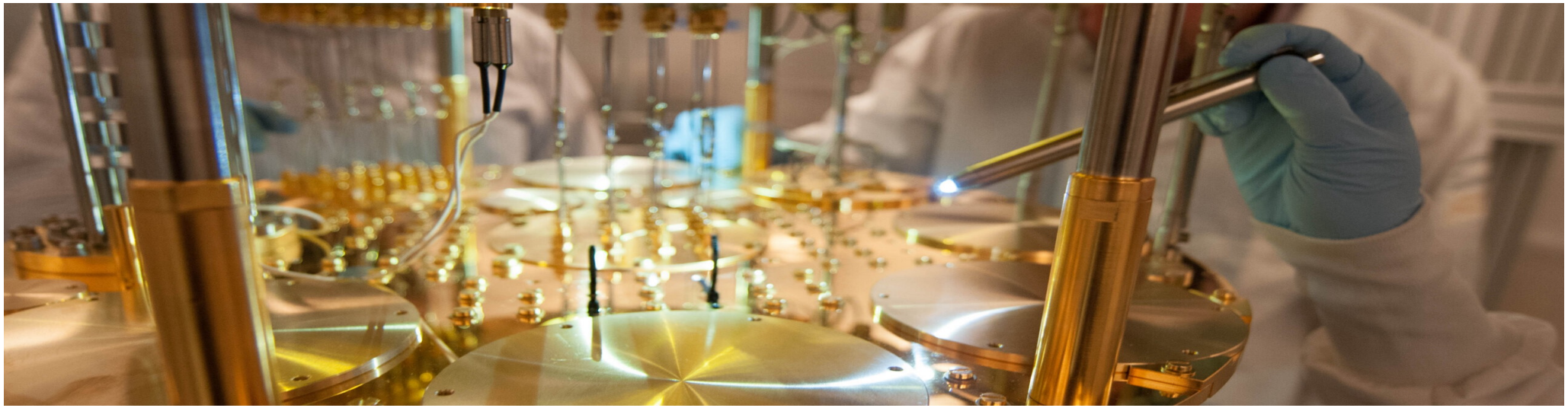


Exploring the Quantum Design Space with Circuit Synthesis

Costin Iancu – Lawrence Berkeley National Laboratory
cciancu@lbl.gov

BQSKit

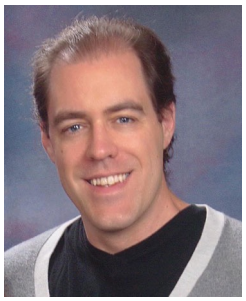
⟨AIDE|QC⟩

U.S. DEPARTMENT OF ENERGY
Office of Science

BERKELEY LAB
Bringing Science Solutions to the World

Team

Ed Younis



Wim Lavrijsen

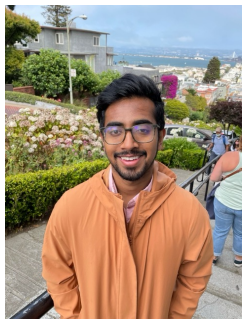
Marc Davis

Mathias Weiden

Alon Kukliansky

Ji Liu

Aaron Szasz



Justin Kallor

Siyuan Niu

Roel van Beeumen

Bert de Jong

Past:

Lindsay Bassman
Xin-Chuan Ryan Wu (Intel)
Tirthak Patel (Rice)
Ethan Smith

BQSKit – Berkeley Quantum Synthesis Toolkit

(biss-kit)

Quantum “compilation” infrastructure for NISQ and FT:

- Portable across gates sets (NISQ and FT), QPU architectures
- Supports higher dimension abstractions (multi-qubit gates, qutrits etc.)
- Manipulates circuits in multiple parameterized and concrete encodings
- Provides error mitigation

1. Generates resource efficient circuits
2. Enables hardware and algorithm design exploration

BQSKit

https://github.com/BQSKit - 200K+ downloads

3 IEEE QCE Best Paper Awards (2020, 2021, 2022)

bqskit.lbl.gov

Users:

- LANL
- Sandia
- ANL
- ORNL
- U Chicago
- U Kansas
- U Tokyo
- NCSU
- Duke
- CUNY
- AWS - Braket
- Microsoft
- NVIDIA
- UnitaryFund
- HSBC
- ColdQuanta/Infleqtion(Superstaq)

Turn-key functionality, works “everywhere”

Python 3.8+ on Windows, Mac, Linux
(from laptop to supercomputer)

Any gate set in or out, qubits, qutrits, states, many-states, unitaries
(from experiment to programs)

```
pip install bqskit
```

```
import bqskit
out_circuit = bqskit.compile(circuit)
```

Configurable Compilation Workflows

```python
from bqskit import Circuit
circuit = Circuit.from_file('in.qasm')

from bqskit import MachineModel
from bqskit.ir.gates import CZGate, U3Gate
gate_set = {CZGate(), U3Gate()}
model = MachineModel(64, gate_set=gate_set)

from bqskit import compile
out_circuit = compile(circuit, model)

out_circuit.save('out.qasm')
```

Turn-key (-O1 .. -O4):

- No user-defined rules necessary; all gates support
- Simply load a circuit, a predefined gate set, and compile to it.

Configurable:

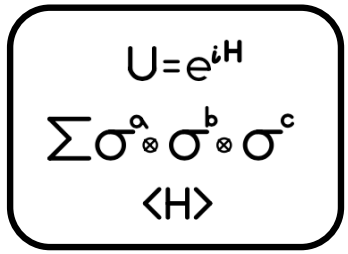
- `MachineModel` **(topology)**
- `ir.gates` **(gate set)**
- `Compile()` **(transformation workflow)**

Berkeley Quantum Synthesis Toolkit Tutorial

github.com/bqskit/bqskit-tutorial

Mathematical Representations

Function: fixed
Structure: none

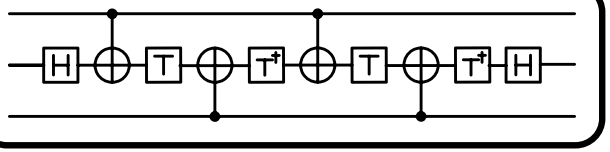


Mathematical Representations

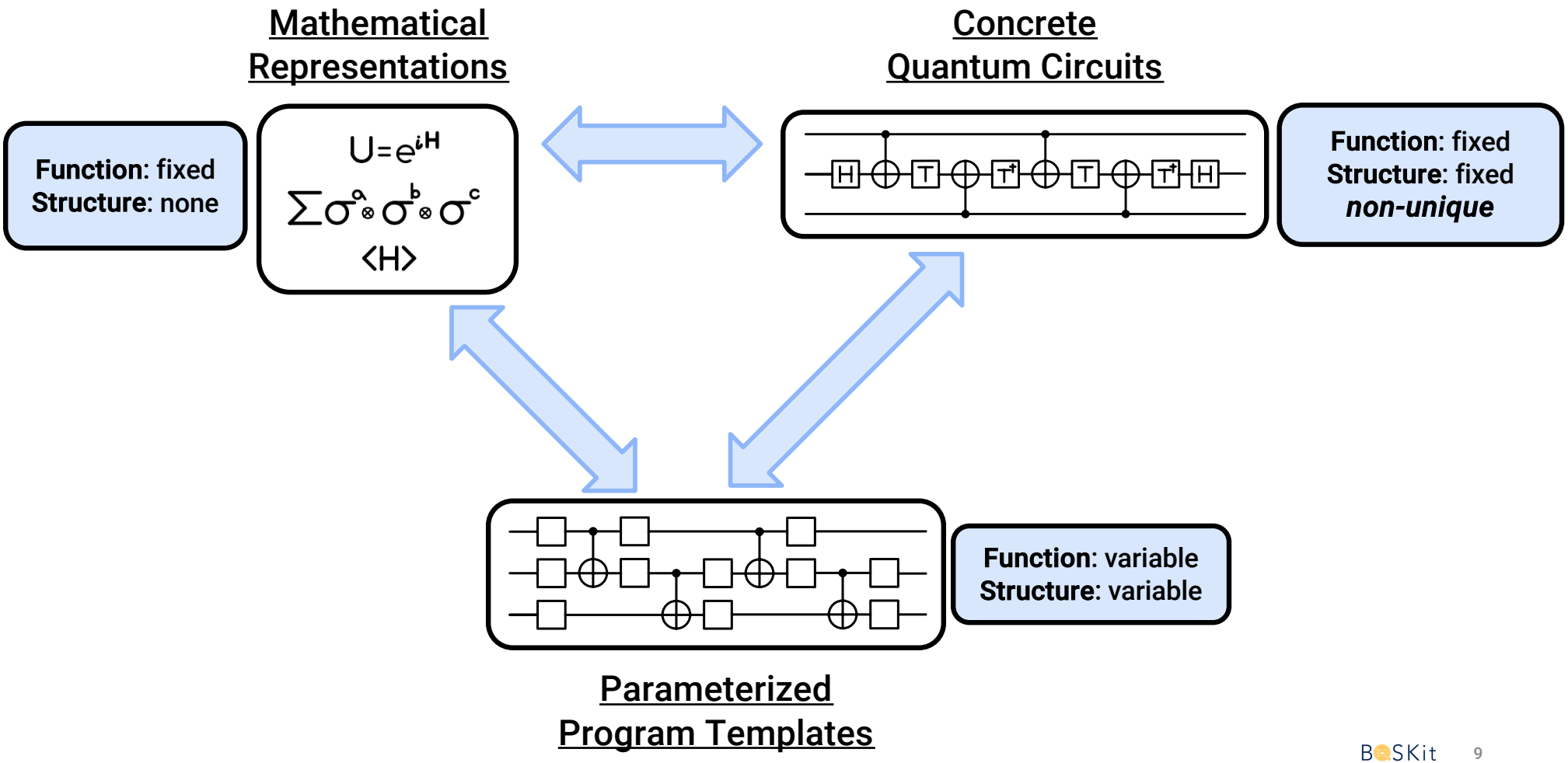
Function: fixed
Structure: none

$$U = e^{iH}$$
$$\sum \sigma^a \otimes \sigma^b \otimes \sigma^c$$
$$\langle H \rangle$$

Concrete Quantum Circuits



Function: fixed
Structure: fixed
non-unique


Mathematical Representations
Concrete Quantum Circuits
Function: fixed
Structure: none
U=e^{iH}
∑σ^a ⊗ σ^b ⊗ σ^c
⟨H⟩
H
T
T†
T
T†
H
Function: fixed
Structure: fixed
non-unique
Function: variable
Structure: variable
Parameterized Program Templates

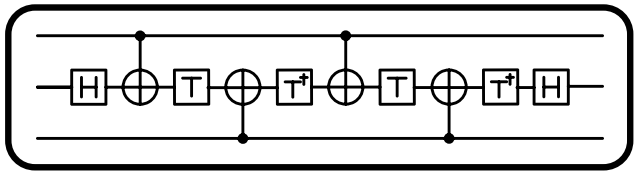
Mathematical Representations

$U = e^{iH}$

$\sum \sigma^a \otimes \sigma^b \otimes \sigma^c$

$\langle H \rangle$

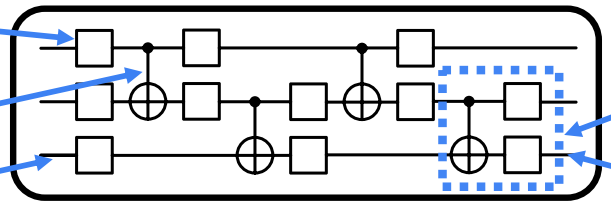
Concrete Quantum Circuits



Parameterized qubit gates (U3, RZ, B...)

Fixed gates (CNOT, iSWAP..)

Qutrits, qudits, multi-qubit SU(n), mixed radix...



Parameterized Circuit Templates

Arbitrary Atomic Unit

- Unitary

$$\frac{1}{2}\begin{bmatrix} i\sqrt{2} & 0 & 0 & \sqrt{2} \\ 0 & 1-i & 1-i & 0 \\ -i\sqrt{2} & 0 & 0 & \sqrt{2} \\ 0 & -1+i & 1-i & 0 \end{bmatrix}$$

- Pauli operators

$$F(Q, \vec{\alpha}) = \exp(\mathrm{i}(\vec{\alpha} \cdot \pi_Q(\sigma^{\vec{\otimes} m})))$$

Numerical Instantiation with BQSKit

Instantiation: Given a parameterized quantum circuit $C : \mathbb{R}^k \mapsto U(N)$ and a target unitary $V \in U(N)$, solve

$$\underset{\alpha}{\text{argmax}}\ tr(V^\dagger C(\alpha))$$

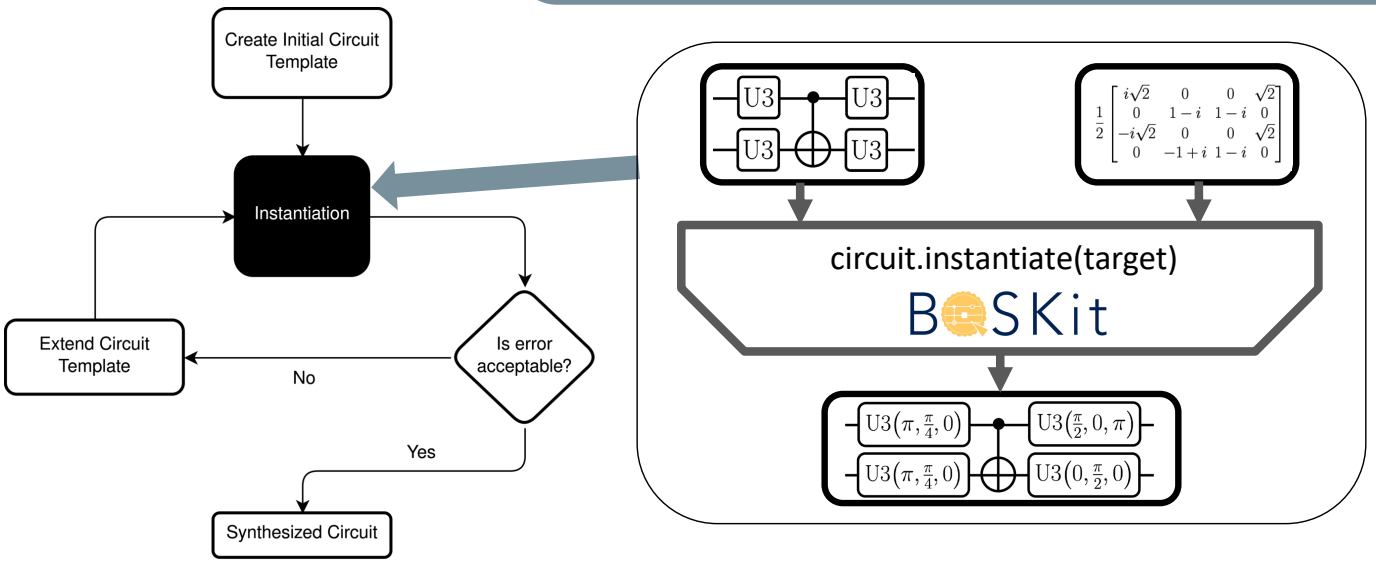
Configurable objective functions

$$\Delta(C(\alpha), V) = 1 - \frac{|tr(V^\dagger C(\alpha))|}{N}$$

$$\Delta_f(C(\alpha), V) = 1 - \frac{Re(tr(V^\dagger C(\alpha)))}{N}$$

$$\Delta_p(C(\alpha), V) = \sqrt{1 - \frac{|tr(V^\dagger C(\alpha))|^2}{N^2}}$$

...



