PHOENIX: Pauli-Based High-Level Optimization Engine for Instruction Execution on NISQ Devices

Zhaohui Yang[1], Dawei Ding[2], Chenghong Zhu[3], Jianxin Chen[4], Yuen Xie[1]

[1]Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong
[2]Yau Mathematical Sciences Center, Tsinghua University, Beijing, China
[3]The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China
[4]Department of Computer Science and Technology, Tsinghua University, Beijing, China

62 THE CHIPS TO SYSTEMS CONFERENCE

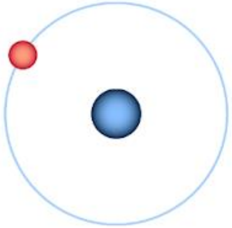
SPONSORED BY CEDA sigda

Quantum computing for Hamiltonian simulation problems

Goal: Molecular properties of drugs and materials

Difficult: Exponential state space requirements

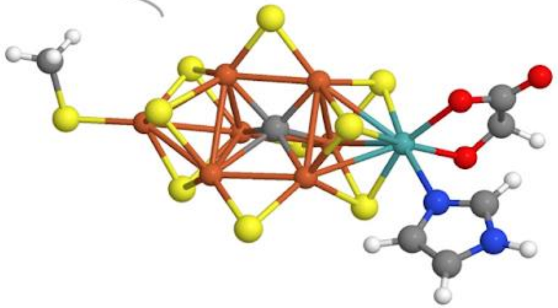
Use a QC to simulate a quantum system!



$$H = \begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} \\ \alpha_{2,1} & \alpha_{2,2} \end{pmatrix}$$

Hamiltonian describes allowed energy levels of the system

Increasing complexity

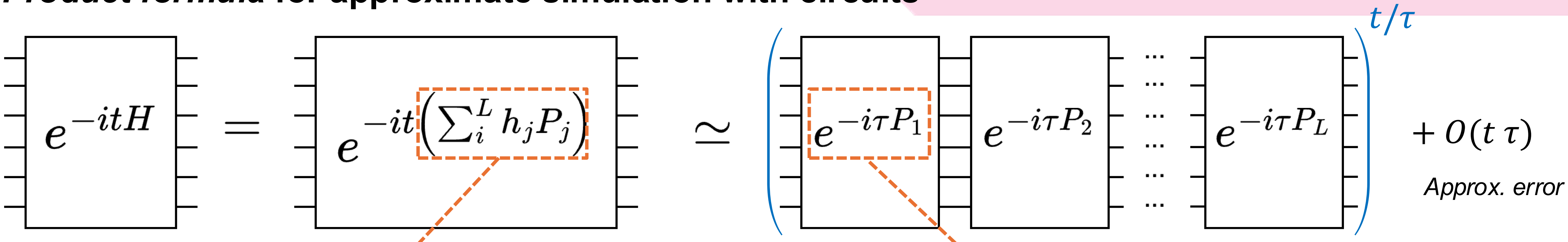


$$H = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,2^n} \\ \vdots & \ddots & \vdots \\ \alpha_{2^n,1} & \cdots & \alpha_{2^n,2^n} \end{pmatrix}$$

Figure: Reiher et al. 2017, DOI:10.1073/pnas.1619152114

How to simulate the unitary evolution governed by a system Hamiltonian?

Product formula **for approximate simulation with circuits**



$$H = \sum_{j=1}^{L} h_j P_j = \sum_{j=1}^{L} h_j \sigma_0^{(j)} \otimes \cdots \otimes \sigma_{n-1}^{(j)}$$

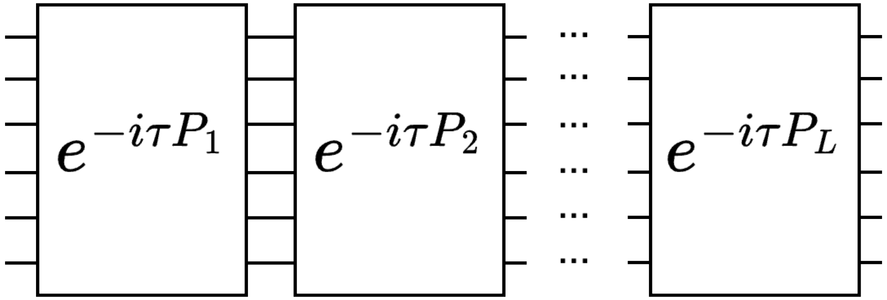
Hamiltonian as "linear combination" of Pauli operators

Each Pauli exponential can be directedly synthesized by basic 1Q and 2Q gates

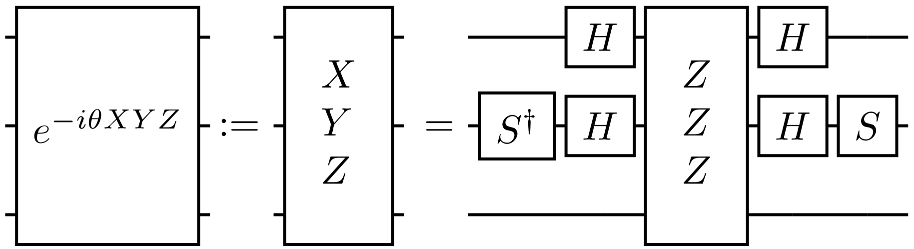
σ_i is basic 2x2 Pauli matrix:

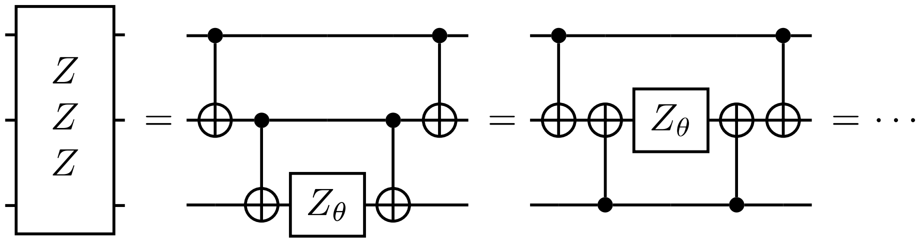
I= $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, X= $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, Y= $\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$, or Z= $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Basic synth. of Pauli exponentials (IRs)



- *Pauli exponential as IR (intermediate representation)* $P_j = h_j \sigma_0^{(j)} \otimes \cdots \otimes \sigma_{n-1}^{(j)}$
- *IR synthesis → basic 1Q and 2Q gates*

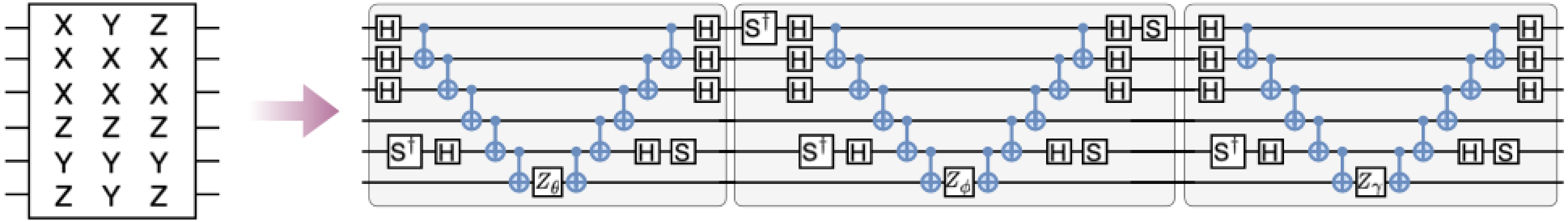




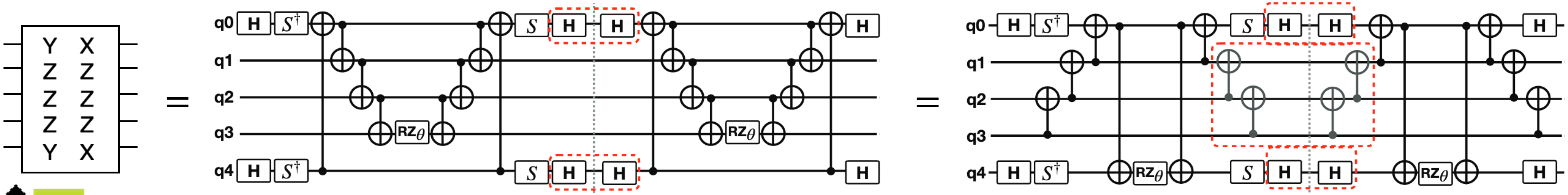
Problem statement for IRs synth. & Opt.

- Compilation goal: As less basic quantum gates (especially 2Q gates) as possible

E.g.,
Naïve synth.



- Previous optimization methods
  - Gate cancellation opportunities through 1) variational IR arrangement, 2) IR synth. variants
  - E.g., [TKet, Cowtan+2020], [PauliOpt, Griend+2023], [Paulihedral, Li+ASPLOS'22], [Tetris, Jin+ISCA'24]



E.g., Synth. Strategy in Paulihedral/Tetris

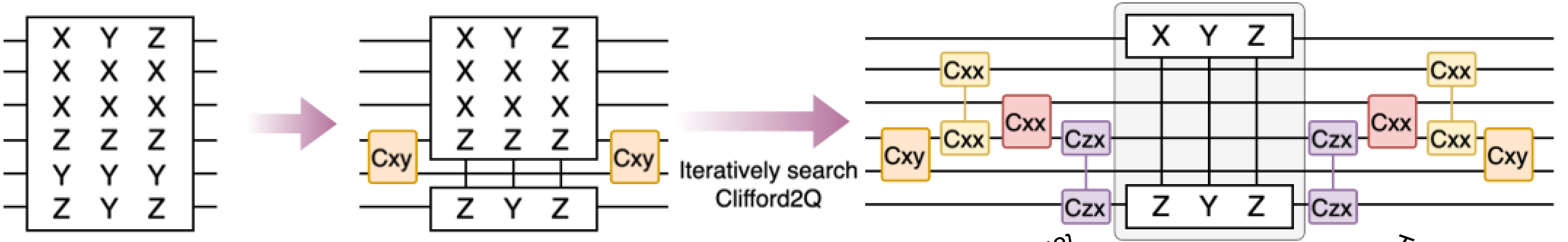
Local optimization (limited opt. space & complicated tricks)

Dependent on CNOT tree unrolling

Is there any efficient synthesis approach?

