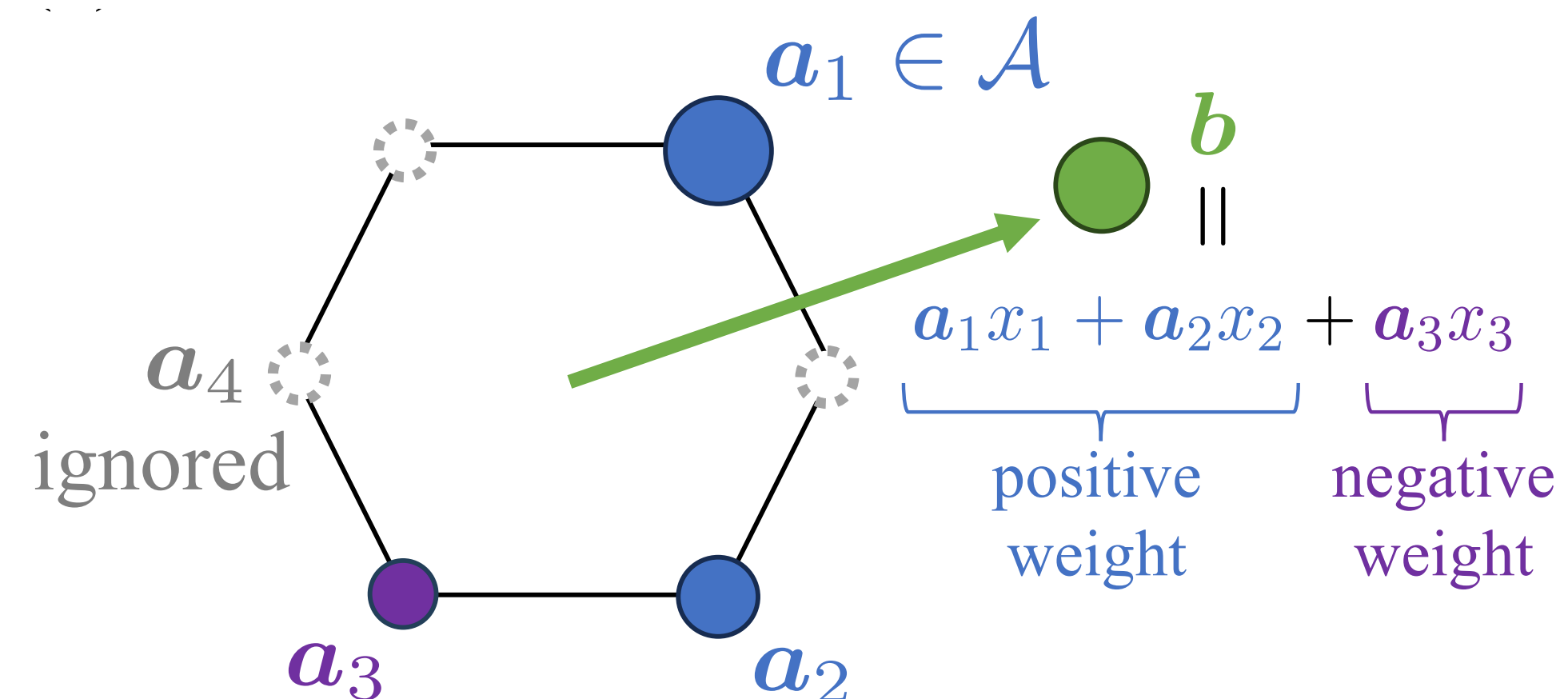
- (1) No guarantee for exactness
- (2) No quantitative way to measure the quality
- (3) Convergence from “pretty good” to “exact” is slow

We can remedy all of them using CG

Primal Problem

$$\begin{aligned} \text{(P) minimize} \quad & \| \mathbf{x} \|_1 \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b} \end{aligned}$$