General instantiation workflow enables the optimization, synthesis, and transpilation of quantum circuits

Unitary Synthesis: Generation and Optimization

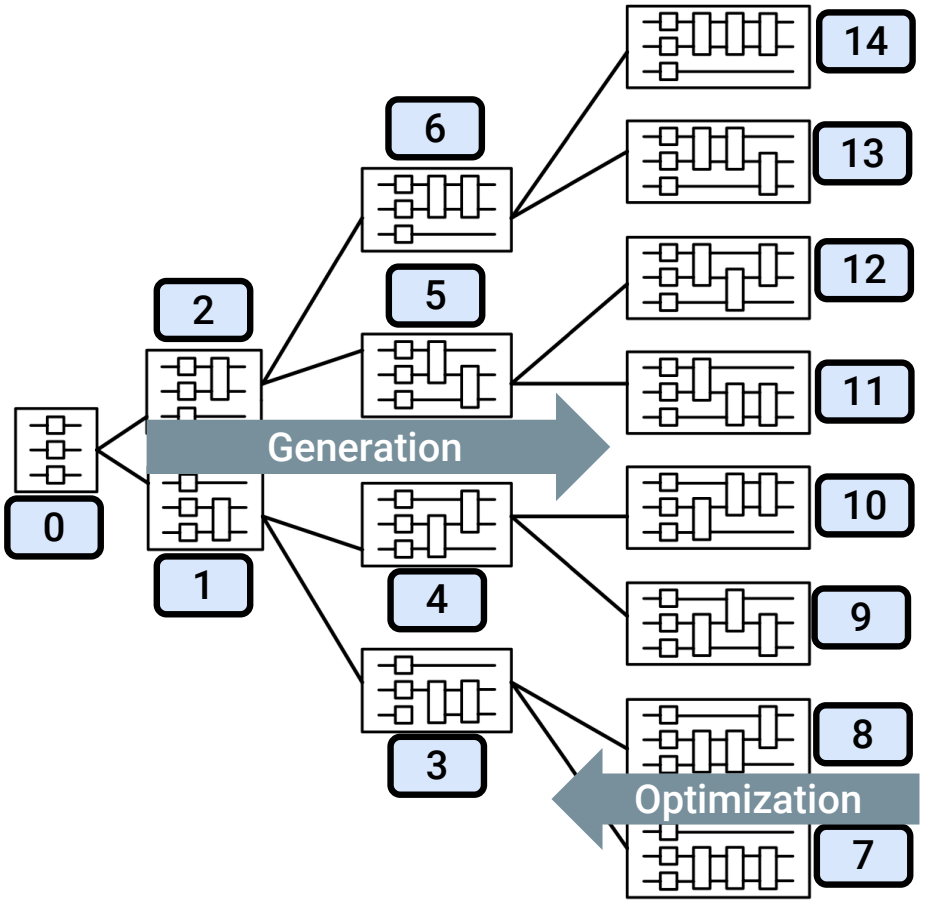
Consider tree of possible circuits that can implement an unitary

Transformation algorithm implements search on tree, instantiate at each step

- Generation: build circuits bottom-up, e.g. synthesis
- Optimization: process top-down, e.g. delete gate

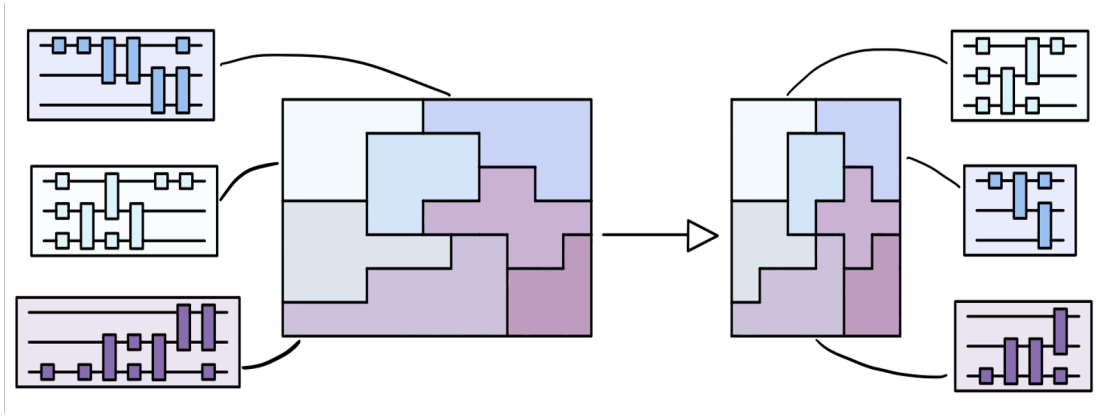
Stop when finding the first “good” solution (multi-objective)

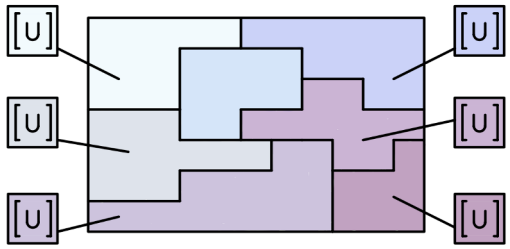
- Gate count (e.g. CNOT, Clifford+T etc.)
- Depth, gate parallelism
- Heterogenous gates mix
- Defects, patterns



Scaling With Qubits

- Unitary instantiation has width/depth limits
  - QFactor scales to 12 (+) qubits!
- Wide circuits are broken into manageable panels
  - Each panel has an associated partitioned unitary
  - In practice, partitions/panels of 3 qubits are a good tradeoff quality vs runtime overhead





Partitioned unitaries are taken from circuit panels

Circuit partitions can be handled independently and in parallel

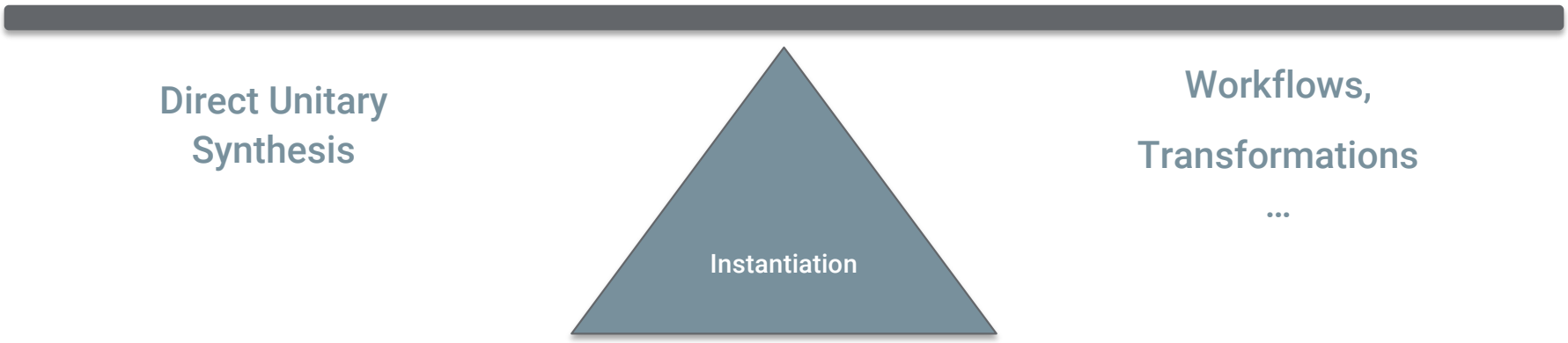
2K qubits already demonstrated ☺

Instantiation-Based Algorithms in BQSKit

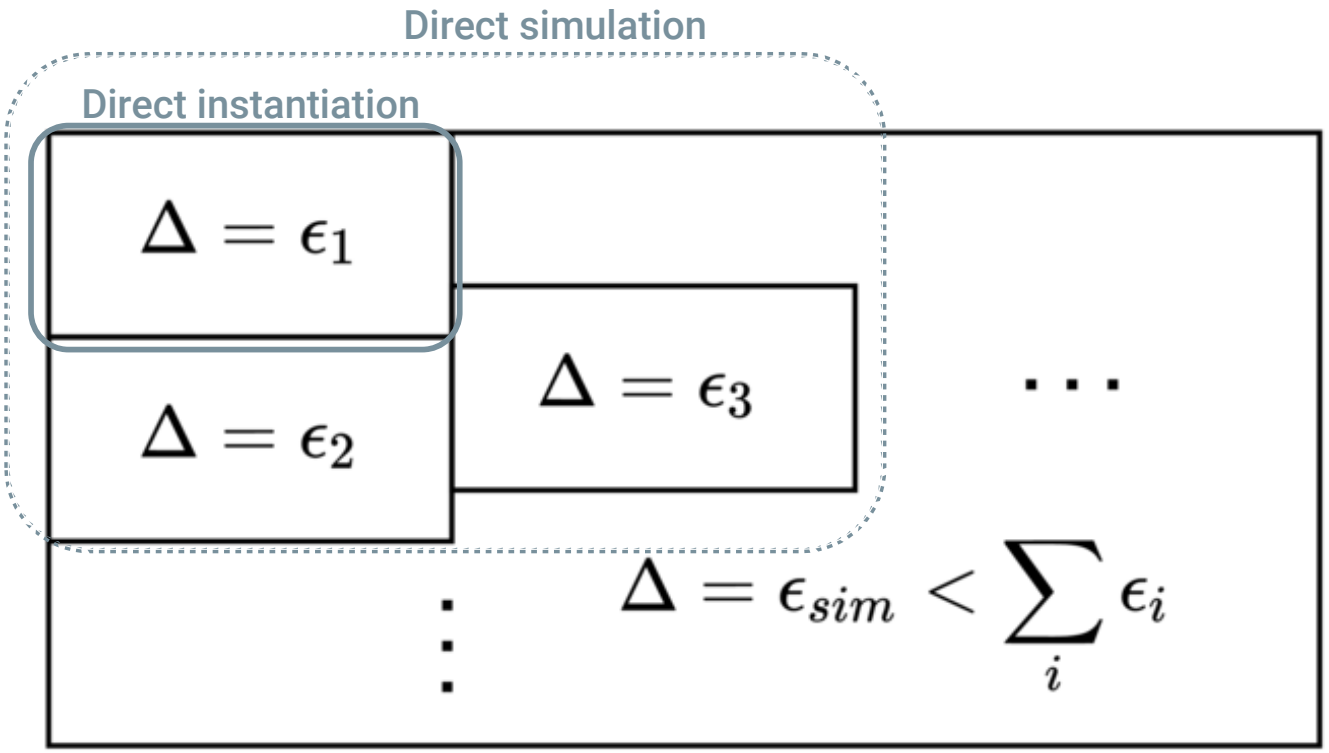
- QSearch – 3-4 qudits, optimal
- LEAP – 4-6 qudits
- QFAST – 6-7 qudits
- QPredict - 6-12+ qudits
- PAS – 3 qudits

1000s qubits

- Optimization (re-synthesis, gate deletion)
- Mapping/Routing (Generalized SABRE, PAM)
- Approximations (QuEST)
- Gate Set Retargeting



Support for Circuit Verification



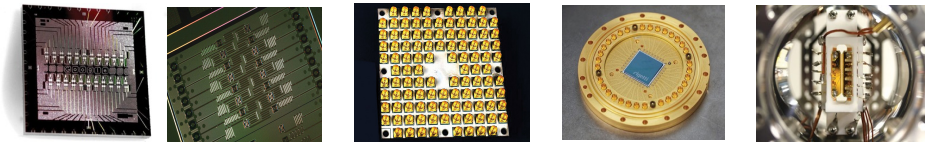
For our ”additive error” objective functions…

- Small circuits can have their error directly calculated
- In most of our publications we target 10^{-10} instantiation verification error, most often is less
- Larger circuits can be first partitioned into large simulatable sections, then summing the section errors gives a tighter upper bound on total error

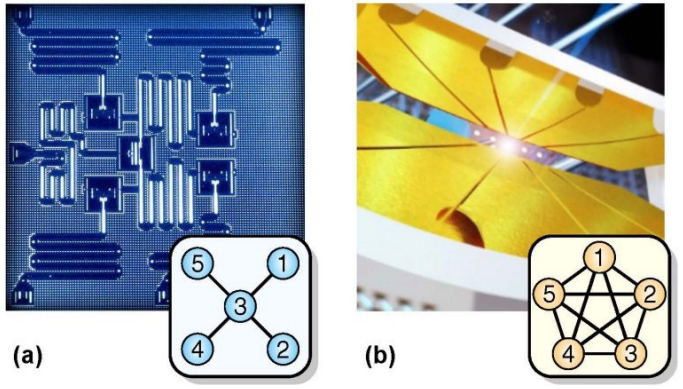
Circuit Optimization, Mapping and Transpilation

(aka Topology Aware Synthesis)

Hardware is Diverse: Portability



Google IBM Intel IonQ Rigetti



- Each chip has a native gate set
  - Single qubit: parameterized rotations
  - Two qubit: CNOT, Syc..
- Each chip has a constrained qubit interconnection topology
  - Ring, array, tree

Resources == Performance

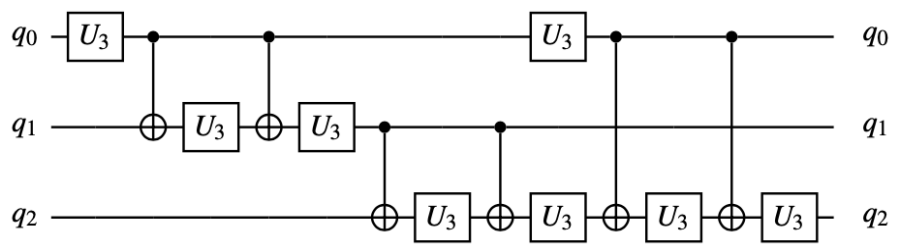
Number of gates and circuit depth are direct measures of performance.

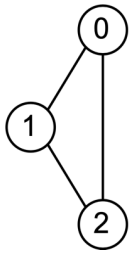
Resource Optimization
(gates, depth etc.)

Performance Enhancer + Capability Provider

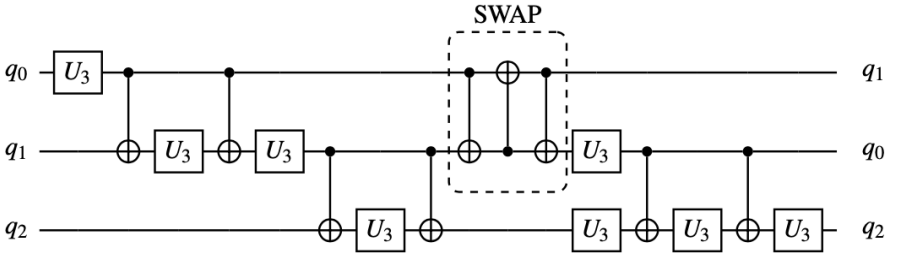
Generating 3-qubit QFT (for Linear Architecture)

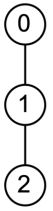
- Optimal 3-qubit QFT (analytic):
  6 CNOTs





- Optimal mapping (OLSQ[2]):
  9 CNOTs





Tan, B. et al, Optimal layout synthesis for quantum computing ICCAD. 2020.

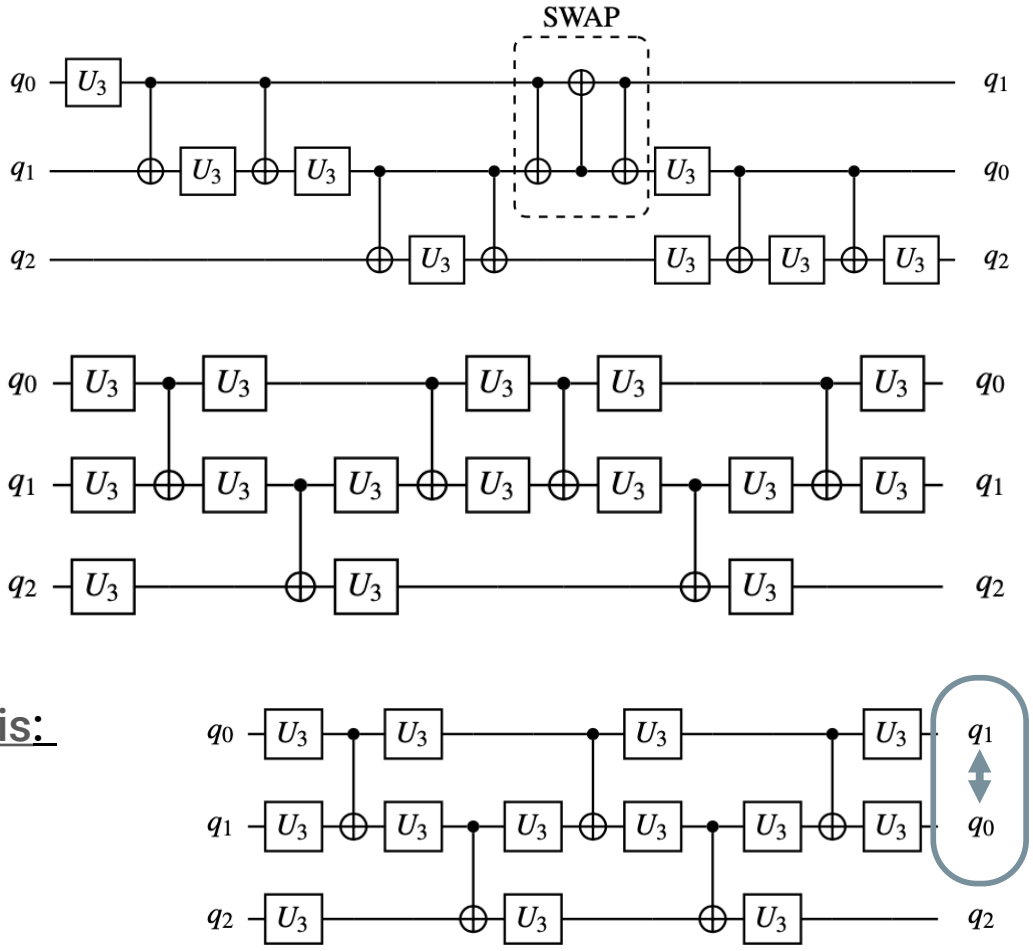
QFT3 Synthesis and Mapping

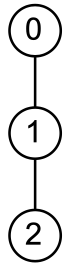
- Optimal mapping (OLSQ):
  9 CNOTs

- QSearch synthesis:
  6 CNOTs

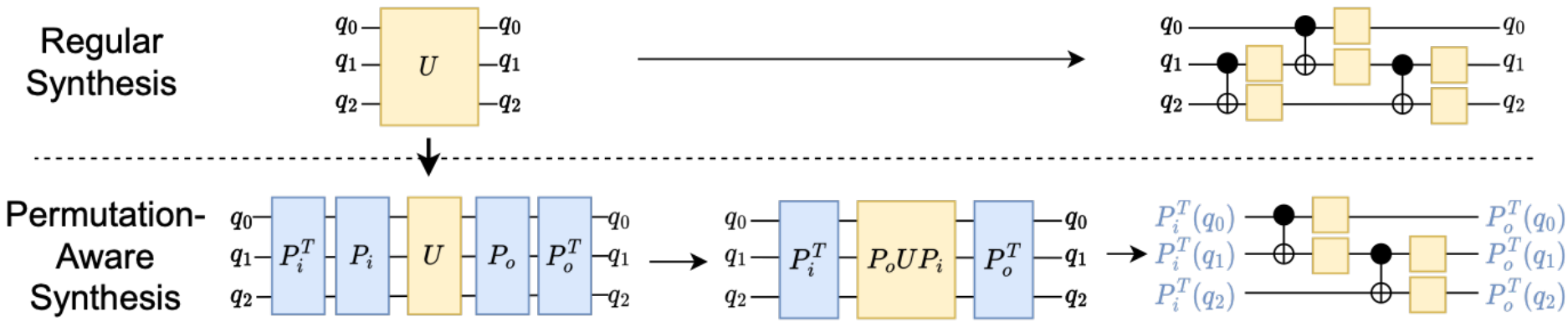
- Permutation Aware Synthesis:

  5 CNOTs



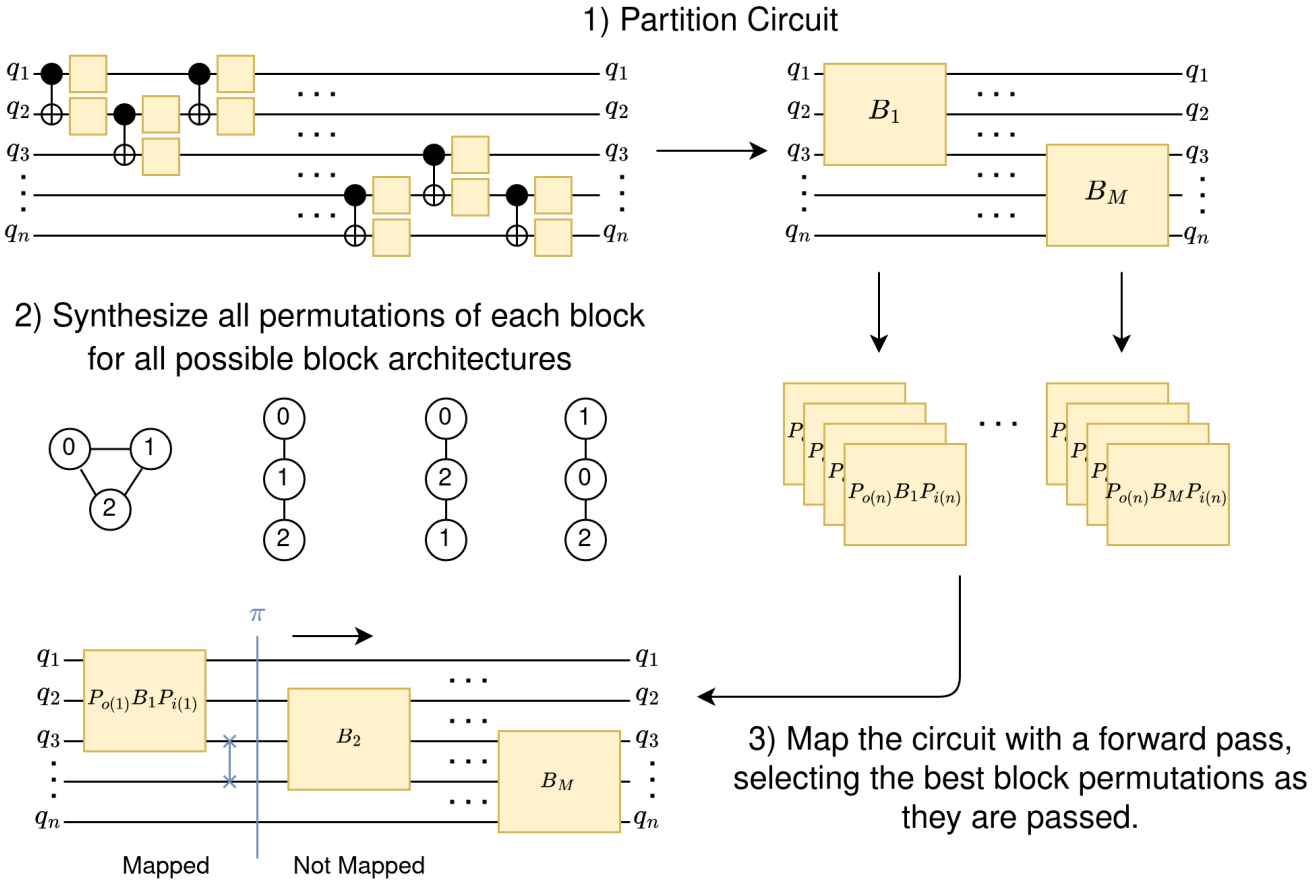


Permutation-aware synthesis (PAS)



1. Generate all input-output qubit permutations
2. Communication ”absorbed” at unitary level

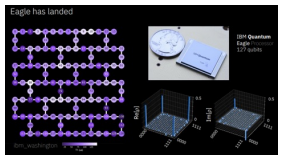
Scaling: Permutation-aware Mapping (PAM)

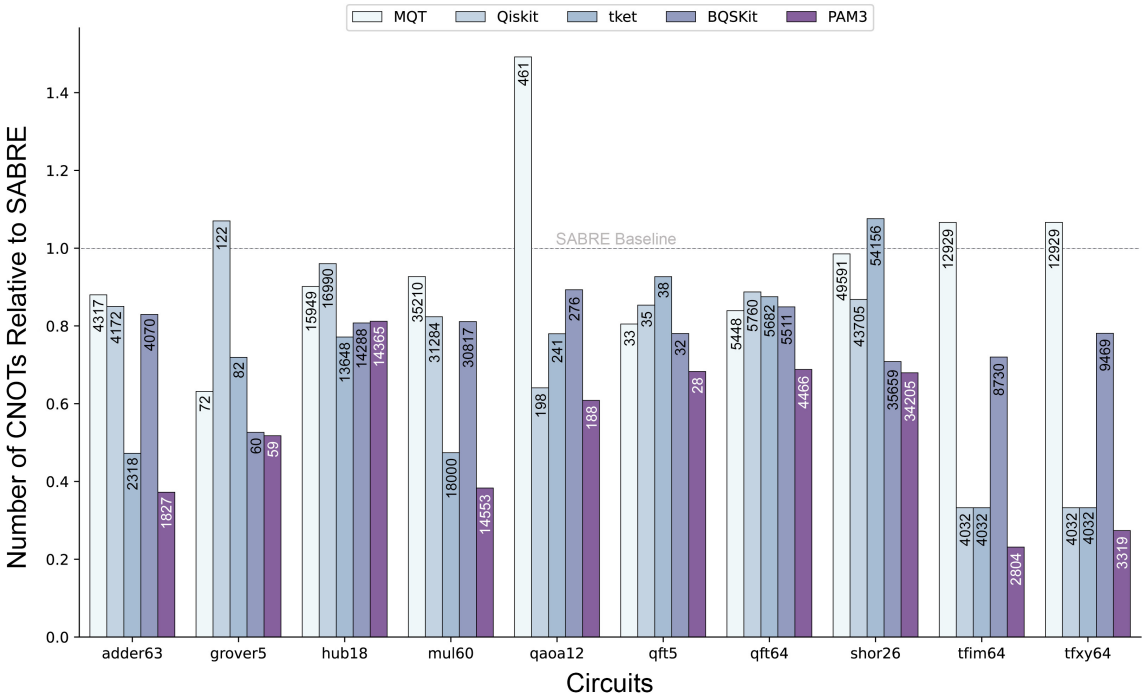


3) Map the circuit with a forward pass, selecting the best block permutations as they are passed.

Example: Mapping to IBM's 127-qubit Eagle Chip

IBM (copyright)

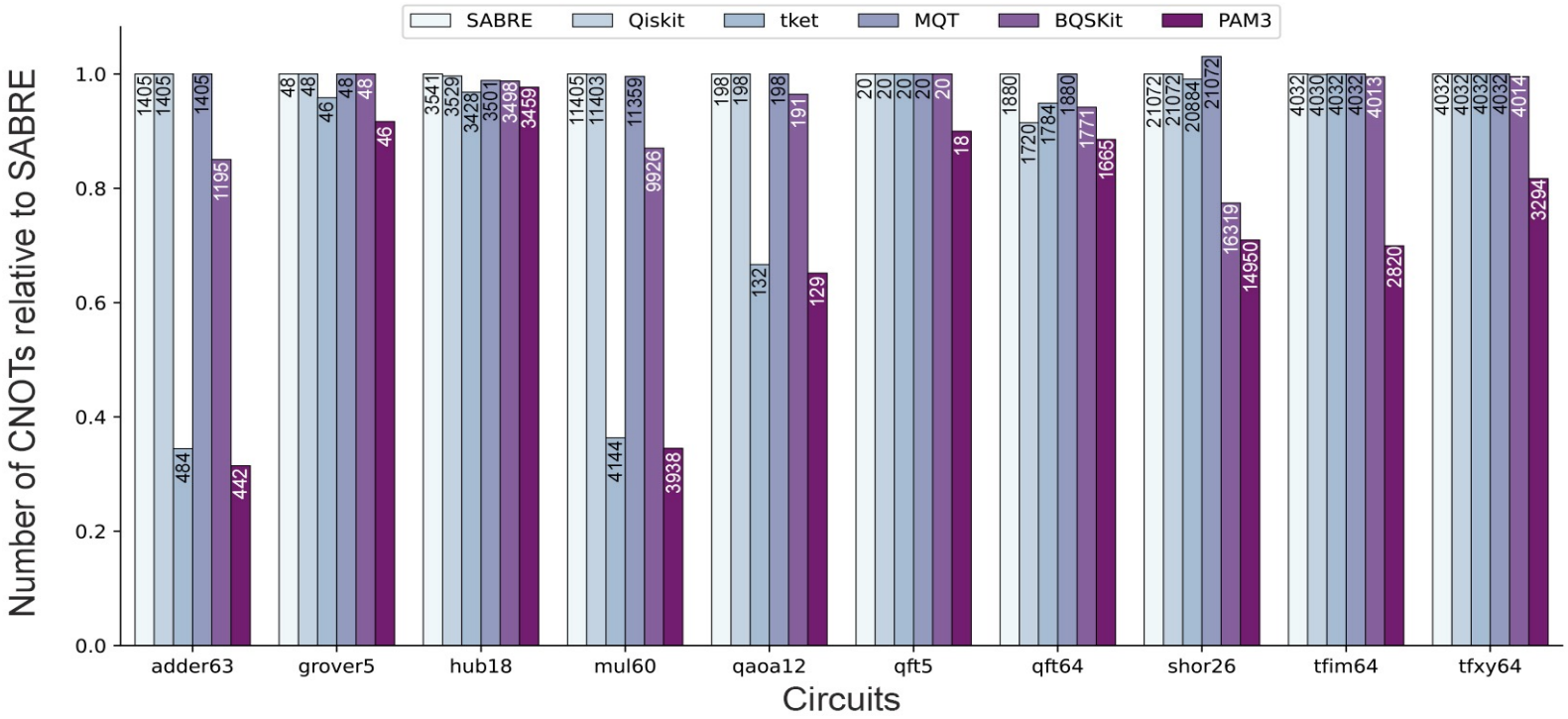




CNOT reduction (MAX/AVG)

- 78% / 33% MQT (QMAP)
- 68% / 18% Qiskit
- 36% / 9% Tket
- 67% / 21% BQSKit

Mapping on a Fully-Connected (all-to-all) Device



Most compilers may not change circuit when mapping is already legal.
(e.g. from restricted → all-to-all).

QUANTUM CIRCUIT OPTIMIZATION AND TRANSPILATION VIA PARAMETERIZED CIRCUIT INSTANTIATION

Scientific Achievement

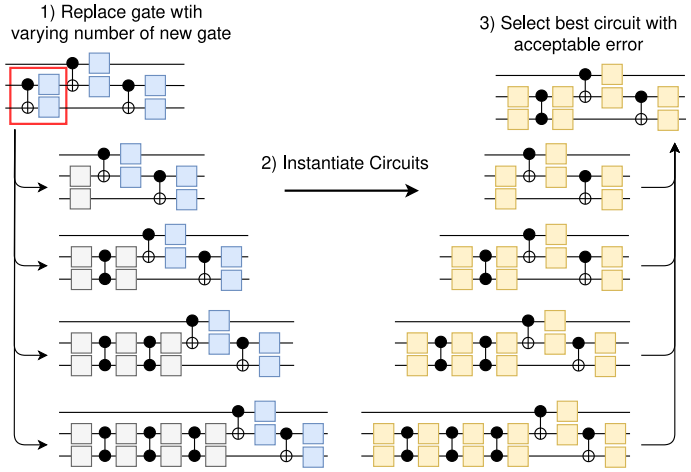
Leverage parameterized circuit instantiation via numerical optimization to achieve improved circuit optimization and gate-set transpilation, leading to significant reductions in gate counts, enhanced circuit quality, and greater portability between quantum processors. Solution scales to 1000s of qubits.

Significance and Impact

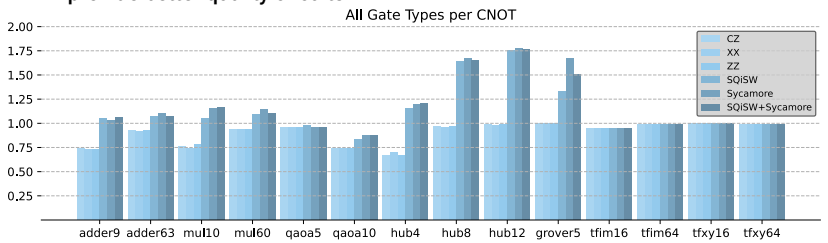
Gate deletion and rule-free gate transpilation algorithms enable automatic transformations of quantum circuits between any gate-sets, eliminating the need for manual user-defined rules, greatly improving quantum program portability and reducing barriers to entry for gate-set exploration studies. All algorithms are released as part of the BQSKit code. BQSKit significantly outperforms commercial compilers (Qiskit, Cirq, Tket).

Technical Approach

- Numerical optimization of circuits is leveraged to transform circuits.
- We remove or change gates and reinstantiate the remaining to accommodate for the lost or transformed gate.
- Circuit partitioning techniques are used to scale to 1000s of qubits/qudits.