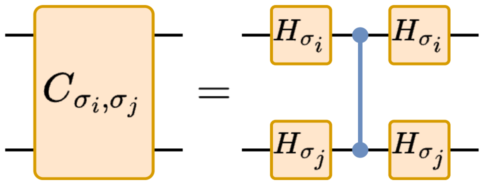
Insight of our optimization method

Simultaneous Pauli strings simplification via Clifford conjugations

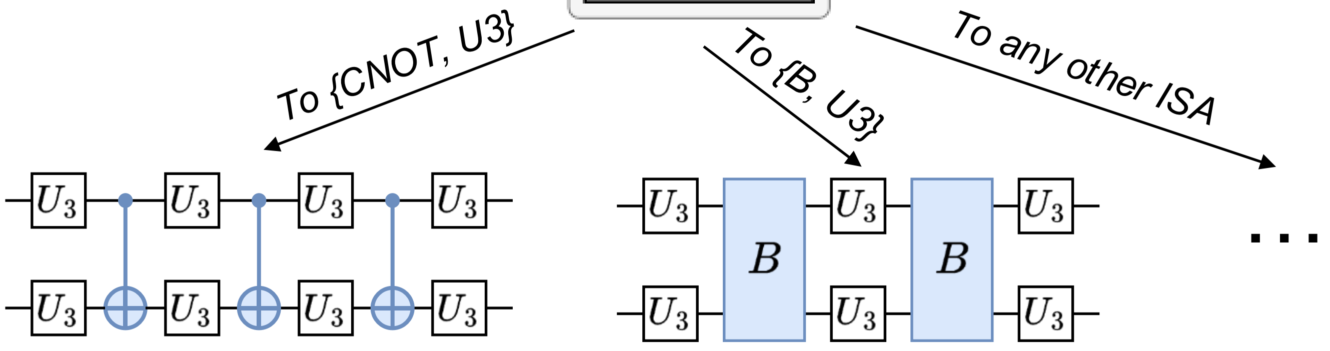


C_{σ_i,σ_j} means some Clifford2Q operator

E.g.,



$H_Z := I, H_X := H, H_Y := S\ H\ S^\dagger$



Highly effective
(e.g., 8+3 = 11 CNOTs v.s. 30 CNOTs via naïve synth)

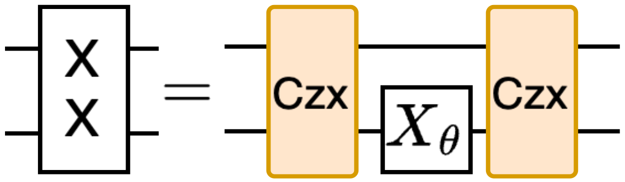
Global opt.:
Simult. Paulis simp.

ISA independent

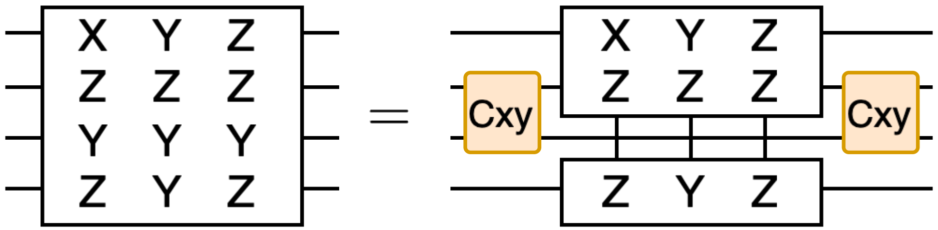
Problem formulation

Reformulate the synthesis process: ***Pauli exp. synthesis → Clifford transformation on Paulis***