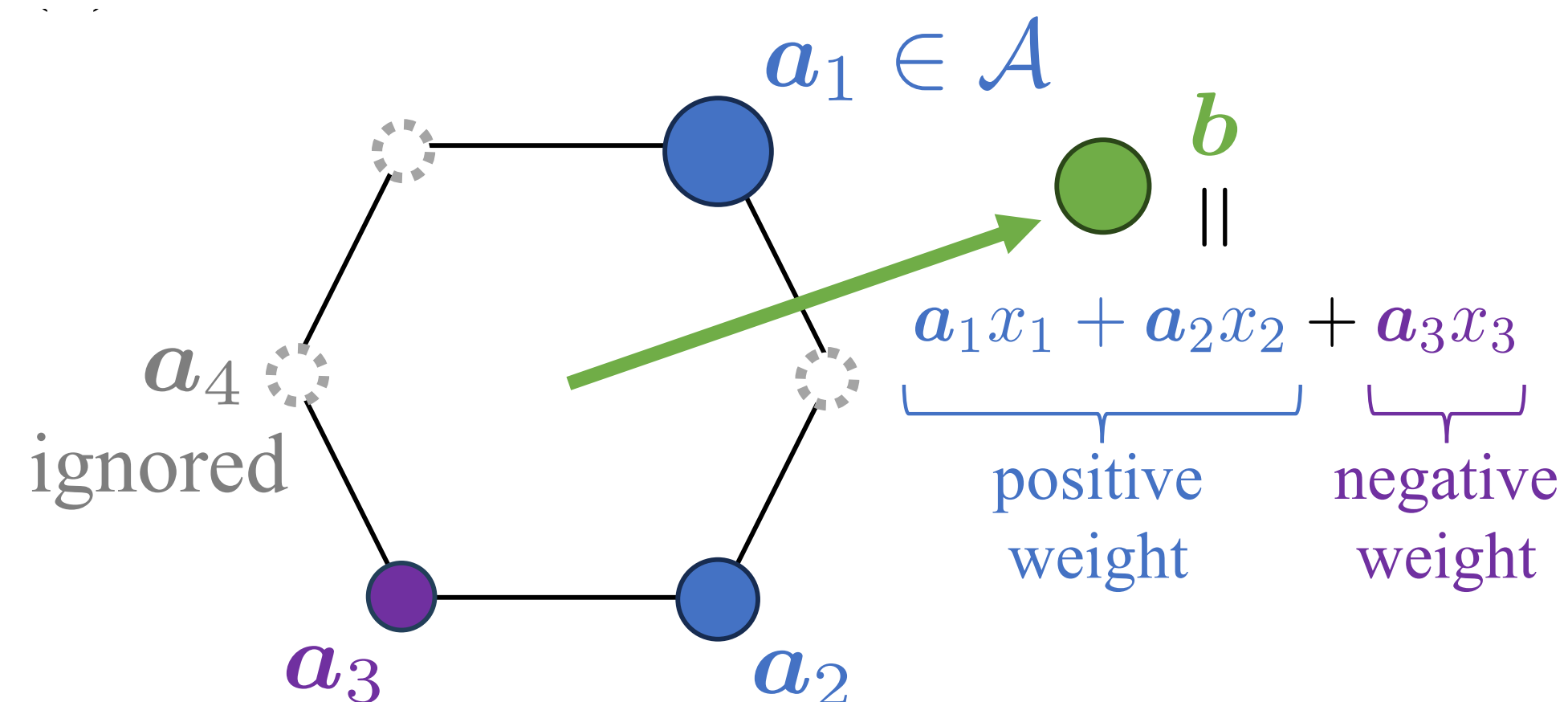
- Columns = stabilizer states.
- It is difficult to choose columns that reduces the optimal values.



Primal Problem

$$\begin{aligned} \text{(P) minimize } & \|x\|_1 \\ \text{subject to } & Ax = b \end{aligned}$$

- Columns = stabilizer states.
- It is difficult to choose columns that reduces the optimal values.