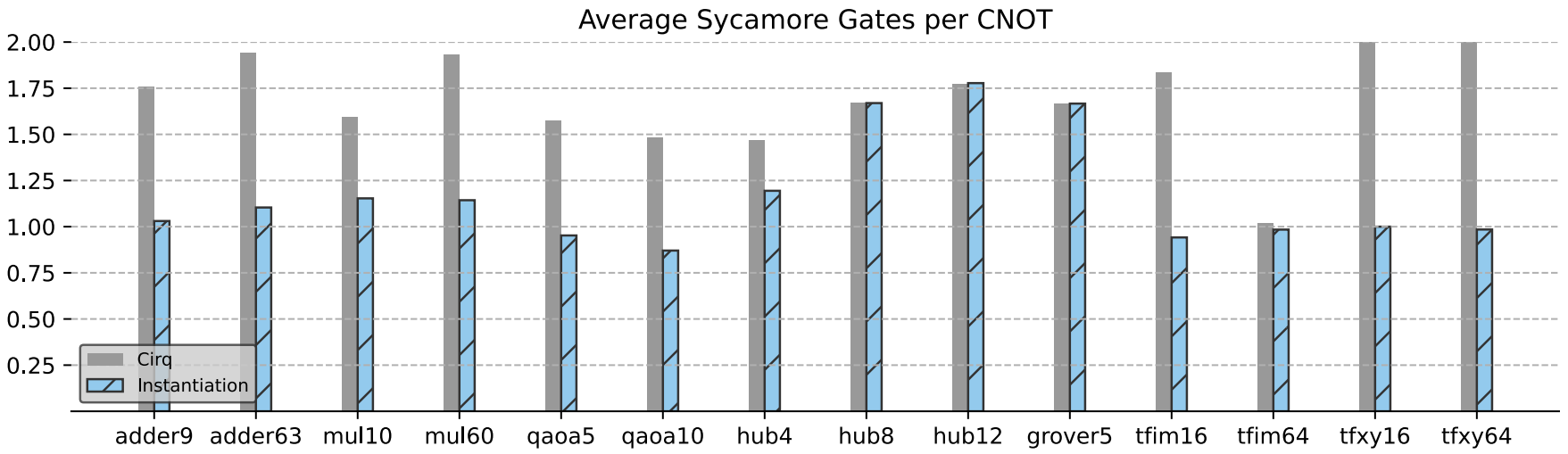
In gate set transpilation, one native 2-qubit gate is replaced with a series of templates implemented in the new target gate set. Instantiation is used to select the shortest suitable template. Partial solutions are expressed in a combination of gate sets and globally optimized. When comparing to traditional compilers BQSKit eliminates the need for theoretical gate-per-gate decomposition rules and provide better quality circuits.



PI: Costin Iancu; Berkeley Lab POC: Ed Younis
ASCR Program: AQRC (AIDE-QC)
ASCR PM: Ceren Susut-Bennett
Publication(s) for this work: Ed Younis, Costin Iancu. "Quantum Circuit Optimization and Transpilation via Parameterized Circuit Instantiation," IEEE International Conference on Quantum Computing and Engineering (QCE) (2022): IEEE. doi:10.1109/QCE53715.2022.00068.
Code Developed or Datasets: https://github.com/bqskit/bqskit

Transpilation and Gate Set Portability

Specialized algorithms for gate set transpilation, more powerful than 1-1 translation/rewriting rule in most compilers



E. Younis et al., "Quantum Circuit Optimization and Transpilation via Parameterized Circuit Instantiation," in 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), IEEE, 2022

The power of circuit synthesis and instantiation comes from the ability to do **configurable** global optimization on programs

1. M.Davis et al., "***Towards Optimal Topology Aware Quantum Circuit Synthesis***," in 2020 IEEE QCE. (Best Paper Award)

2. E. Younis et al***, "QFast: Conflating Search and Numerical Optimization for Scalable Quantum Circuit Synthesis,"*** in 2021 IEEE QCE. (Best Paper Award)

3. E. Younis et al., "***Quantum Circuit Optimization and Transpilation via Parameterized Circuit Instantiation***," in 2022 IEEE QCE.

4. X.C. Wu et al, "***Reoptimization of Quantum Circuits via Hierarchical Synthesis***," in 2021 IEEE International Conference on Rebooting Computing (ICRC), IEEE, 2021

5. *T. Patel et al, "**QUEST: Systematically Approximating Quantum Circuits for Higher Output Fidelity**", ASPLOS 2022*

6. J. Liu et al, "*Tackling the Qubit Mapping Problem with Permutation-Aware Synthesis*", IEEE QCE 2023

Handling Higher Dimension Objects

(multi-qubit gates, qutrits etc.)

Alon Kukliansky et al. "***QFactor: A Domain-Specific Optimizer for Quantum Circuit Instantiation***". IEEE QCE 2023

QFactor: A Domain-Specific Optimizer for Quantum Circuit Instantiation

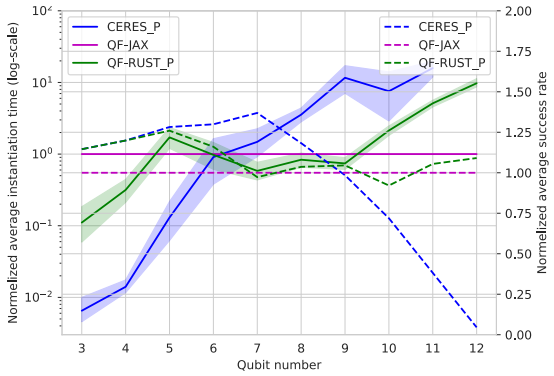
Scientific Achievement Novel domain-specific algorithm for numerical optimization used by quantum circuit instantiation, synthesis, and compilation methods. QFactor uses a tensor network formulation together with analytic methods and an iterative local optimization algorithm to reduce the number of problem parameters.

Significance and Impact The formulation is amenable to portable parallelization across CPU and GPU architectures, challenging in general purpose optimizers (GPO). QFactor achieves exponential memory and performance savings with optimization success rates similar to GPOs. QFactor can process directly circuits with more than 12 qubits. We enable BQSKit optimizations of 100+ qubit circuits to scale out linearly with the hardware resources allocated for compilation in GPU environments.

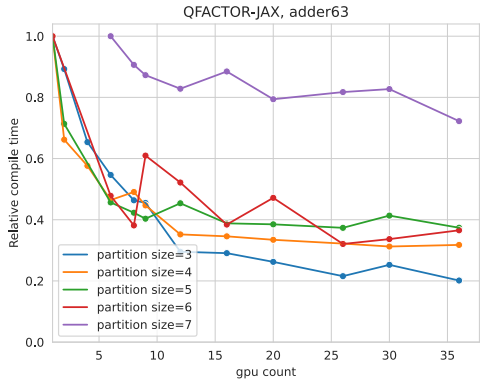
Technical Approach

- Tensor network formulation enables the algorithm to work with whole unitaries. Drastically reduced total number of optimized parameters compared to GPOs.
- CPU-based implementation written in Rust, together with a Python implementation written using JAX , portable across CPUs and GPUs.

PI: Costin Iancu; Berkeley Lab POC: Costin Iancu
ASCR Program: AQRC (AIDE-QC)
ASCR PM: Ceren Susut-Bennett
Publication(s) for this work: Alon Kukliansky et al. "QFactor: A Domain-Specific Optimizer for Quantum Circuit Instantiation". IEEE QCE 2023
Code Developed or Datasets: https://github.com/bqskit/bqskit



Average instantiation time normalized to QF-JAX instantiation time (left-hand side y-axis), together with normalized success rate (right-hand side y-axis), showing the strength of QF actor for larger circuits.

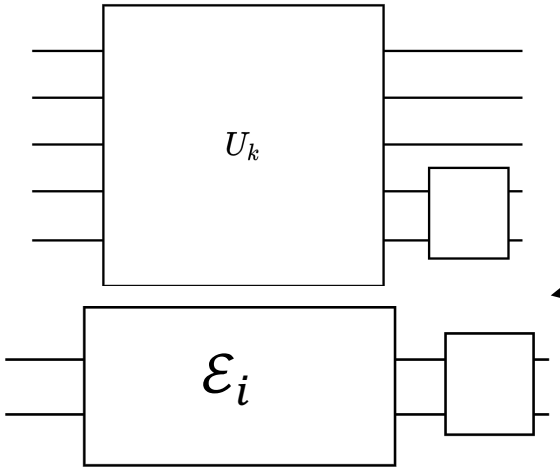


QFactor benefits from GPU acceleration and enables good strong scaling for compilation workflows.

QFactor Enables Scalability

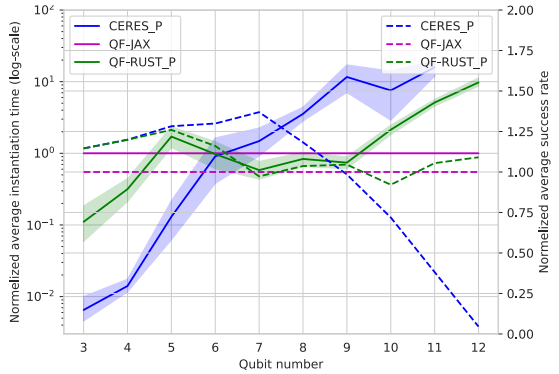
Tensor network formulation allows

- Handling of arbitrary sized unitaries as single parameter
- GPU acceleration and parallelization



Environment matrix

$$Tr\left(U_t^\dagger \cdot u_1 \cdots u_p\right) = Tr(\mathcal{E}_i \cdot u_k)$$



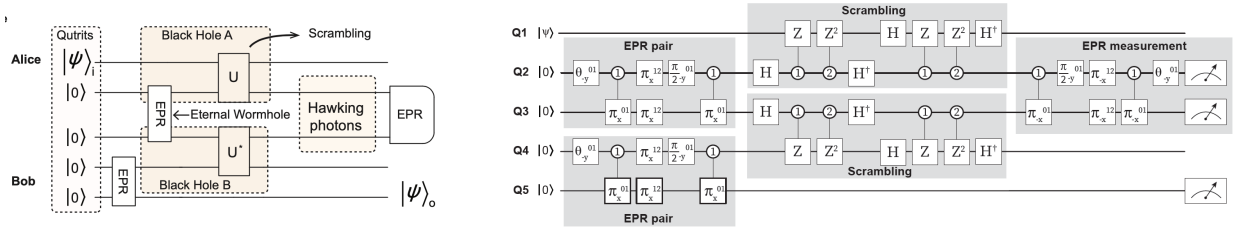
In this context, what is the best gate or set of parameters for the gate?

$$u_{k_{opt}} = YX^\dagger$$

$$\mathcal{E}_i = XDY^\dagger$$

BQSKit Supports Qutrits, Qudits etc.

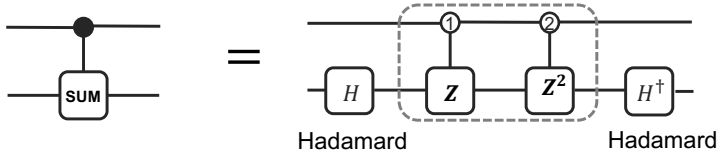
Black Hole Information Scrambling



Qutrit synthesis using CSUM and parameterized single qutrit gates

$$\begin{bmatrix} cos(\theta_1)cos(\theta_2)e^{i\phi_1} & sin(\theta_1)e^{i\phi_3} & cos(\theta_1)sin(\theta_2)e^{i\phi_4} \\ sin(\theta_2)sin(\theta_3)e^{-i\phi_4 - i\phi_5} - sin(\theta_1)cos(\theta_2)cos(\theta_3)e^{i\phi_1 + i\phi_2 - i\phi_3} & cos(\theta_1)cos(\theta_3)e^{i\phi_2} & -cos(\theta_2)sin(\theta_3)e^{-i\phi_1 - i\phi_5} - sin(\theta_1)sin(\theta_2)cos(\theta_3)e^{i\phi_2 - i\phi_3 + i\phi_4} \\ -sin(\theta_1)cos(/theta_2)sin(\theta_3)e^{i\phi_1 - i\phi_3 + i\phi_5} - sin(\theta_2)cos(\theta_3)e^{-i\phi_2 - i\phi_4} & cos(\theta_1)sin(\theta_3)e^{i\phi_5} & cos(\theta_2)cos(\theta_3)e^{-i\phi_1 - i\phi_2} - sin(\theta_1)sin(\theta_2)sin(\theta_3)e^{-i\phi_3 + i\phi_4 + i\phi_5} \end{bmatrix}$$

$\text{CSUM}|i,j\rangle = |i, i+j \bmod 3\rangle$ $\quad$ $\text{Cphase}|i,j\rangle = \omega^{ij}|i,j\rangle$



BQSKit

M.S. Blok et al. "Quantum Information Scrambling on a Superconducting Qutrit Processor". Phys. Rev. X 11, 021010, 2021

Portability and Hardware Design Studies (Quantum Roofline?)

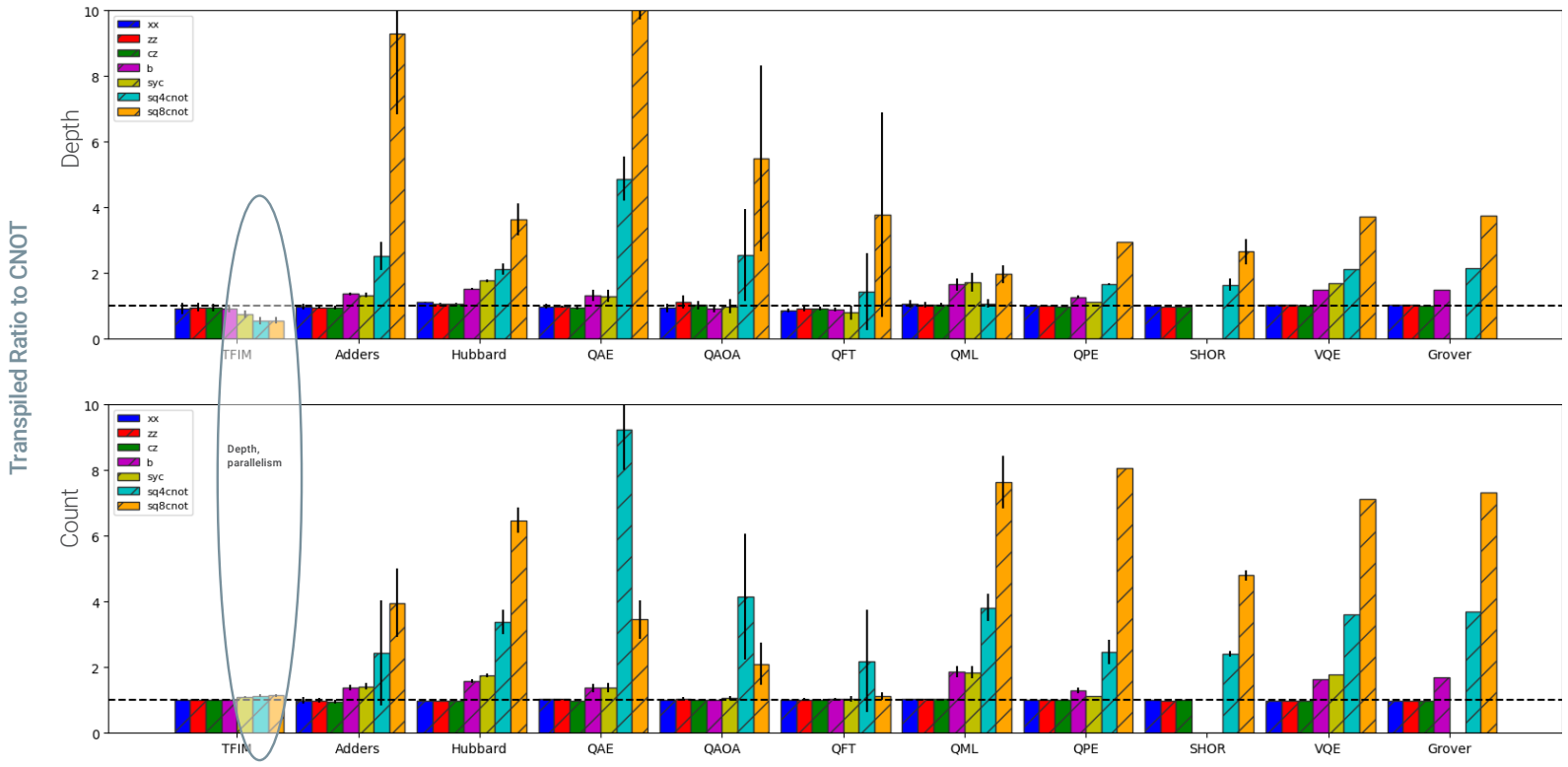
“Where should I run my algorithm?”

“How can I improve my machine?”

1. Gates have different expressive power
2. Gates have different fidelity, latency
3. Chips have different topology

Given optimal circuits, I can ask:

What is the better architecture (gates, topology) for a class of algorithms?