- Clifford formalism

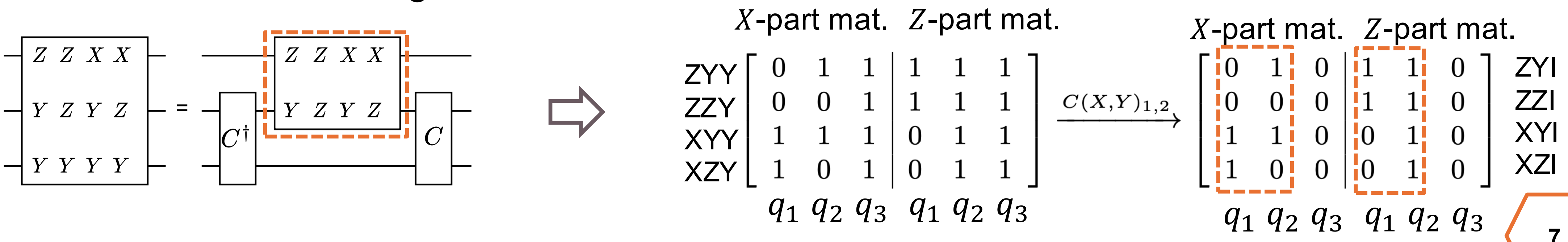


One Pauli string → another Pauli string



A set of Pauli string → another set of Pauli strings

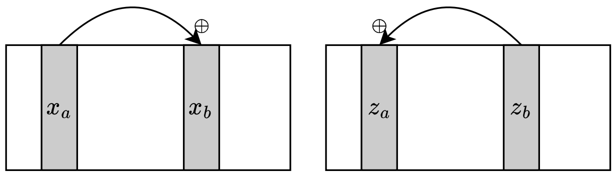
- Formal IR description: Binary symplectic form (BSF)
  - j-th component of i-th Pauli → $[X_{i,j}; Z_{i,j}]$ (e.g., X is $[1,0]$, Y is $[1,1]$)
  - Accommodate Global High-level info.



Problem formulation

Reformulate the synthesis process: ***Pauli exp. synthesis → Clifford transformation on Paulis***

- Clifford formalism as binary operation on column vectors of BSF tableau

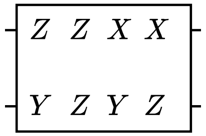


E.g., CNOT transformation

$$\left[\begin{array}{ccc|ccc} 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{array}\right] \xrightarrow{C(X,Y)_{1,2}} \left[\begin{array}{ccc|ccc} 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

E.g., Clifford $C(X,Y)$: $[x_a, x_b \mid z_a, z_b] \to [x_a \oplus x_b \oplus z_b, z_a \oplus z_b \mid z_a, z_a \oplus z_b]$

- Optimization goal:
  - When $w_{\text{tot.}} := \left|\bigvee_i \left(r_x^{(i)} \vee r_z^{(i)}\right)\right|$ is at most 2 (directly synthesized by basic 1Q/2Q gates)



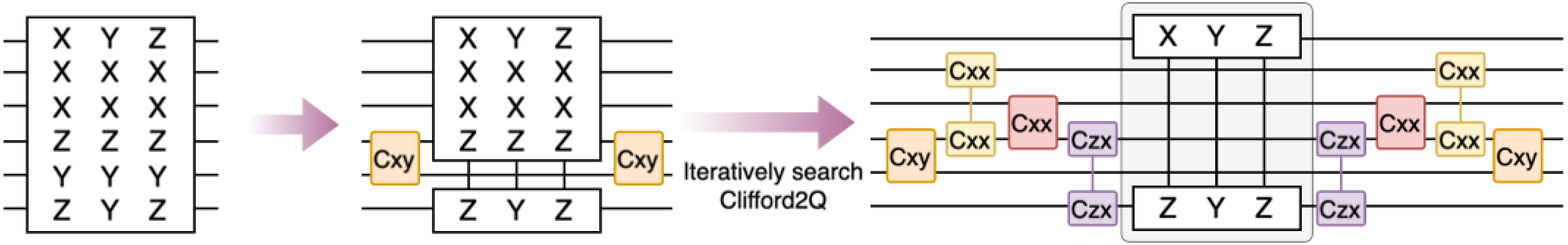
$$\left[\begin{array}{ccc|ccc} 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

E.g., $w_{\text{tot.}} = 2$

  - Now the key is: **How to search for the most appropriate Clifford2Q for BSF simplification?**

BSF simplification algorithm

- A cost function: disparity between current BSF and optimization goal
  - A sophisticated metric heuristically designed. See paper for details.
- Heuristic Clifford2Q search: Greedily search for the most appropriate one
  - Select from $\left\{C_{\sigma_i,\sigma_j}\right\} \times \{(q_m, q_n)\}$ that mostly minimize the cost function
  - Apply the selected Clifford2Q; Peel 1Q Pauli rotations before each search step
  - Iterate, until $w_{\text{tot.}}$ is less than 2