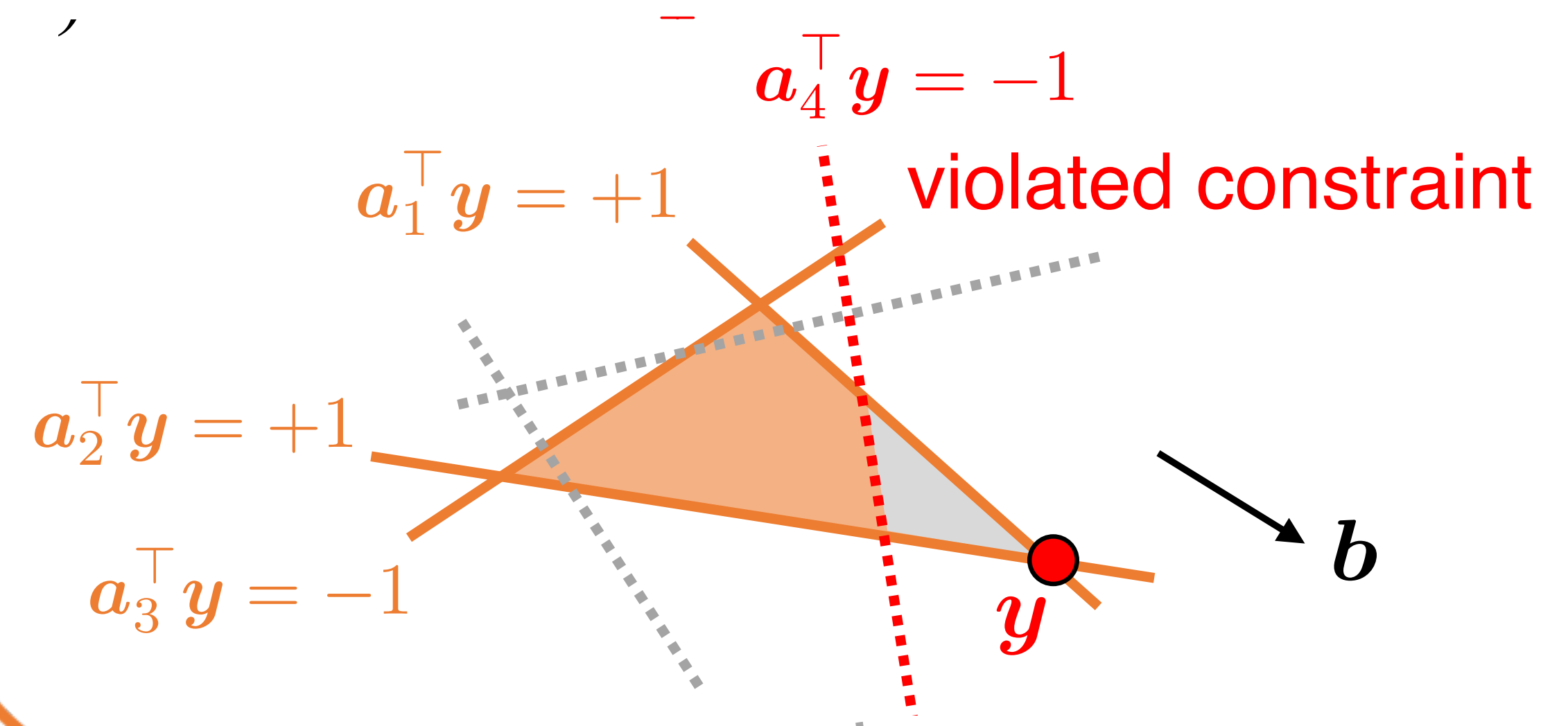


strong
duality

Dual Problem

$$\begin{aligned} \text{(D) maximize } & b^\top y \\ \text{subject to } & -1 \leq A^\top y \leq 1 \end{aligned}$$

- Limiting columns = Relaxing constraints
- Columns such that **the corresponding constraint is violated** should be added.



Step 1. Construct subset of stabilizers C from overlap info

Step 2. Repeat the following until convergence:

2.1 Solve the constrained optimization problem and obtain dual variable \hat{y}

2.2 Check all the constraints by computing $A^T \hat{y}$

2.3 Update C

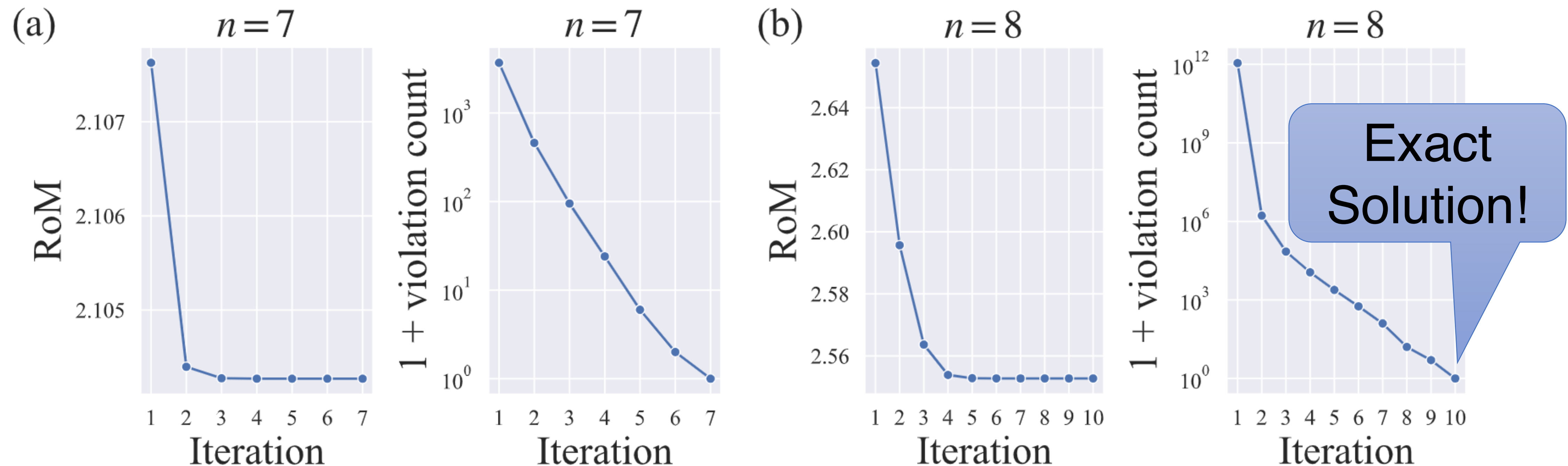
Step 1. Construct subset of stabilizers C from overlap info

Step 2. Repeat the following until convergence:

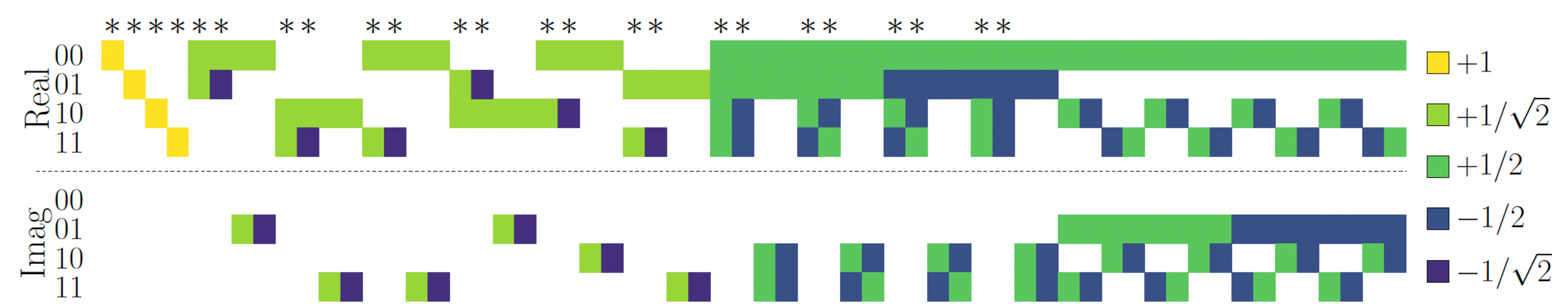
2.1 Solve the constrained optimization problem and obtain dual variable \hat{y}