Fidelity models:

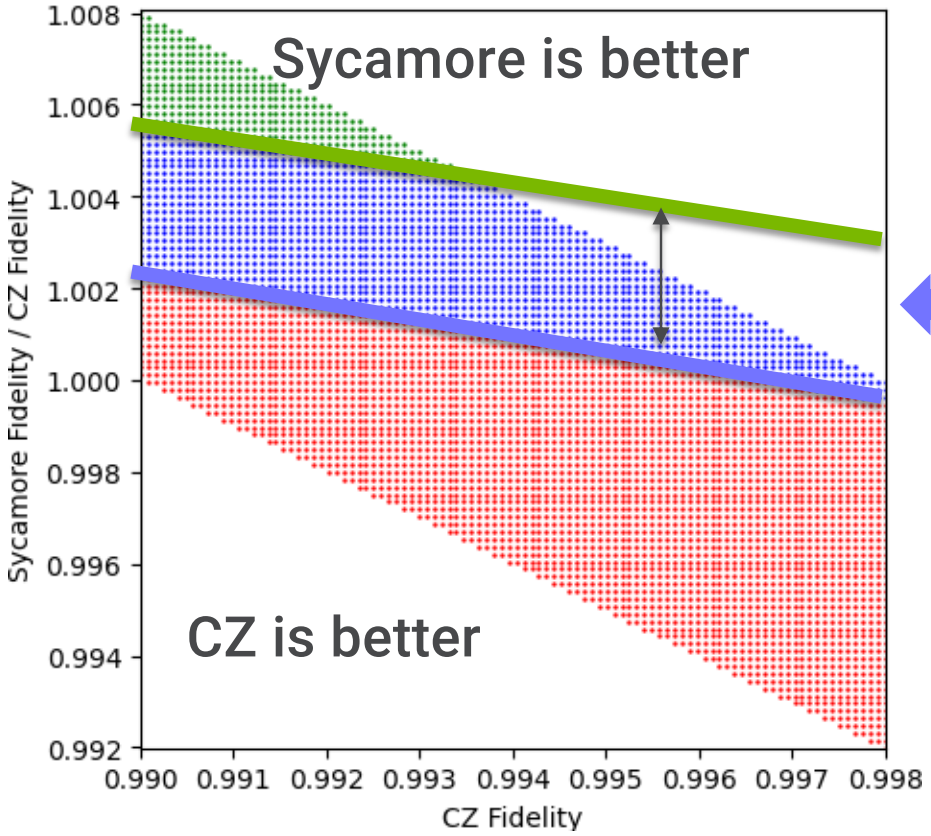
1. Gate count

$$F = \prod_{i=1,2} f_i^{n_i}$$

1. Depth...
2. Depth and parallelism...

BQSKit

Which 2-qubit gate can provide higher fidelity?



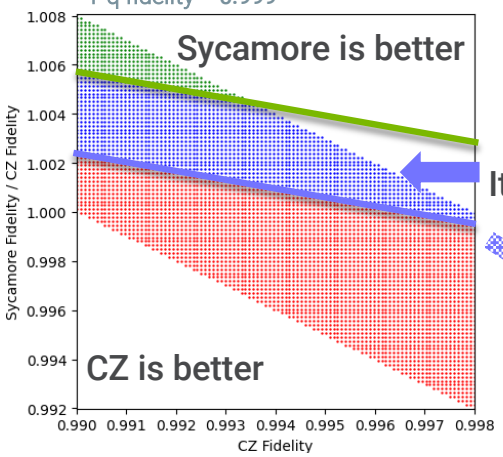
It depends on algorithm, circuit

1-q fidelity =0.999

BQSKit

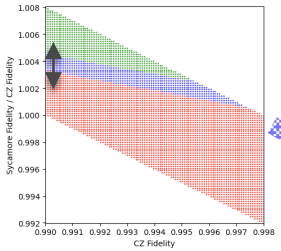
Which 2-qubit gate can provide higher fidelity?

1-q fidelity = 0.999



It depends on algorithm, circuit

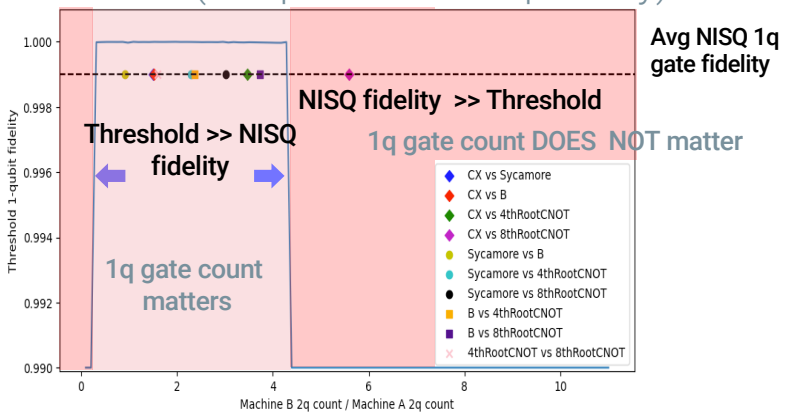
1-q fidelity = **0.9995 > 0.999**



Algorithm dependent range shrinks

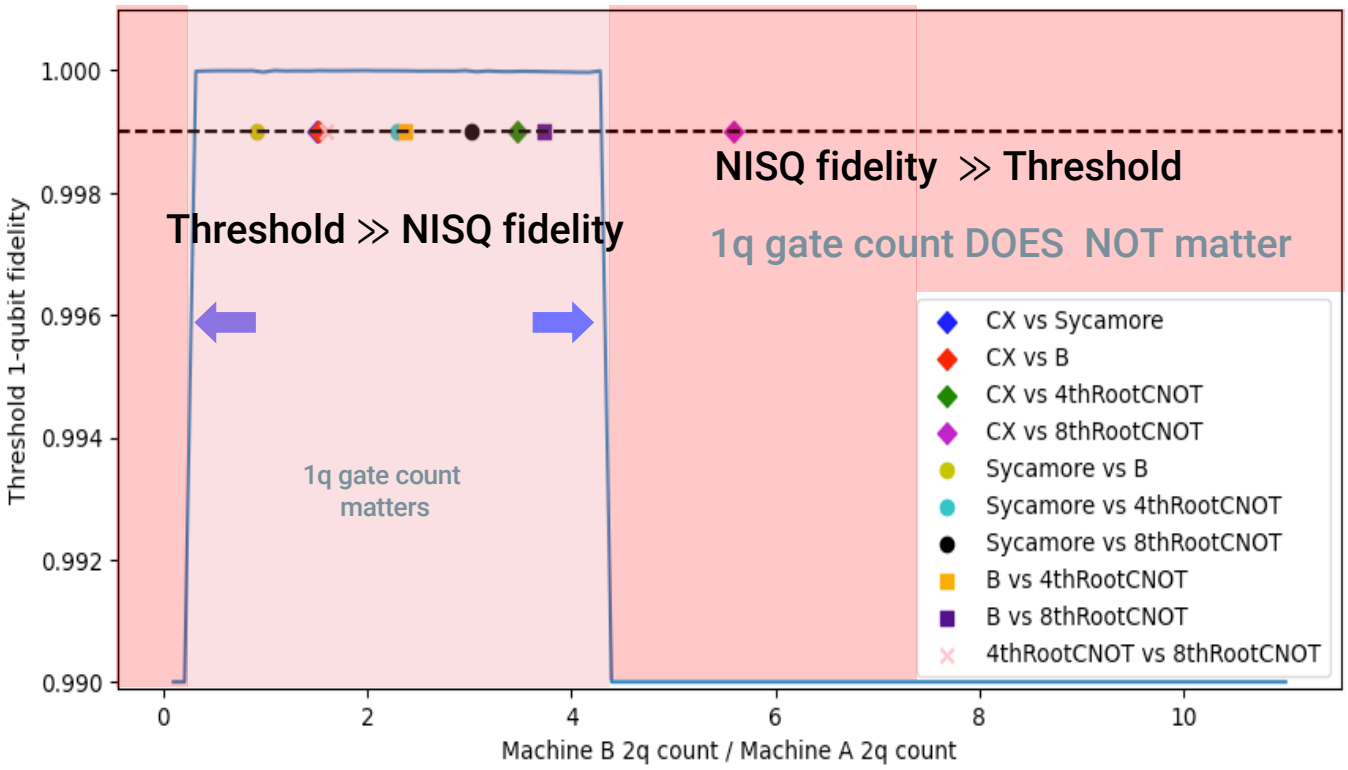
When do 1-q gates matter?

(independent of 2-q fidelity)



When do 1-q gates matter?

(independent of 2-q fidelity)



Avg NISQ 1q gate fidelity

Algorithm Exploration for NISQ and Beyond

1. Circuit Approximations (Error mitigation and Performance)

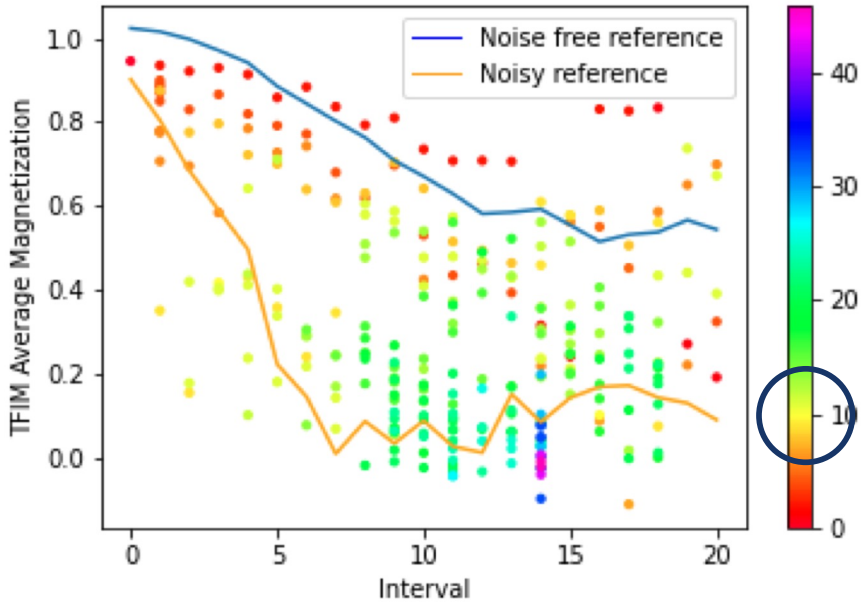
- Wilson et al. "***Empirical Evaluation of Circuit Approximations on Noisy Quantum Devices***.", in SC 2021
- *T. Patel et al, "**QUEST: Systematically Approximating Quantum Circuits for Higher Output Fidelity***", ASPLOS 2022

2. Discover Algorithm Generators

- M. Weiden et al, "***Improving Quantum Circuit Synthesis with Machine Learning***", in 2023 IEEE QCE.
- L. Bassman et al, "***Constant-Depth Circuits for Free-Fermion Dynamic Simulations on Quantum Computers***," Mat. Theory 6, 13 (2022)

Why Approximate Quantum Circuits?

- Approximations are built using process metrics: $|U-U'| < \varepsilon$
- The performance (resources) of a quantum program is determined by the number of gates
- Approximations can minimize the number of gates (circuit depth)
  - Circuits up to 2X shorter can produce the same quality output and have better fidelity on NISQ devices
  - Circuits will run faster in FT devices



Wilson et al. "Empirical Evaluation of Circuit Approximations on Noisy Quantum Devices." International Conference for High Performance Computing, Networking, Storage, and Analysis (2021).

QuEST: Robust Generation of Quantum Circuit Approximation Using Synthesis

Scientific Achievement

We provide a sound and scalable method for generating circuit approximations. Approximations significantly reduce circuit depth, while providing error mitigation. **QuEst provides an orthogonal technique to randomized compiling.**

Research Details

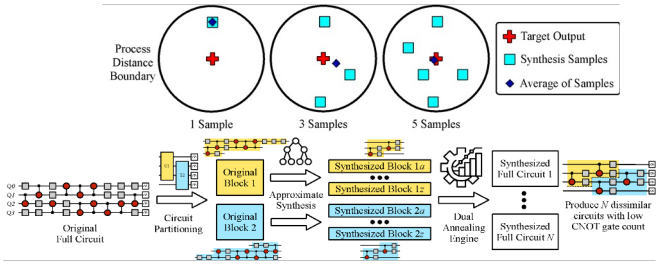
- We use synthesis and circuit partitioning (BQSKit) for circuit generation
- **We bound error - theoretical upper bound on HS distance under partitioning and approximation.**
- Apriori approximation selection criteria to ensures high fidelity output (triangle inequality on Hilbert-Schmidt distance, annealing)

Significance and Impact

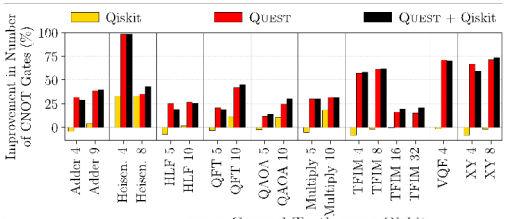
Resource efficiency is an important measure of circuit performance. Our technique can be directly used for circuit optimization in NISQ and fault-tolerant quantum computing. In NISQ, we provide additional capability for very good error mitigation.

- We show 30%-80% depth reduction on many important algorithms
- We show fidelity improvements of 30% on noisy systems, independent of noise level
- The program output is accurate for science purposes
- Although computationally intensive, the technique scales up to high qubit counts (up 128 qubits demonstrated)

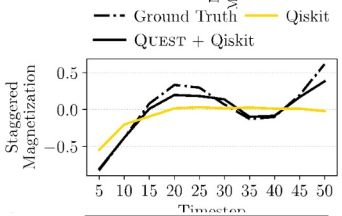
Sampling improves fidelity



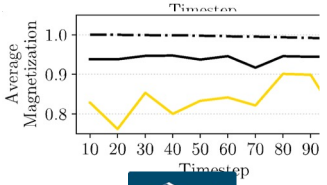
Circuit depth reduction



Heisenberg 4



TFIM 4

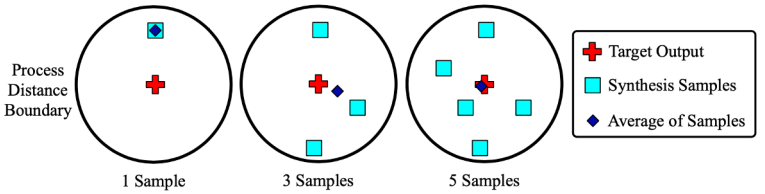


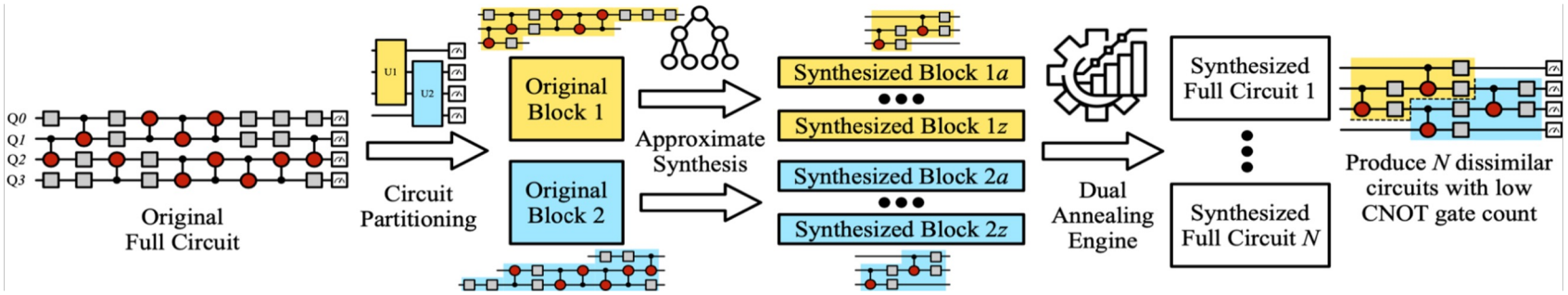
T. Patel et al, "QUEST: Systematically Approximating Quantum Circuits for Higher Output Fidelity", ASPLOS 2022

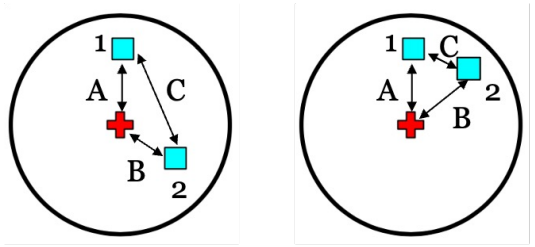
Funded by DOE ASCR AIDE-QC

BERKELEY LAB

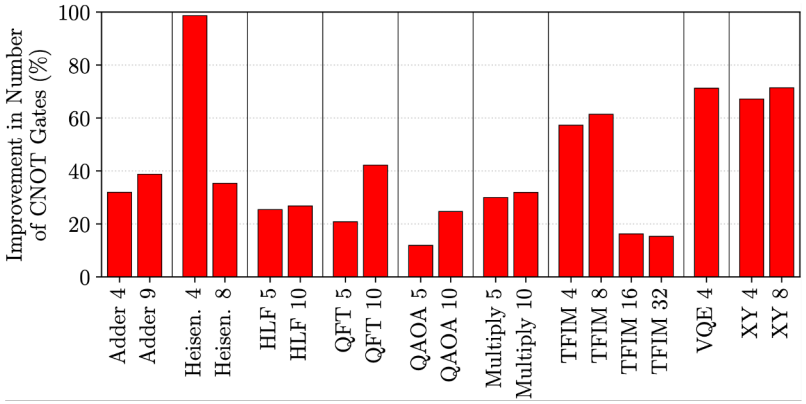
Select “dissimilar” approximations that sample the approximation space “uniformly”.