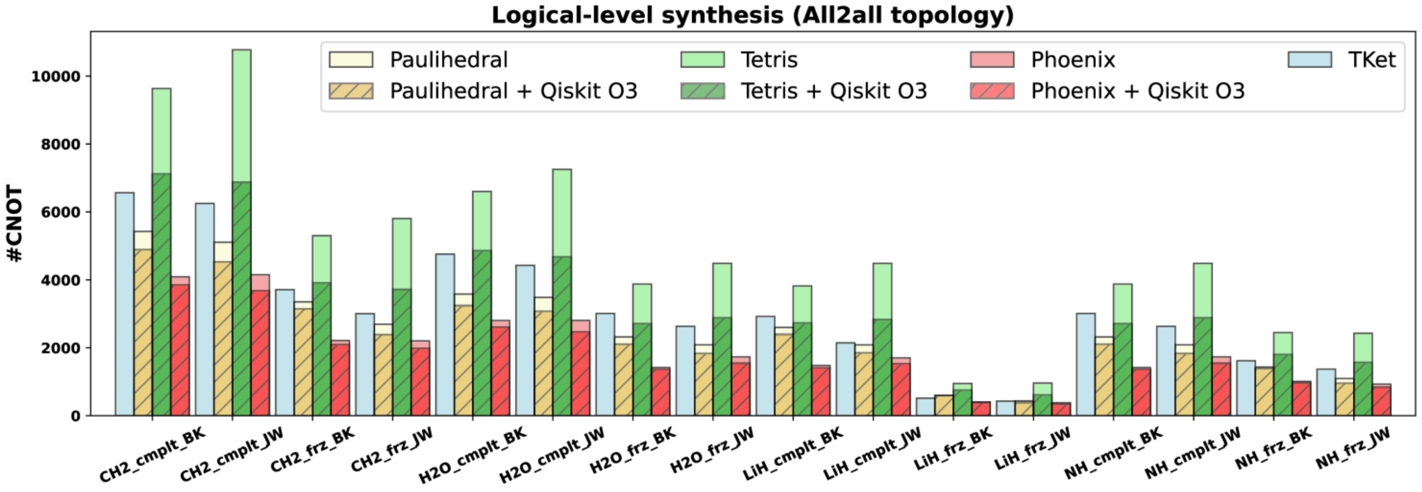
E.g., Targe unitary evolution (snippet from H4 UCCSD):
$e^{-i(0.1*XXXZYZ+0.2*YXXZYY+0.3*ZXXZYZ)}$

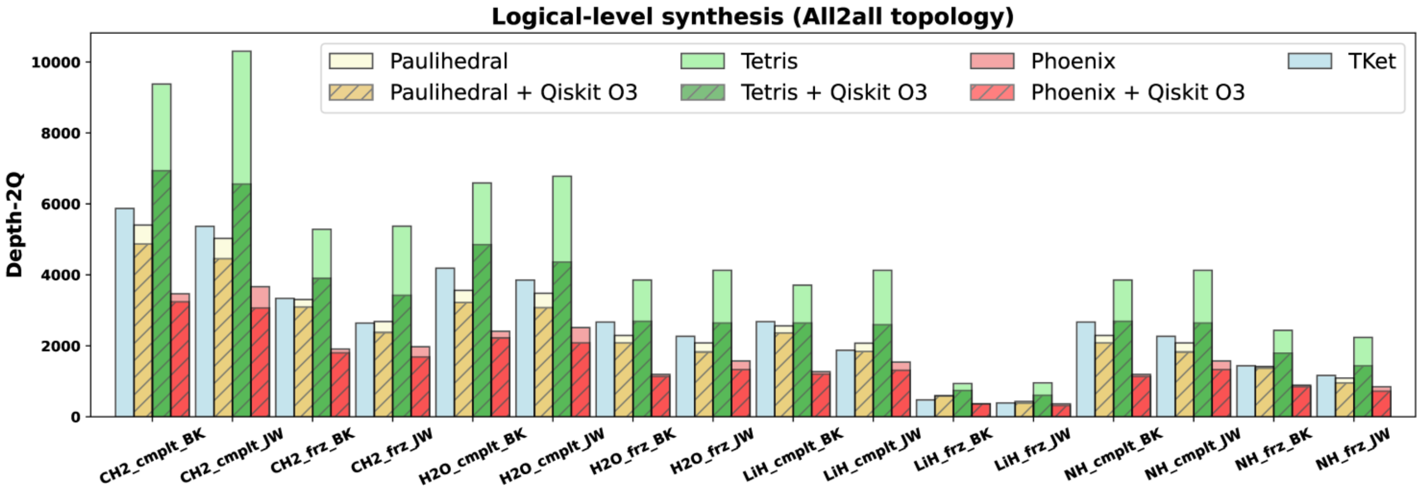
Final result

Moreover

- In PHOENIX (Pauli-based High-level Optimization ENgine for Instruction eXecution), we design
  - BSF as formal description (preserve high-level global info of IRs)
  - Formulate IR synth. & opt. as "simultaneously lowering weights" of BSF tableau by Clifford2Q (global opt.; ISA-independent)
  - BSF simplification algorithm as the core optimization pass (highly effective; polynomial complexity)
  - Extra opt. points: Assemble simplified IR groups, gate cancellation, lowering depth
- See paper for details

Evaluation: Main results





AVERAGE (GEOMETRIC-MEAN) OPTIMIZATION RATES ON UCCSD.