2.2 Check all the constraints by computing $A^T \hat{y}$

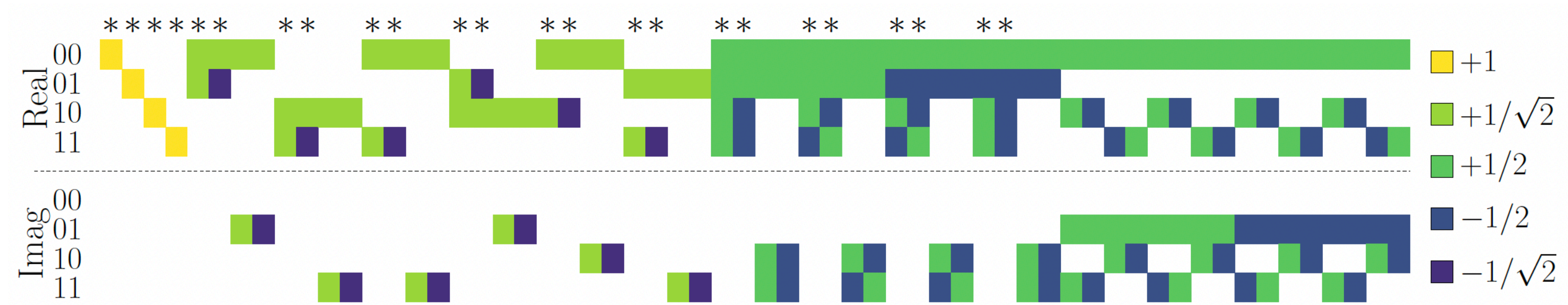
2.3 Update C



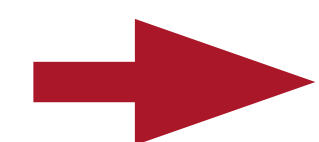
Main difference: matrix A^{SE} is quite a mess. No hope for FWHT.



Main difference: matrix A^{SE} is quite a mess. No hope for FWHT.



But... the now the small overlapping states are not contributing. Do not need all the overlaps.



Branch and bound method

Contribution 1: Proposal of new canonical form De Haene and Moore ('03), van den Nest ('10), Struchalin et al. ('21)

$$\mathcal{S}_n = \cup_{k=0}^n \mathcal{S}_{n,k} \quad \text{with}$$

$$\mathcal{S}_{n,k} := \left\{ \frac{1}{2^{k/2}} \sum_{x=0}^{2^k-1} (-1)^{x^\top Q x} i^{c^\top x} |Rx + t\rangle \mid Q \in \mathcal{Q}_k, c \in \mathbb{F}_2^k, R \in \mathcal{R}_k, t \in \mathcal{T}_R \right\}$$

where

$$\mathcal{Q}_k := \{Q \mid Q \in \mathbb{F}_2^{k \times k} \text{ is an upper triangular matrix}\},$$

$$\mathcal{R}_k := \{R \mid R \in \mathbb{F}_2^{n \times k} \text{ is a reduced column echelon form matrix with } \text{rank}(R) = k\},$$

$$\mathcal{T}_R := \{t \mid t \in \mathbb{F}_2^n \text{ is a representative of element in the quotient space } \mathbb{F}_2^n / \text{Im}(R)\}$$

In this notation, the overlap between target $|\psi\rangle$ and $|\phi_j\rangle \in \mathcal{S}_{n,k}$ is

$$|\langle \phi_j | \psi \rangle| = \left| \frac{1}{2^{k/2}} \sum_{x=0}^{2^k-1} \left((-1)^{x^\top Q x} i^{c^\top x} \right)^\dagger \langle Rx + t | \psi \rangle \right|$$

$$\langle \phi_j | \psi \rangle = \sum_{x=0}^{2^n-1} (-1)^{x^\top Q x} i^{c^\top x} P_x$$
$$Q = \begin{bmatrix} Q_{00} & Q_0^\top \\ 0 & \overline{Q} \end{bmatrix} \quad c = \begin{bmatrix} c_0 \\ \overline{c} \end{bmatrix} \quad x = \begin{bmatrix} x_0 \\ \overline{x} \end{bmatrix}$$

$$\langle \phi_j | \psi \rangle = \sum_{x=0}^{2^n-1} (-1)^{x^\top Q x} i^{c^\top x} P_x \quad Q = \begin{bmatrix} Q_{00} & Q_0^\top \\ 0 & \bar{Q} \end{bmatrix} \quad c = \begin{bmatrix} c_0 \\ \bar{c} \end{bmatrix} \quad x = \begin{bmatrix} x_0 \\ \bar{x} \end{bmatrix}$$