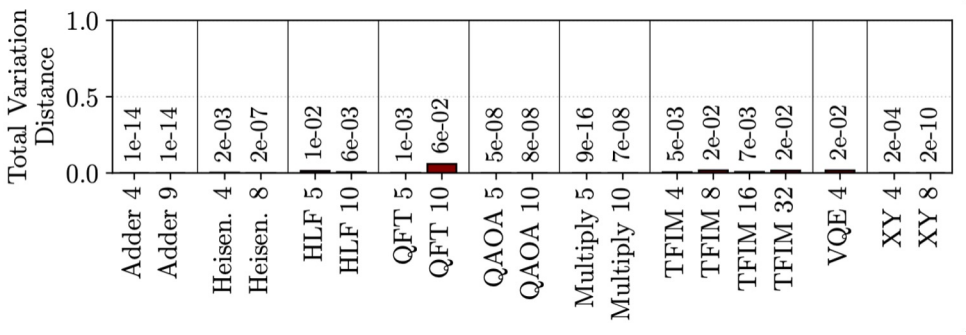




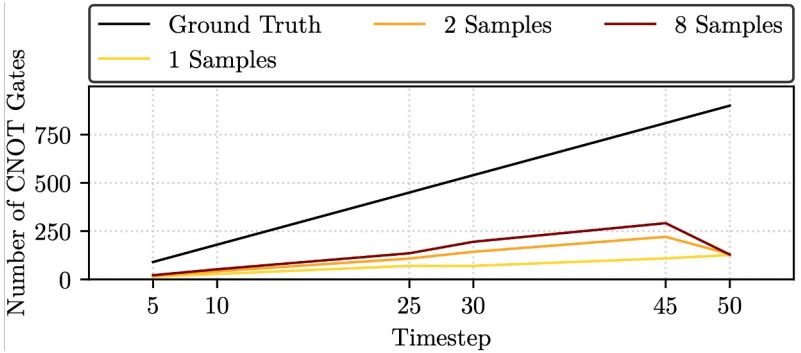
If C > *max*(A, B), *dissimilar*
Else *similar*



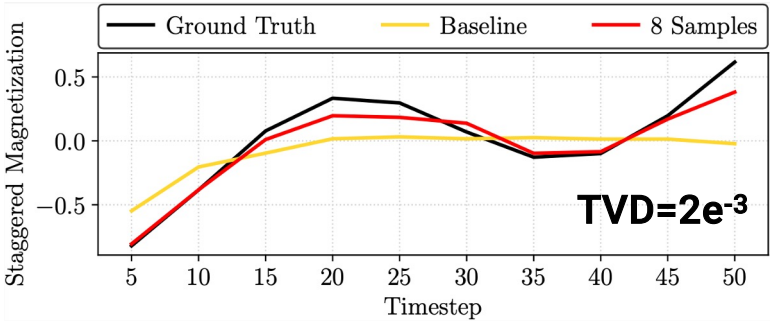
Depth reduction in the range of 30-90%



The output distance to the ground truth is good (< 10^{-2})



Depth reduction up to 85% for Heisenberg16



Approximations track the ground truth of the 4-spin Heisenberg on a quantum computer.

Existing approach is empirical, we finally have a soundness and robustness proof.

Hope to answer:

“ *How can I relate my circuit’s output to domain science expectations and constraints?”*

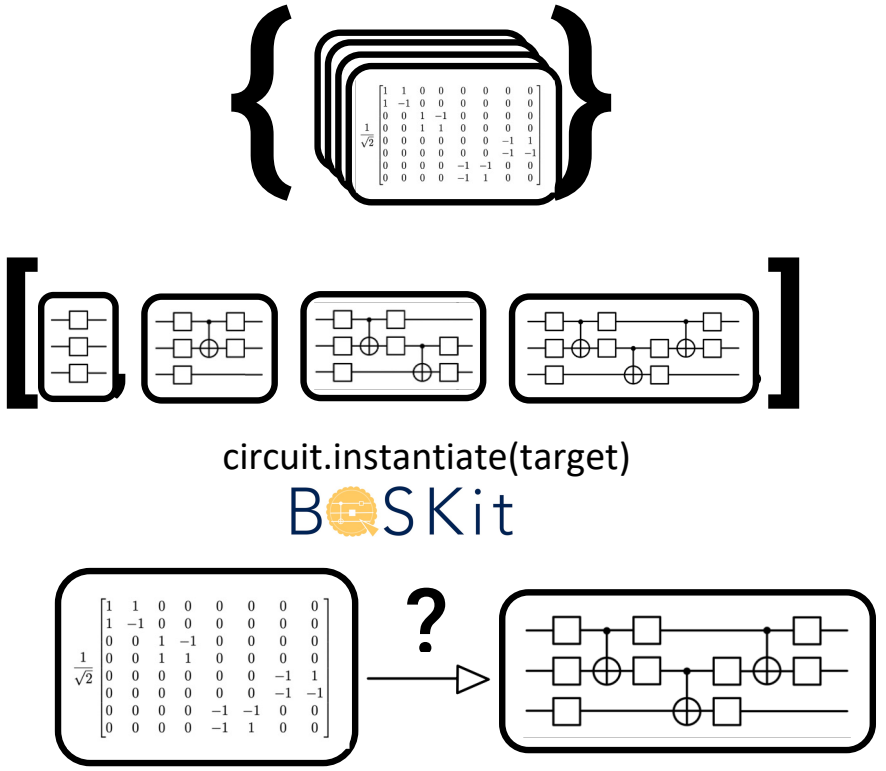
Learning Circuit Generating Templates

“Can I discover structure in circuits without domain science knowledge?”

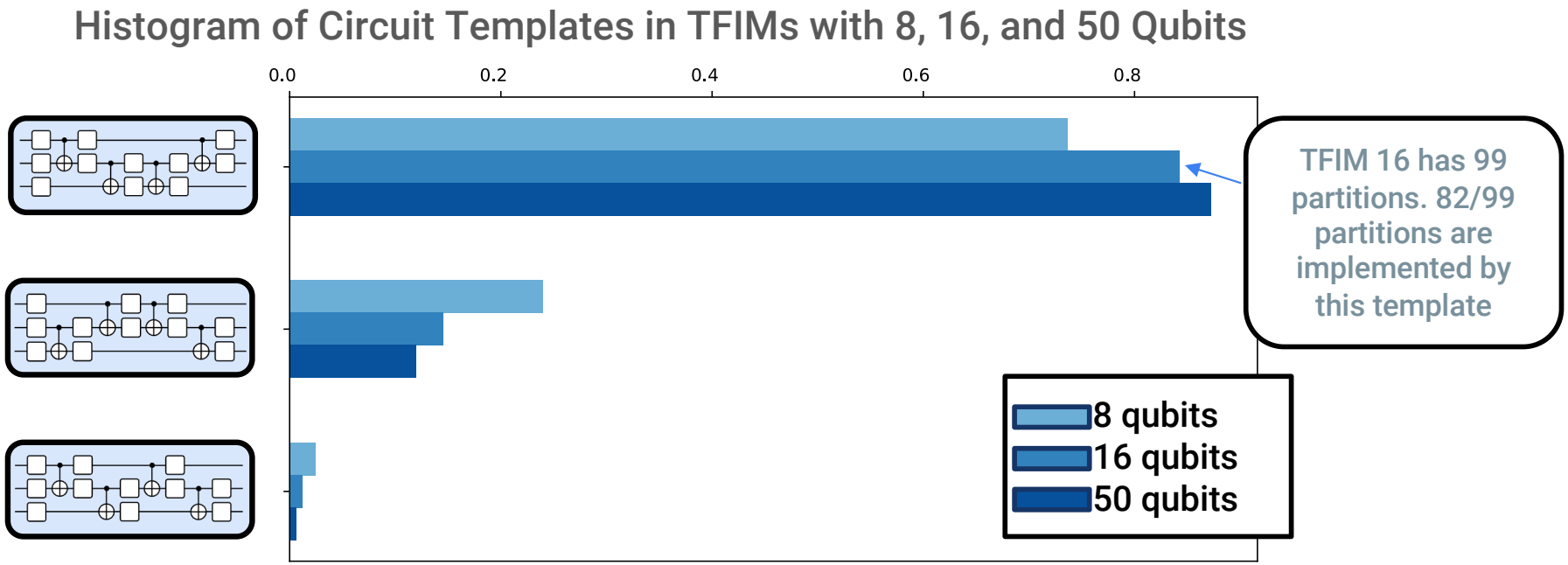
Detecting Minimal Generating Templates

Can I predict a good ansatz for any given unitary?

1. Collect a large dataset of unitaries
   - Consider unitaries from partitioned circuits
     - Suite of algorithms
     - Variety of circuit widths
2. Enumerate a finite number of circuit templates
3. Try instantiating each template given each unitary



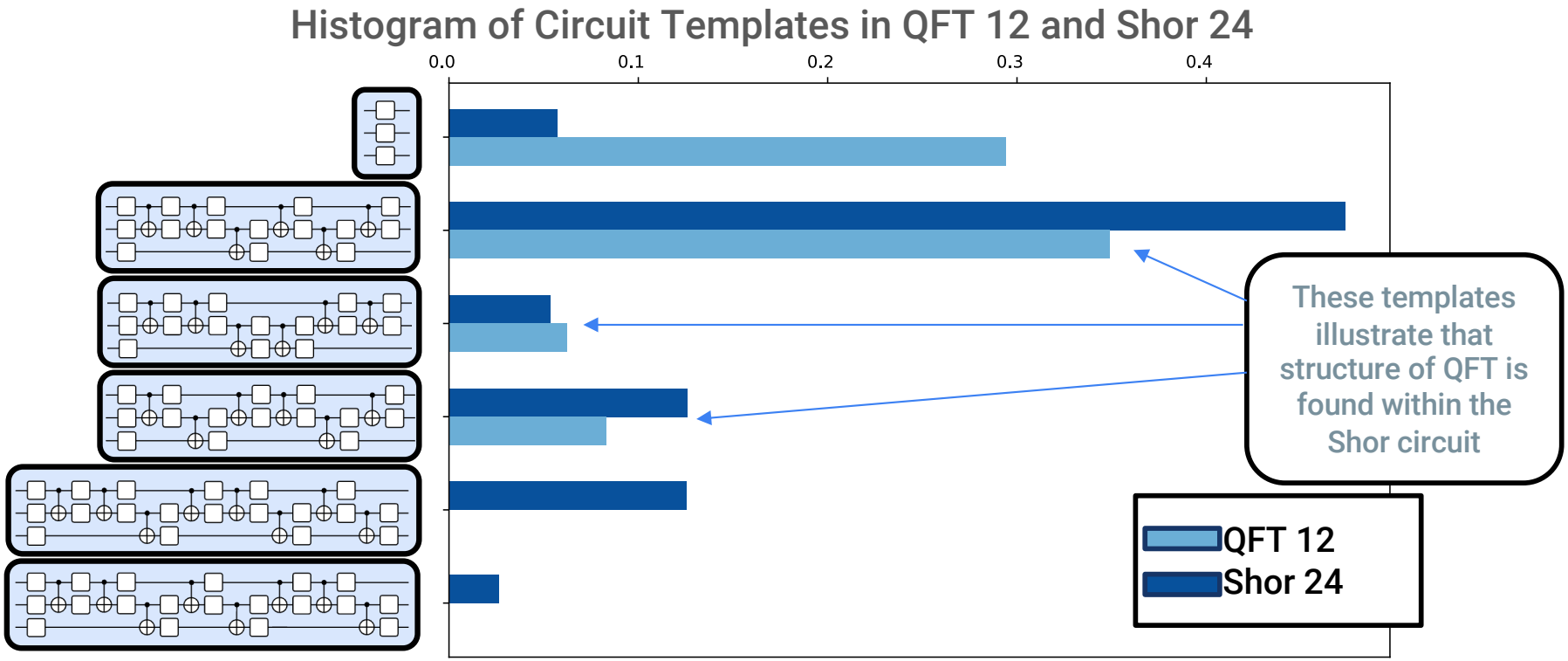
Patterns Occur when Algorithms Scale



Three templates account for 99% of partitioned TFIM circuit blocks of various widths

M. Weiden et al, "Improving Quantum Circuit Synthesis with Machine Learning", in 2023 IEEE International Conference on Quantum Computing and Engineering (QCE), IEEE, 2023

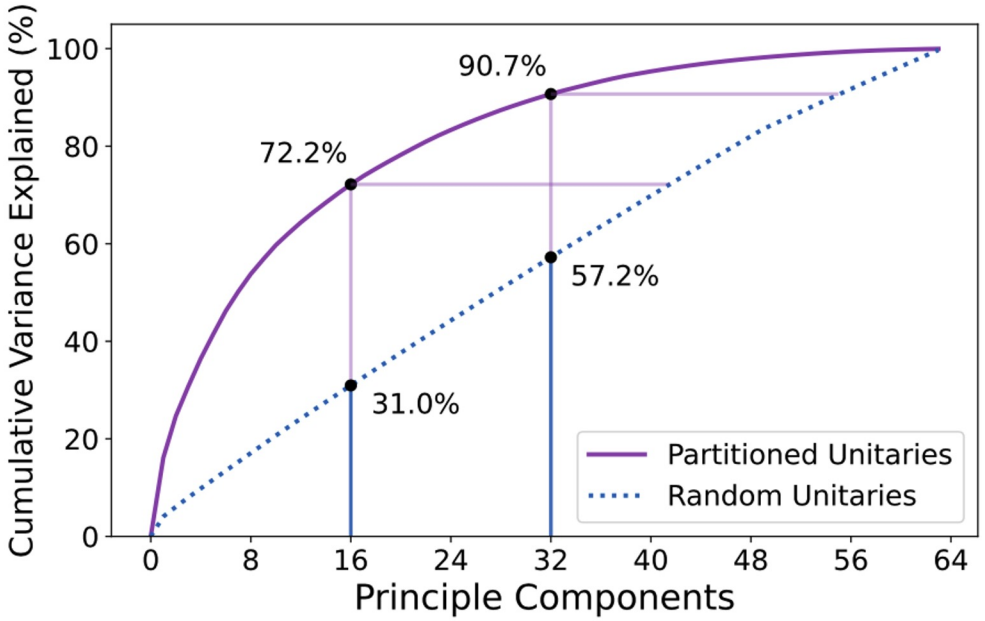
Patterns Occur When Algorithms are Composed

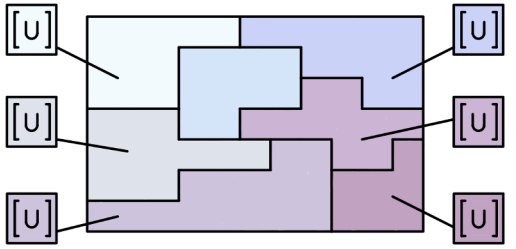


QFT 12 is contained in Shor 24 - QFT and IQFT appear 16 times

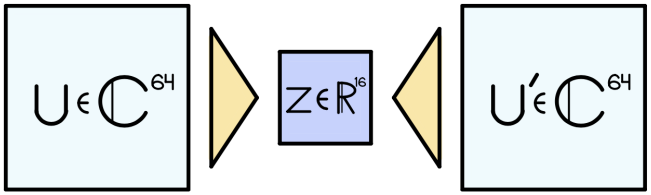
Algorithms are Different From Random Programs

- Principal Component Analysis reveals low dimensionality of unitaries of interest
  128 -> 16, 32
- Implies learning patterns in unitaries is possible





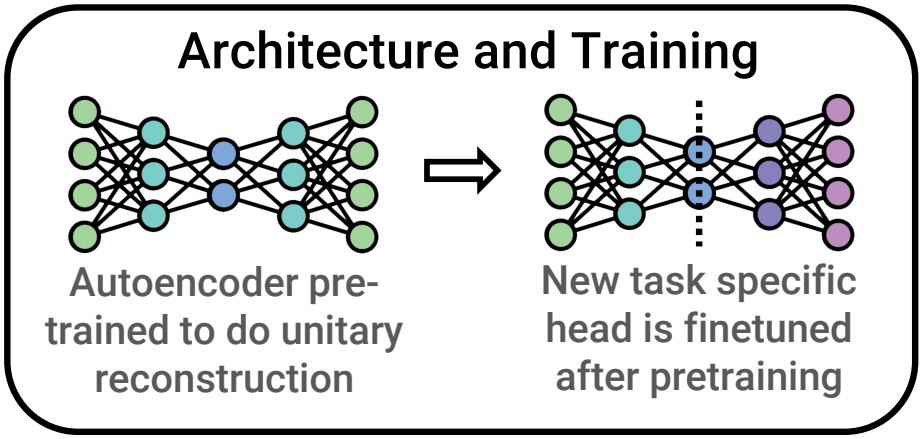
Partitioned unitaries are taken from circuit panels