Compiler	#CNOT opt.	Depth-2Q opt.
TKET	33.07%	30.14%
PAULIHEDRAL	28.41%	29.07%
PAULIHEDRAL + O3	25.72% (-8.54% v.s. no O3)	26.3% (-8.6% v.s. no O3)
TETRIS	53.66%	53.26%
TETRIS + O3	36.73% (-30.94% v.s. no O3)	36.37% (-31.08% v.s. no O3)
PHOENIX	**21.12%**	**19.29%**
PHOENIX + O3	**19.53%** (**-6.64%** v.s. no O3)	**17.28%** (**-8.51%** v.s. no O3)

- Hardware-agnostic compilation benchmarking
  - Baselines: [Tket, Cowtan+2020], [Paulihedral, Li+ASPLOS'22), [Tetris, Jin+ISCA'24]
- Phoenix significantly outperforms other SOTAs
- Phoenix's high-level optimization leaves the least optimization space for local optimization (Qiskit O3)

Benchmarks: 16 molecule simulation programs

Conclusion

- Contributions: PHOENIX, a high-level VQA application-specific compiler
  - Formal description by BSF; Problem modeling via BSF tableau update by Cliford2Q; Heuristic algorithms
  - ISA-independent (CNOT/B/SQiSW; Clifford2Q could even be iSWAP-equivalent other than CZ-equivalent ones)
  - High-level & global optimization (Highly effective; scalable)
  - Outperforms other SOTAs across diverse VQA applications, device topologies, and backend ISAs (more evaluation details in paper)
- Future directions
  - Deep co-optimization (e.g., topology-aware opt.)?
  - BSF as formal description leveraged for stabilizer circuit optimization?

Paper link

Zhaohui Yang

PhD student

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Thanks for listening!

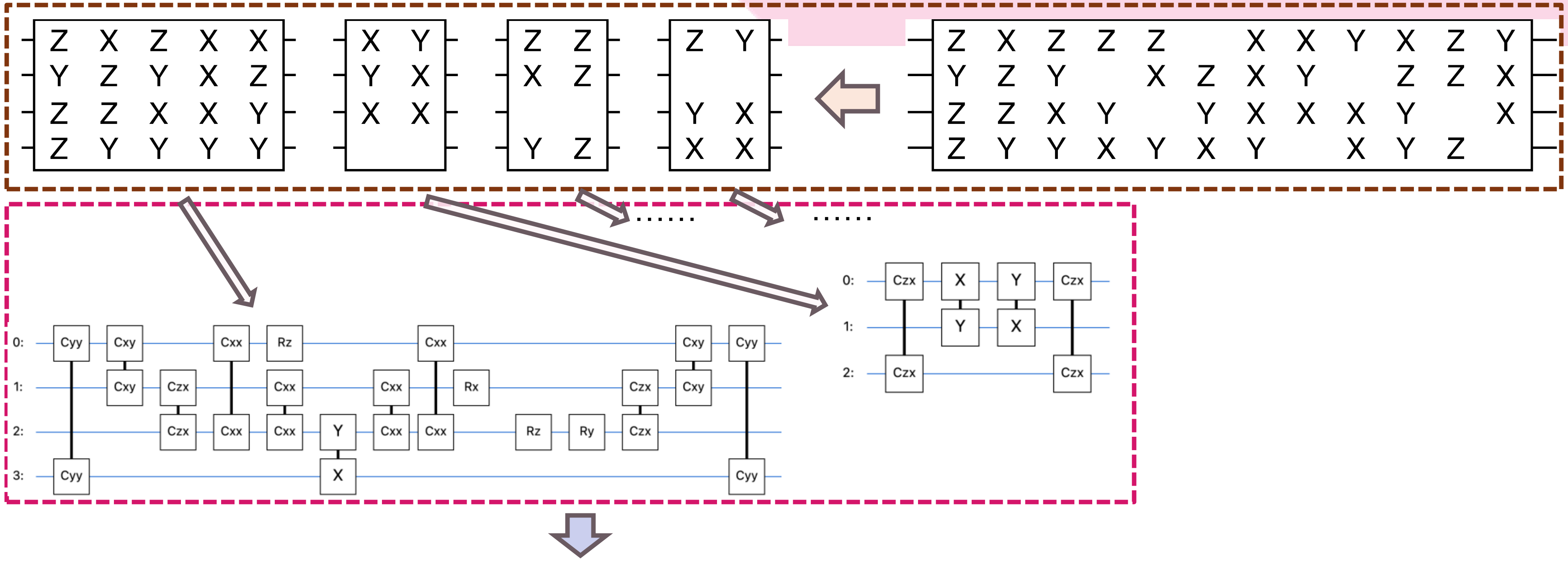
Backup slides

End-to-end workflow and further opt.