$$= \sum_{\bar{x}=0}^{2^{n-1}-1} (-1)^{\bar{x}^\top \bar{Q} \bar{x}} i^{\bar{c}^\top \bar{x}} \left(P_{2\bar{x}} + (-1)^{Q_{00} + Q_0^\top \bar{x}} i^{c_0} P_{2\bar{x}+1} \right) \quad \text{Cost: } O(2^{n-1}n)$$

$$\langle \phi_j | \psi \rangle = \sum_{x=0}^{2^n-1} (-1)^{x^\top Q x} i^{c^\top x} P_x \quad Q = \begin{bmatrix} Q_{00} & Q_0^\top \\ 0 & \bar{Q} \end{bmatrix} \quad c = \begin{bmatrix} c_0 \\ \bar{c} \end{bmatrix} \quad x = \begin{bmatrix} x_0 \\ \bar{x} \end{bmatrix}$$

$$= \sum_{\bar{x}=0}^{2^{n-1}-1} (-1)^{\bar{x}^\top \bar{Q} \bar{x}} i^{\bar{c}^\top \bar{x}} \left(P_{2\bar{x}} + (-1)^{Q_{00} + Q_0^\top \bar{x}} i^{c_0} P_{2\bar{x}+1} \right) \quad \text{Cost: } O(2^{n-1}n)$$

$$= \sum_{\bar{x}=0}^{2^{n-1}-1} (-1)^{\bar{x}^\top \bar{Q} \bar{x}} i^{\bar{c}^\top \bar{x}} \bar{P}_{\bar{x}} \quad \text{In-place calculation, Time complexity } O(2^n n^2)$$

$$\langle \phi_j | \psi \rangle = \sum_{x=0}^{2^n-1} (-1)^{x^\top Q x} i^{c^\top x} P_x \quad Q = \begin{bmatrix} Q_{00} & Q_0^\top \\ 0 & \bar{Q} \end{bmatrix} \quad c = \begin{bmatrix} c_0 \\ \bar{c} \end{bmatrix} \quad x = \begin{bmatrix} x_0 \\ \bar{x} \end{bmatrix}$$

$$= \sum_{\bar{x}=0}^{2^{n-1}-1} (-1)^{\bar{x}^\top \bar{Q} \bar{x}} i^{\bar{c}^\top \bar{x}} \left(P_{2\bar{x}} + (-1)^{Q_{00} + Q_0^\top \bar{x}} i^{c_0} P_{2\bar{x}+1} \right) \quad \text{Cost: } O(2^{n-1}n)$$

$$= \sum_{\bar{x}=0}^{2^{n-1}-1} (-1)^{\bar{x}^\top \bar{Q} \bar{x}} i^{\bar{c}^\top \bar{x}} \bar{P}_{\bar{x}} \quad \text{In-place calculation, Time complexity } O(2^n n^2)$$

$$\max_j |\langle \phi_j | \psi \rangle| = \max_{\bar{Q}, \bar{c}} \max_{Q_{00}, Q_0, c_0} \left| \sum_{\bar{x}=0}^{2^{n-1}-1} (-1)^{\bar{x}^\top \bar{Q} \bar{x}} i^{\bar{c}^\top \bar{x}} \bar{P}_{\bar{x}} \right| \quad \text{Time complexity } O(2^{n+n(n+1)/2}) \sim O(|S_n|)$$

Some branches do not need explicit calculation

(Branch and bound)

RoM for 8 qubits, SE for 9 qubits

qubit count n	5	6	7	8	9
states $ \mathcal{S}_n $	2.4×10^6	3.2×10^8	8.1×10^{10}	4.2×10^{13}	4.3×10^{16}
size of A_n^{RoM}	379 MiB	95 GiB	86 TiB	86 PiB	172 EiB
RoM naive time	2 min	×	×	×	×
our time	2.3 s	7.0 min	1.6 h	2.0 d	×
size of A_n^{SE}	1011 MiB	254 GiB	153 TiB	153 PiB	305 EiB
SE naive time	7.7 min	×	×	×	×
our time	1.5 s	3.8 s	12.9 s	8.8 min	19.2 h

Future directions

- Application to extent-based monotone for mixed states? (e.g. dyadic negativity)
- Incorporate symmetry in stabilizer extent calculation
- Develop integrated library?