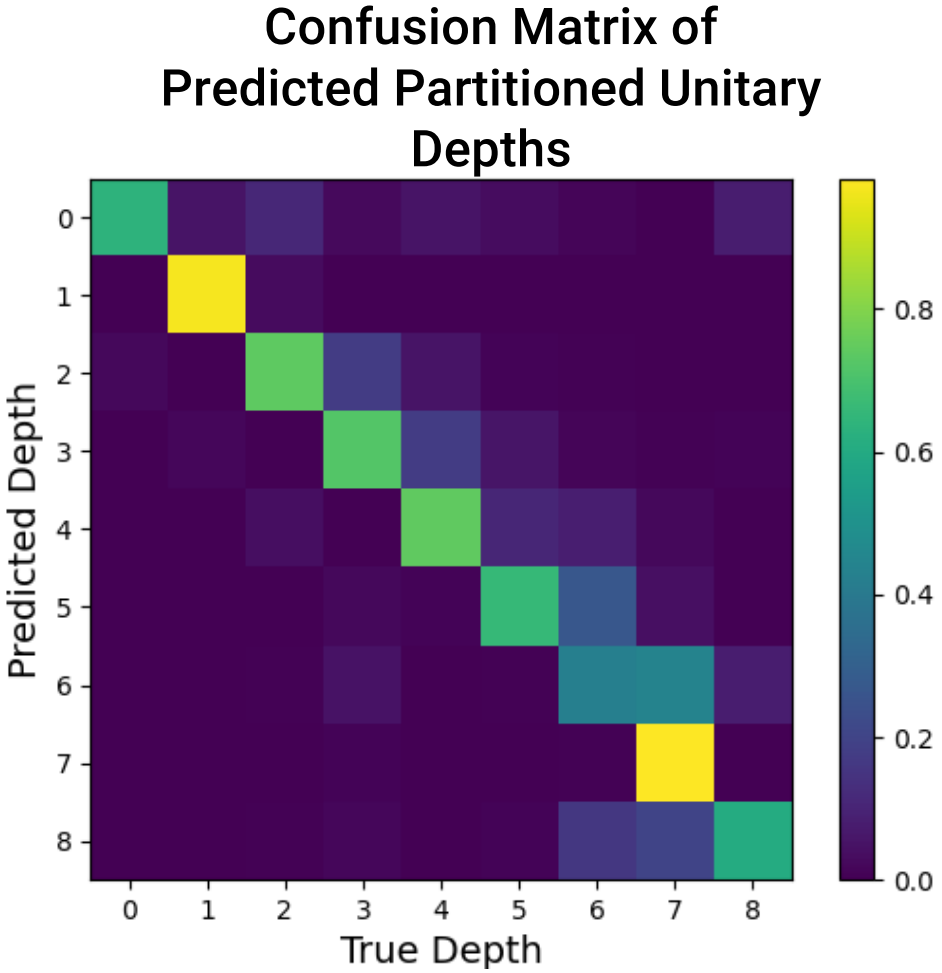


PCA quantifies how much we can compress these unitaries

Learning with Unitaries

- Neural networks can learn interesting properties of partitioned unitaries
- We can train networks to predict the depth of the minimal template that implements a unitary



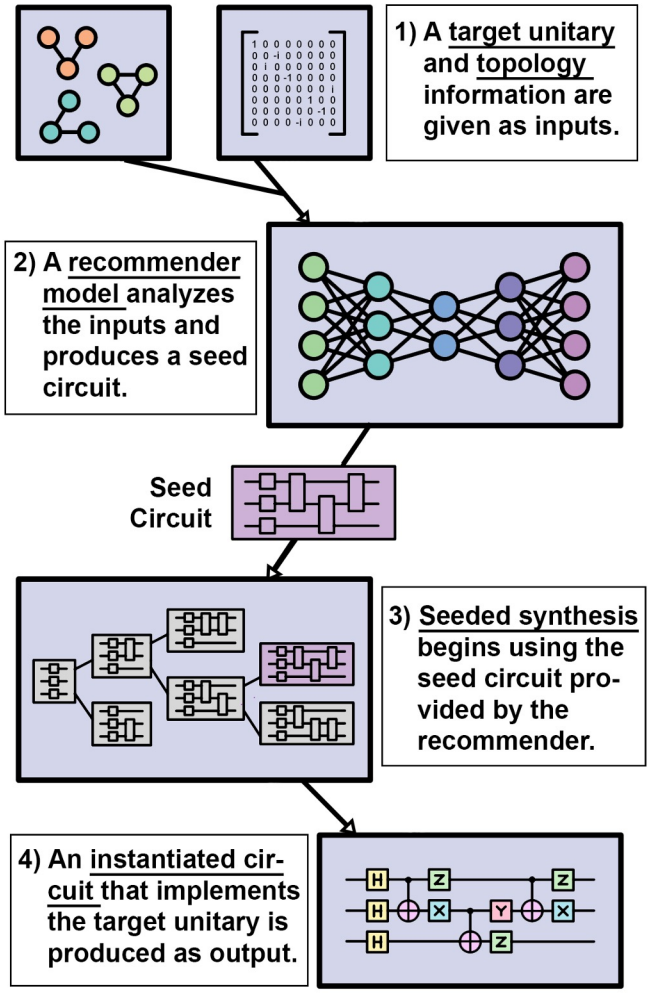


QSeed

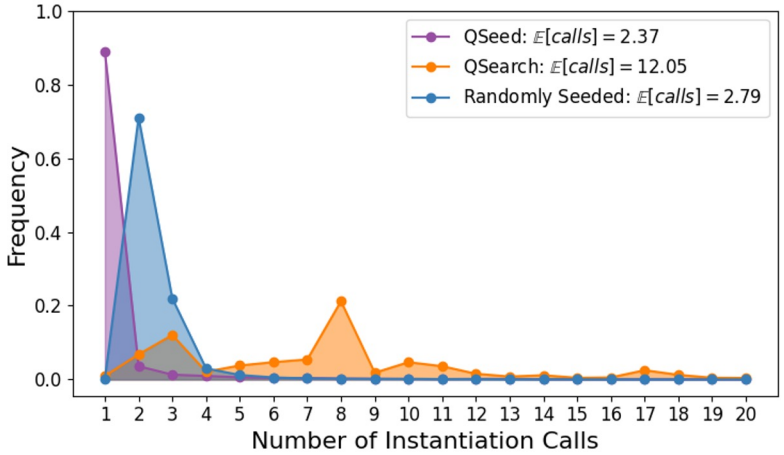
A seeded synthesis algorithm that uses machine learning to predict good seed circuits

Circuit Name	Training Widths	Test Widths
add	17, 65	41
grover	-	10
heisenberg	4, 6, 7, 8, 16, 32, 64	5
hhl	8	6
hubbard	4, 18, 50	8
mult	8, 32, 64	16
qae	11, 33, 101	65
qft	3, 4, 8, 16, 32	64
qml	4, 25, 60, 108	128
qpe	6, 10, 14	18
shor	16, 32	64
tfim	3, 4, 5, 6, 7, 8, 16, 32,	64
vqe	12, 14	18

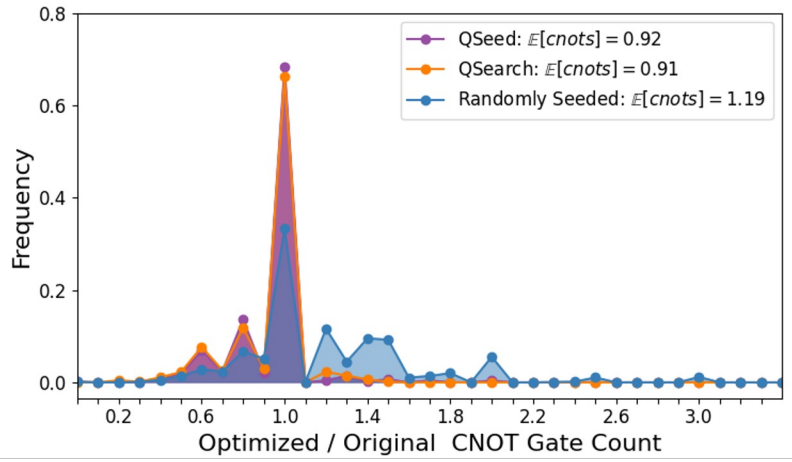
Table I: Split of circuits withheld from recommender training for testing. No Grover's algorithm circuits are used in training.



Measuring Speedup and Solution Quality

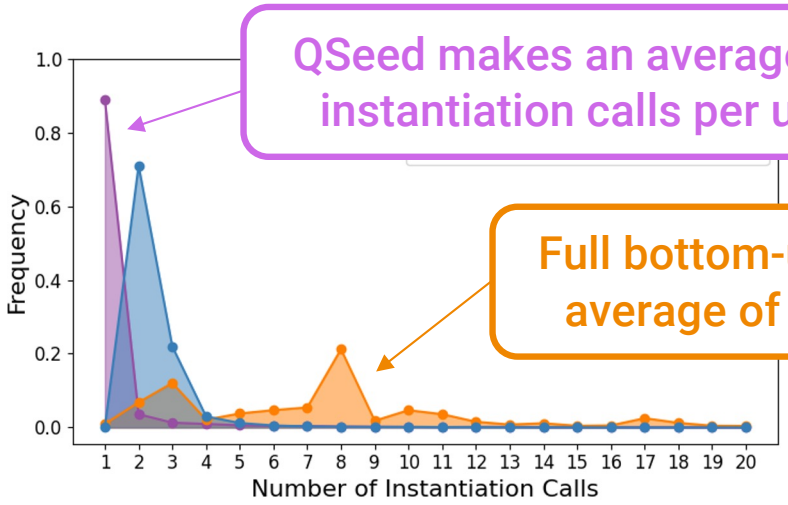


3.7X Speedup: most synthesis runs require only one instantiation call

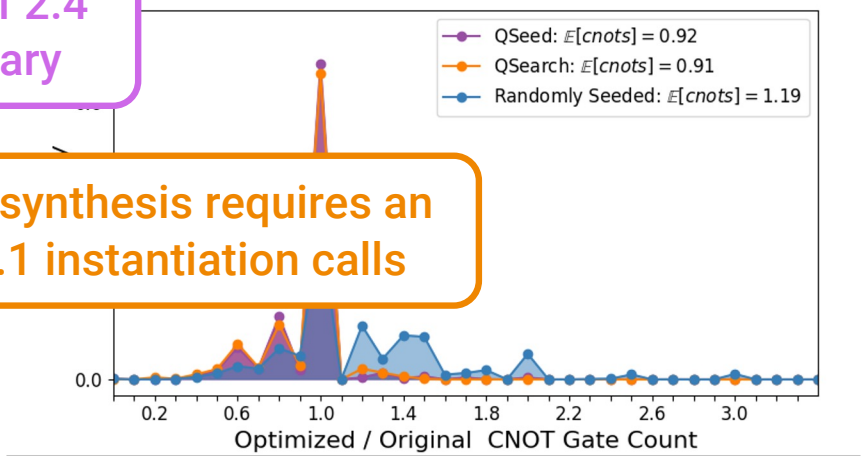


Similar solution quality: gate counts very closely match optimal implementation

Measuring Speedup and Solution Quality

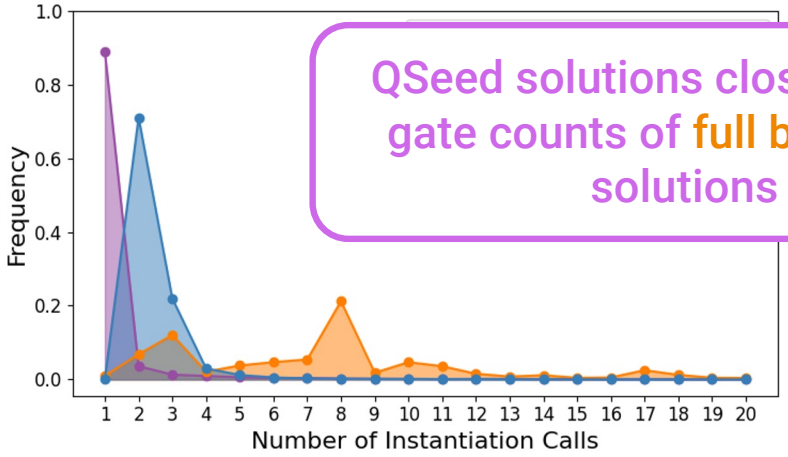


Speedup: most synthesis runs require only one instantiation call

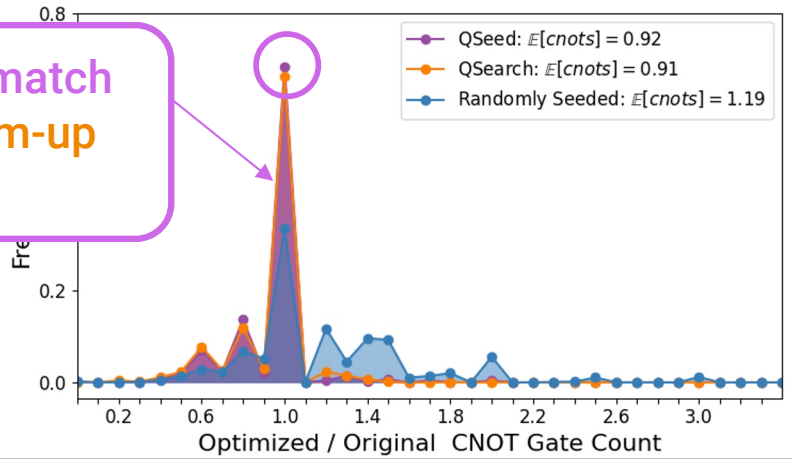


Solution quality: gate counts very closely match optimal implementation

Measuring Speedup and Solution Quality



QSeed solutions closely match gate counts of full bottom-up solutions



Speedup: most synthesis runs require only one instantiation call

Solution quality: gate counts very closely match optimal implementation

Verifying Error Thresholds

Error in synthesized unitaries for test benchmarks remains low $\Longrightarrow$ These are good generators

Circuit	Upper Bound on Approximation Error
add 41	1.4×10^{-9}
grover 10	2.4×10^{-13}
heisenberg 5	5.6×10^{-15}
hhl 6	2.2×10^{-16}
hubbard 8	3.1×10^{-14}
mult 16	2.5×10^{-14}
qae 65	3.6×10^{-9}
qft 64	2.4×10^{-9}
qml 128	4.2×10^{-14}
qpe 18	6.8×10^{-13}
shor 64	3.1×10^{-7}
tfim 64	1.8×10^{-15}
vqe 18	7.8×10^{-16}

For many benchmarks, errors are close to machine precision

Discovering Analytical, Scalable Circuit Generators

"Can I discover circuit, algorithm structure with domain science knowledge?"

Constant-Depth Circuits for Free-Fermion Dynamic Simulations on QC

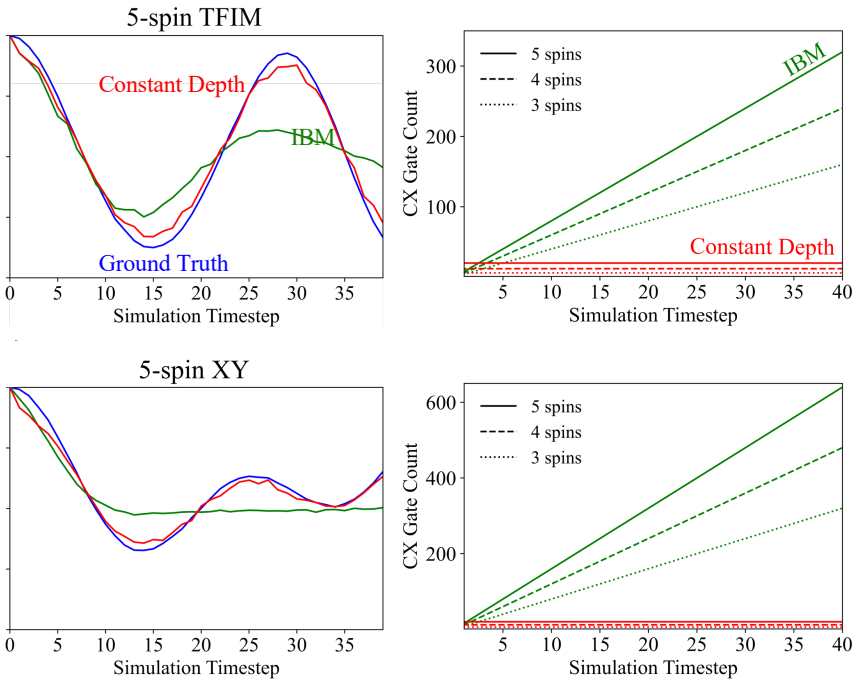
Scientific Achievement

A method for generating circuits which are constant in depth with increasing time-step, thus enabling dynamic simulations on near-term quantum computers out to arbitrarily long simulation times.