Pauli IR grouping

IR group simplification

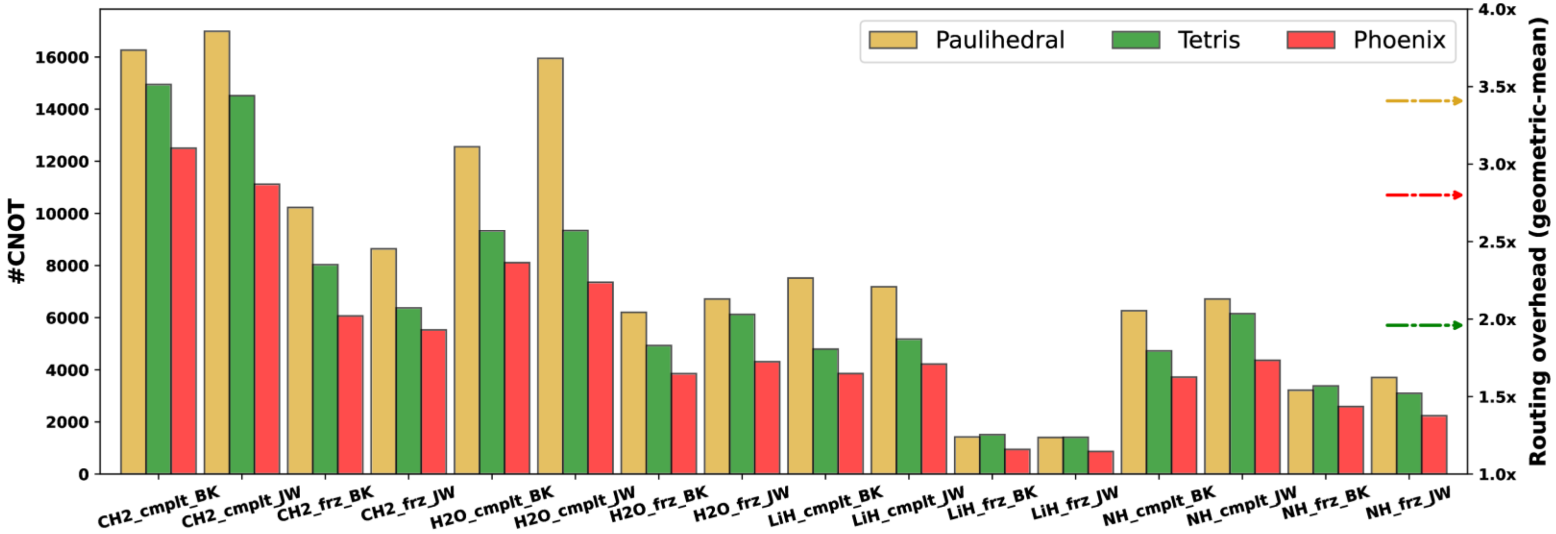
IR groups ordering



Assemble (order) simplified IRs subcircuits to exploit other opportunities, e.g., gate cancellation

Evaluation: For topology-limited devices

- On the heavy-hex topology (IBM's Manhattan), Phoenix significantly outperforms other SOTAs
  - 36.17% (22.62%) #2Q and 43.85% (28.12%) Depth2Q reduction v.s. Paulihedral (Tetris)
- Qubit routing overhead ($\#2Q_{\text{before-mapping}}/\#2Q_{\text{after-mapping}}$)
  - Paulihedral (3.4x) > Phoenix (2.8x) > Tetris (1.9x)



Evaluation: For alternative ISAs

CNOT ISA is not unique in quantum computing! Three-CNOT SWAP-based routing is not unique as well!

- More and more continuous ISAs (gate sets) are adopted (e.g., IBM's fractional gates, IonQ's partial entangling gates)
- Again, Phoenix significantly outperforms other SOTAs in SU(4) ISA (arXiv:2312.05652)
  - The advantage of Phoenix in SU(4) ISA is more impressive than that in CNOT ISA
- We show that without deep co-optimization (e.g., CNOT unrolling, SWAP-based routing), the Phoenix optimization framework proves generic advantage!

	CNOT ISA (all-to-all)		SU(4) ISA (all-to-all)		CNOT ISA (heavy-hex)		SU(4) ISA (heavy-hex)	
PHOENIX's opt. rate	#CNOT	Depth-2Q	#SU(4)	Depth-2Q	#CNOT	Depth-2Q	#SU(4)	Depth-2Q
PHOENIX v.s. TKET	63.87%	64.0%	**56.04%**	**54.22%**	**40.63%**	**48.32%**	44.29%	50.71%
PHOENIX v.s. PAULIHEDRAL	82.12%	73.33%	**75.57%**	**65.2%**	62.38%	54.7%	**39.84%**	**35.07%**
PHOENIX v.s. TETRIS	57.52%	53.04%	**56.54%**	**50.55%**	75.97%	71.18%	**62.23%**	**58.74%**

Comparison for diverse ISAs with all-to-all and limited topologies

Alternative choices of Clifford group generators

	C(Z, X)	C(Z, Y)	C(Z, Z)	C(X, Y)	C(X, X)	C(Y, Y)
XX	XI	-YZ	YY	IX	XX	ZZ
XY	YZ	XI	-YX	XY	IY	XI
XZ	-YY	YX	XI	IZ	IZ	-ZX
YX	YI	XZ	-XY	-ZZ	YI	IX
YY	-XZ	YI	XX	YI	ZZ	YY
YZ	XY	-XX	YI	ZX	-ZY	IZ
ZX	ZX	IX	IX	YZ	ZI	-XZ
ZY	IY	ZY	IY	ZI	-YZ	ZI
ZZ	IZ	IZ	ZZ	-YX	YY	XX

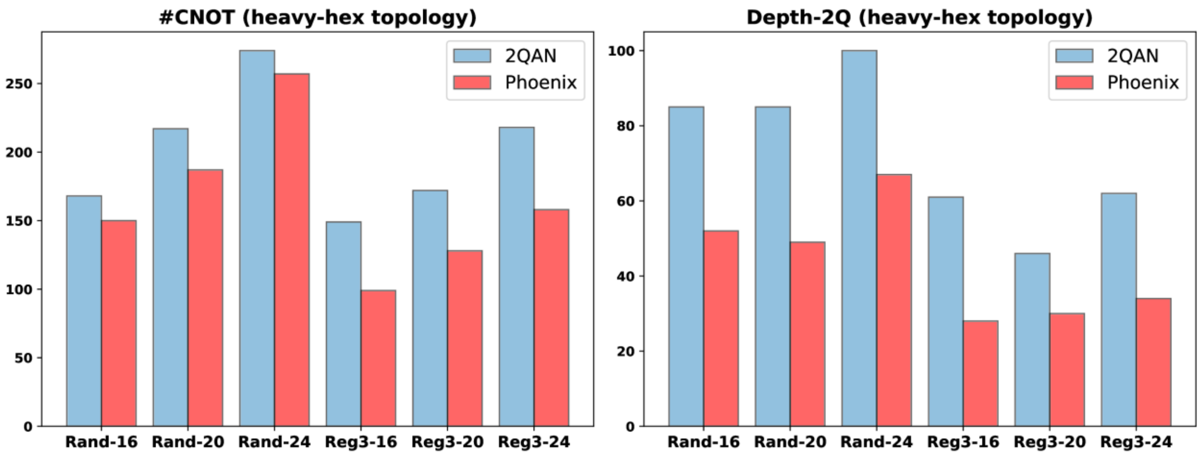
	iSWAP(X, X)	iSWAP(X, Y)	iSWAP(X, Z)	iSWAP(Y, Y)	iSWAP(Y, Z)	iSWAP(Z, Z)
XX	XX	-ZI	-YI	XX	ZY	XX
XY	ZI	XY	-ZI	-IZ	XY	YX
XZ	-YI	-YI	XZ	ZX	-IX	IY
YX	IZ	YX	-ZY	-ZI	XI	XY
YY	YY	IZ	YY	YY	-ZI	YY
YZ	ZY	-ZX	IX	XI	YZ	-IX
ZX	-IY	-YZ	ZX	XZ	ZX	YI
ZY	YZ	IX	-YX	IX	XX	-XI
ZZ	ZZ	ZZ	IY	ZZ	IY	ZZ

- Optimization effects based on iSWAP-equivalent Cliffords or CNOT-iSWAP-mixed Cliffords are comparable to our previous choice of “universal controlled gate” CNOT-equivalent Cliffords
- This provides hardware-friendly property for hardware with non-CNOT native gate sets

Evaluation: UCCSD benchmarks info

Benchmark	#Qubit	#Pauli	w_{max}	#Gate	#CNOT	Depth	Depth-2Q
CH2_cmplt_BK	14	1488	10	37780	19574	23568	19399
CH2_cmplt_JW	14	1488	14	34280	21072	23700	19749
CH2_frz_BK	12	828	10	19880	10228	12559	10174
CH2_frz_JW	12	828	12	17658	10344	11914	9706
H2O_cmplt_BK	14	1000	10	25238	13108	15797	12976
H2O_cmplt_JW	14	1000	14	23210	14360	16264	13576
H2O_frz_BK	12	640	10	15624	8004	9691	7934
H2O_frz_JW	12	640	12	13704	8064	9332	7613
LiH_cmplt_BK	12	640	10	16762	8680	10509	8637
LiH_cmplt_JW	12	640	12	13700	8064	9342	7616
LiH_frz_BK	10	144	9	2890	1442	1868	1438
LiH_frz_JW	10	144	10	2850	1616	1985	1576
NH_cmplt_BK	12	640	10	15624	8004	9691	7934
NH_cmplt_JW	12	640	12	13704	8064	9332	7613
NH_frz_BK	10	360	9	8303	4178	5214	4160
NH_frz_JW	10	360	10	7046	3896	4640	3674

Evaluation: QAOA benchmarking



QAOA BENCHMARKING VERSUS 2QAN.

QAOA		#CNOT		Depth-2Q		#SWAP		Routing overhead	
Bench.	#Pauli	2QAN	Phoenix	2QAN	Phoenix	2QAN	Phoenix	2QAN	Phoenix
Rand-16	32	168	150	85	52	37	29	2.62x	2.34x
Rand-20	40	217	187	85	49	47	39	2.71x	2.34x
Rand-24	48	274	257	100	67	63	56	2.85x	2.68x
Reg3-16	24	149	99	61	28	44	17	3.10x	2.06x
Reg3-20	30	172	128	46	30	46	23	2.87x	2.13x
Reg3-24	36	218	158	62	34	62	30	3.03x	2.19x
Avg. improv.		-16.7%		-40.8%		-29.41%		-16.59%	

Evaluation: Algorithmic error analysis

- Algorithmic error (disparity between circuit and ideal evolution)
  - E.g., $\mathrm{infid} = 1 - \frac{1}{N}|\mathrm{Tr}(U^\dagger V)|$