Significance and Impact

High-fidelity simulation results for long-time dynamic simulations of quantum materials can be obtained on currently available quantum computers.

N qubits in free fermion model need N(N-1) CNOTS



Comparison of simulation results and CX gate count for the TFIM and the XY model using the constant-depth circuits versus the IBM-compiled circuits.

Bassman, Van Beeumen, Younis, Smith, Iancu, de Jong
Mat. Theory 6, 13 (2022)

BQSKit finds Constant-depth Circuits for Time-evolution

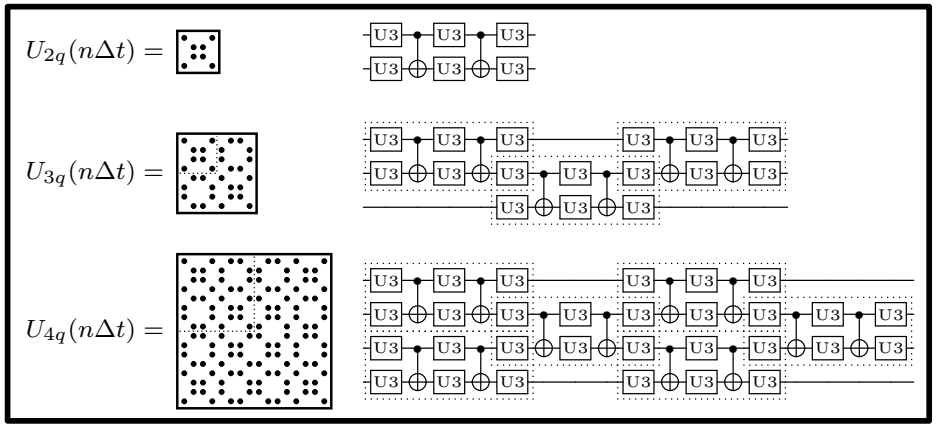
For one-dimensional TFXY, XY, IM and TFIM Hamiltonians of the form

$$H(t) = H_{xx} + H_{yy} + H_z(t) = -J_x \sum_{i=1}^{N-1} \sigma_i^x \sigma_{i+1}^x - J_y \sum_{i=1}^{N-1} \sigma_i^y \sigma_{i+1}^y - h_z(t) \sum_{i=1}^{N} \sigma_i^z$$

- Approx. time-evolution operator:

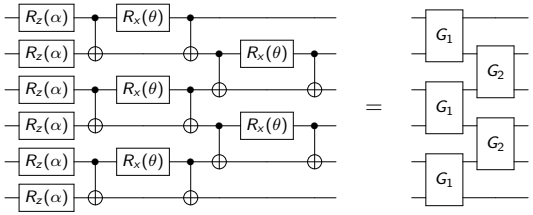
$$U_N(n\Delta t) = \prod_{i=1}^{n} e^{-i(H_{xx}+H_{yy})\Delta t} e^{-iH_z(n\Delta t)\Delta t}$$

- Independent of timestep $t_n = n\Delta t$
- Only gate parameters change
- 1-qubit gate complexity: $\mathcal{O}(N^2)$
- CNOT gate complexity: $N(N-1)$



Generalization to higher Dimensions (TF)IM, (TF)XY Models

$$U(\Delta t) = e^{-i\Delta t H_{xx}} e^{-i\Delta t H_z(\Delta t)}$$



Definition

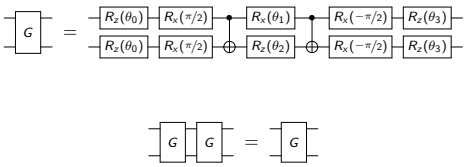
Let the matrices A and B be in $SU(2)$

$$A = \begin{bmatrix} p & q \\ r & s \end{bmatrix}, \qquad B = \begin{bmatrix} w & x \\ y & z \end{bmatrix}, \qquad \det(A) = \det(B)$$

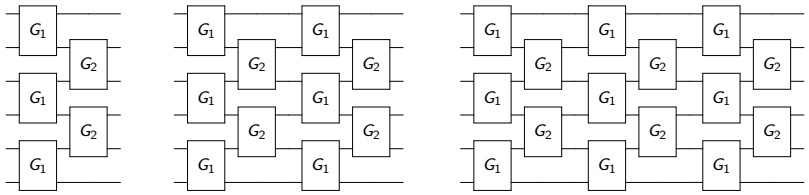
Then the 2-qubit matchgate $G(A, B)$ is defined as follows

$$G(A,B) = \begin{bmatrix} p & & & q \\ & w & x & \\ & y & z & \\ r & & & s \end{bmatrix}$$

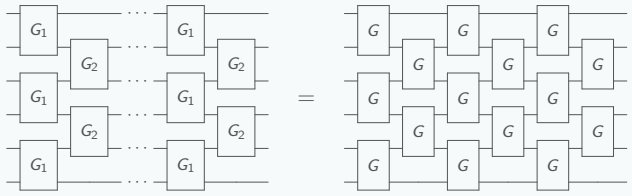
$$G = \begin{bmatrix} e^{-i(\theta_0+\theta_3)}\cos\frac{\theta_1-\theta_2}{2} & & & -ie^{i(\theta_0-\theta_3)}\sin\frac{\theta_1-\theta_2}{2} \\ & \cos\frac{\theta_1+\theta_2}{2} & -i\sin\frac{\theta_1+\theta_2}{2} & \\ & -i\sin\frac{\theta_1+\theta_2}{2} & \cos\frac{\theta_1+\theta_2}{2} & \\ -ie^{-i(\theta_0-\theta_3)}\sin\frac{\theta_1-\theta_2}{2} & & & e^{i(\theta_0+\theta_3)}\cos\frac{\theta_1-\theta_2}{2} \end{bmatrix}$$



$U(\Delta t)$ $U(2\Delta t)$ $U(3\Delta t)$ ...



$U(n\Delta t)$



Constant Depth

$O(q^2)$

BQSKit is useful during NISQ and Beyond

- Resource efficacy == Performance == Capability
  - Optimization
  - Transpilation
  - Mixed radix primitive support
  - Clifford + T support

- Hardware design exploration
- Algorithm design and generation exploration
- Mid-measurement, DQC … published soon

Open Problems

1. ## Domain science benchmark specifications (e.g. what error can I tolerate? ..)
   - Domain constraints may inform compilation
     - What does this benchmark do?
     - How to encode into programs/circuits?

2. ## Scalable circuit generation from domain science specification
   - Try generating a > 2048 qubit QFT..
   - Algorithms may not parallelize
   - Numerical stability problems may lead to **approximations**, circuit errors

3. ## It might take a data center to compile a big program

Acknowledgements: DOE ASCR, NERSC

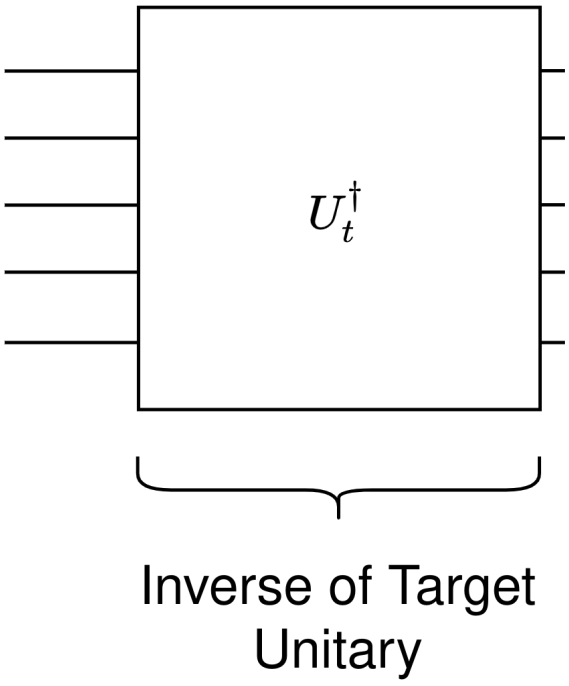
Thank You!

cciancu@lbl.gov

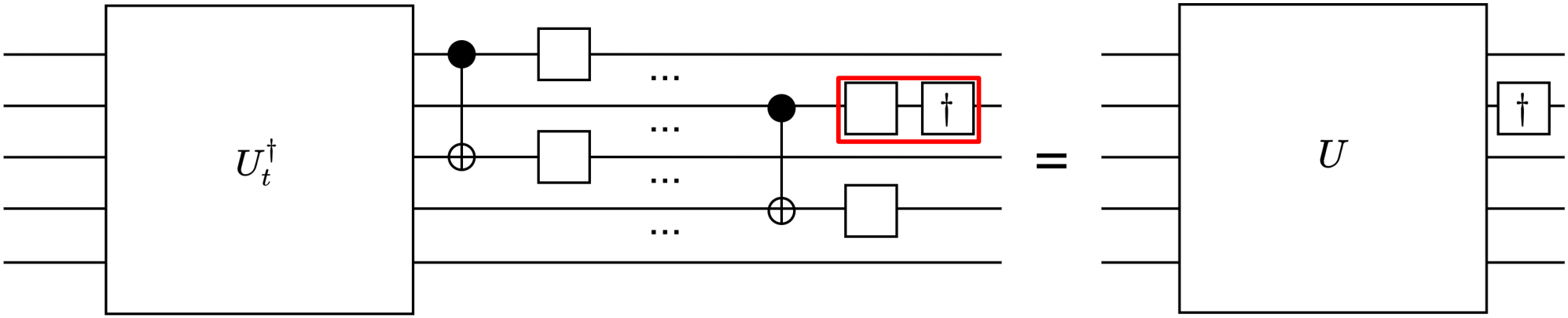
BQSKit

bqskit.lbl.gov

QFactor – Circuit Tensor Initialization

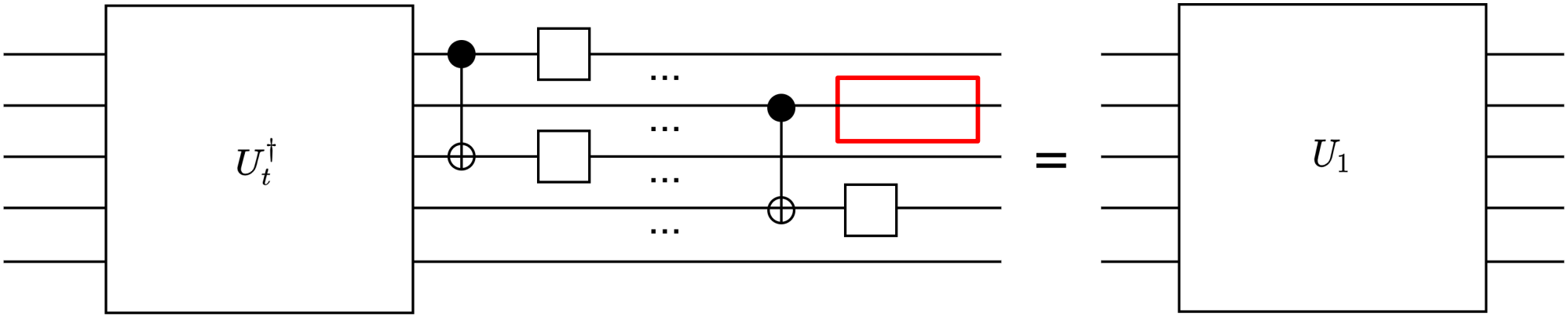


QFactor – The Sweep



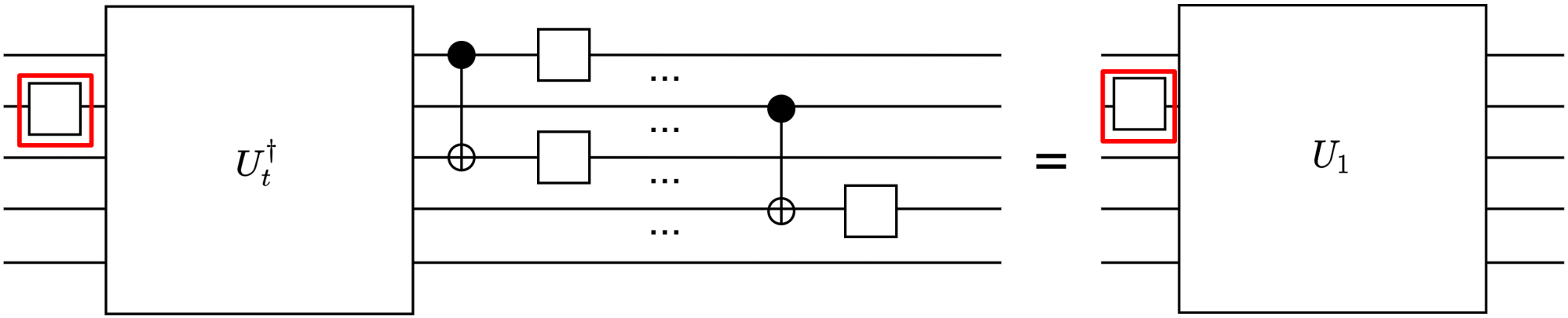
Remove gate by applying inverse on right

QFactor – The Sweep



Remove gate by applying inverse on right
Find optimal new value

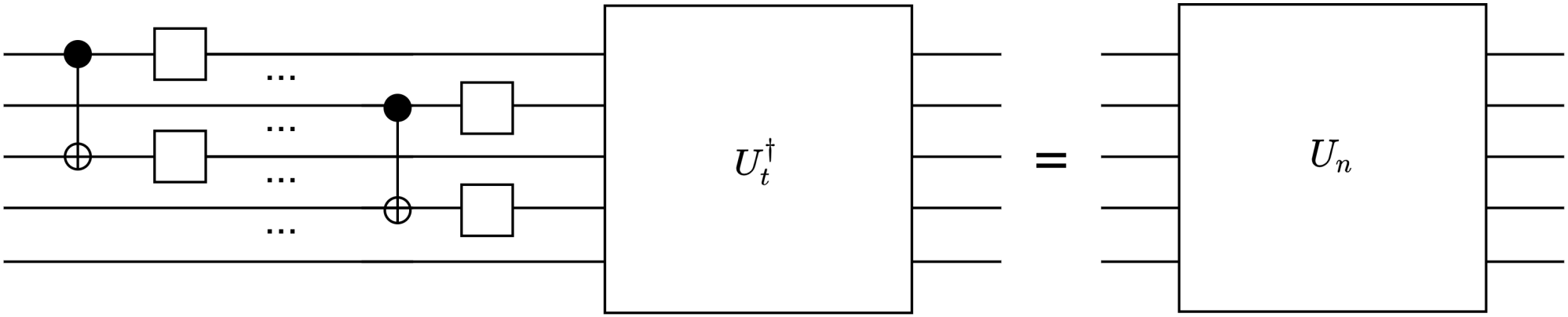
QFactor – The Sweep



Put the optimized gate back on the left, keeping the cost function trace value

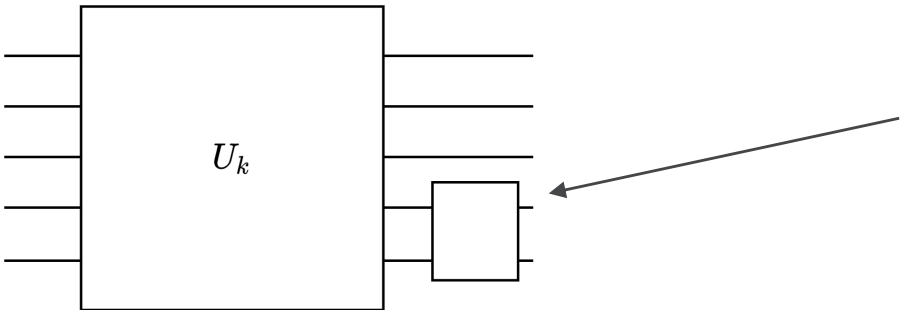
$$Tr\left(U_t^\dagger \cdot u_1 \cdots u_p\right) = Tr\left(u_p \cdot U_t^\dagger \cdot u_1 \cdots u_{p-1}\right)$$

QFactor – The Sweep



After finishing one sweep, repeat in reverse order

QFactor – Local Optimization



In this context, what is the best gate or set of parameters for the gate?

$$Tr\left(U_t^\dagger \cdot u_1 \cdots u_p\right) = Tr(U_k \cdot u_k)$$