Circuit Design for Near-Term Quantum Computers

Bhaskar Roberts and
Mathias Weiden

Why Build Quantum Computers?

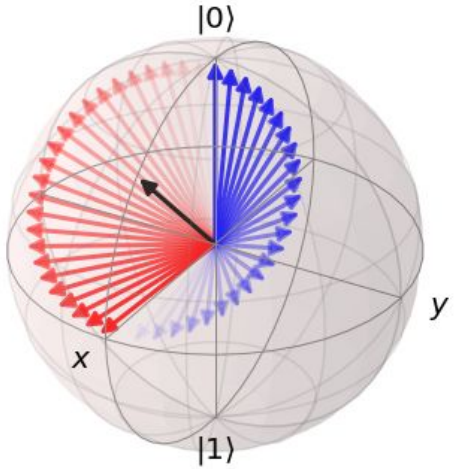
- Factoring: Shor's quantum algorithm runs in $O(n^3)$ time. No poly-time classical algorithm is known.
- Chemistry: Faster simulations of quantum systems
- Machine Learning: HLL algorithm to solve linear systems of equations
- Crypto: Unclonability is useful for copy-protection of programs and certified deletion of data.

Overview

- Near-term quantum computers are error-prone and have limited space
- Useful quantum algorithms are difficult to run on near-term devices
- We will show how to design quantum circuits to mitigate some of these issues

Theoretical Model

- We model a qubit as a vector in $\mathbb{C}^2$
- Bloch sphere models the qubit as a vector in 3-dimensional space
- Single-qubit operations rotate the vector



Single-qubit rotation (H) on the Bloch sphere [source]

Theoretical Model

Classical Concept	Quantum Analog
bit $\in \{0,1\}$	qubit $\in \mathbb{C}^2$
n-bit string $\in \{0,1\}^n$	n-qubit state $\in \mathbb{C}^{2^n}$
Logical operation	Unitary matrix
Read the state (state does not change)	Measure the state (state can change)

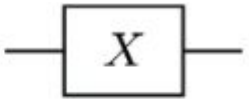
All operations, except for measurement, must be reversible (the matrix must be unitary).

Theoretical Model

- We'll model a quantum computer as a circuit.
- Common logic gates: X, H, CNOT

$\lvert\psi\rangle$	$X\lvert\psi\rangle$
$\lvert 0\rangle$	$\lvert 1\rangle$
$\lvert 1\rangle$	$\lvert 0\rangle$

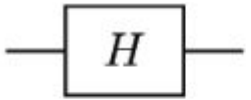
$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$



Theoretical Model

$\lvert\psi\rangle$	$H\lvert\psi\rangle$
$\lvert 0\rangle$	$\frac{1}{\sqrt{2}}(\lvert 0\rangle + \lvert 1\rangle)$
$\lvert 1\rangle$	$\frac{1}{\sqrt{2}}(\lvert 0\rangle - \lvert 1\rangle)$

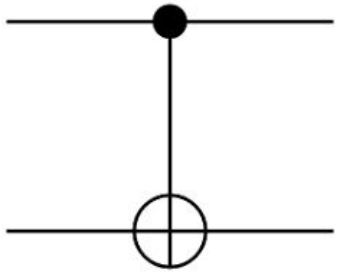
$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$



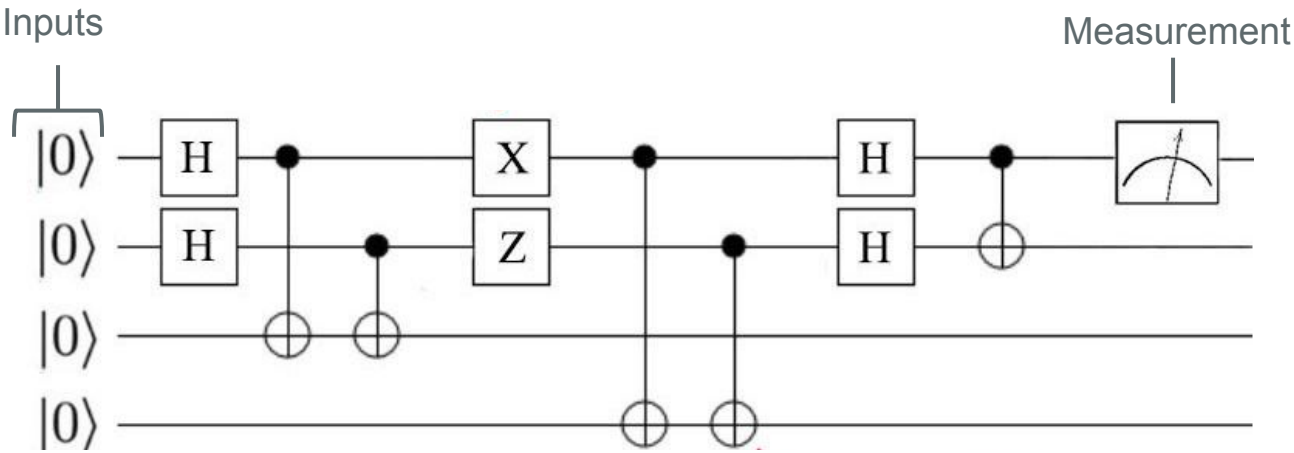
Theoretical Model

$\|\psi\rangle$	$CNOT\|\psi\rangle$
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|01\rangle$
$\|10\rangle$	$\|11\rangle$
$\|11\rangle$	$\|10\rangle$

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



Theoretical Model



An Example Quantum Circuit [source]

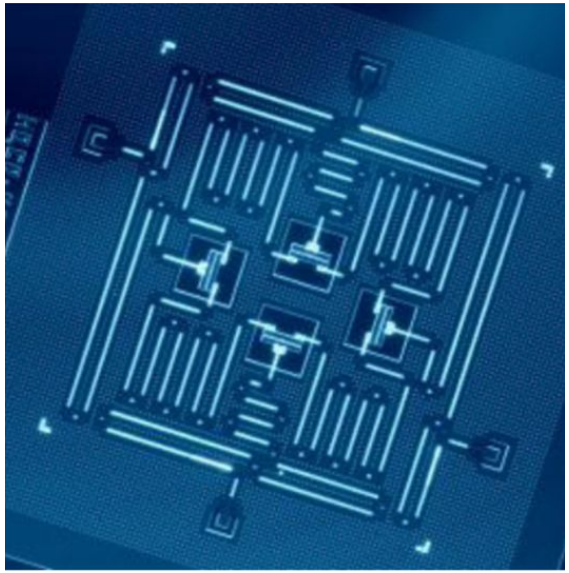
Theoretical Model

- Universal set of gates: CNOT and all single-qubit unitaries
- Often only need a few single-qubit unitaries: X, Z, H

Physical Platforms

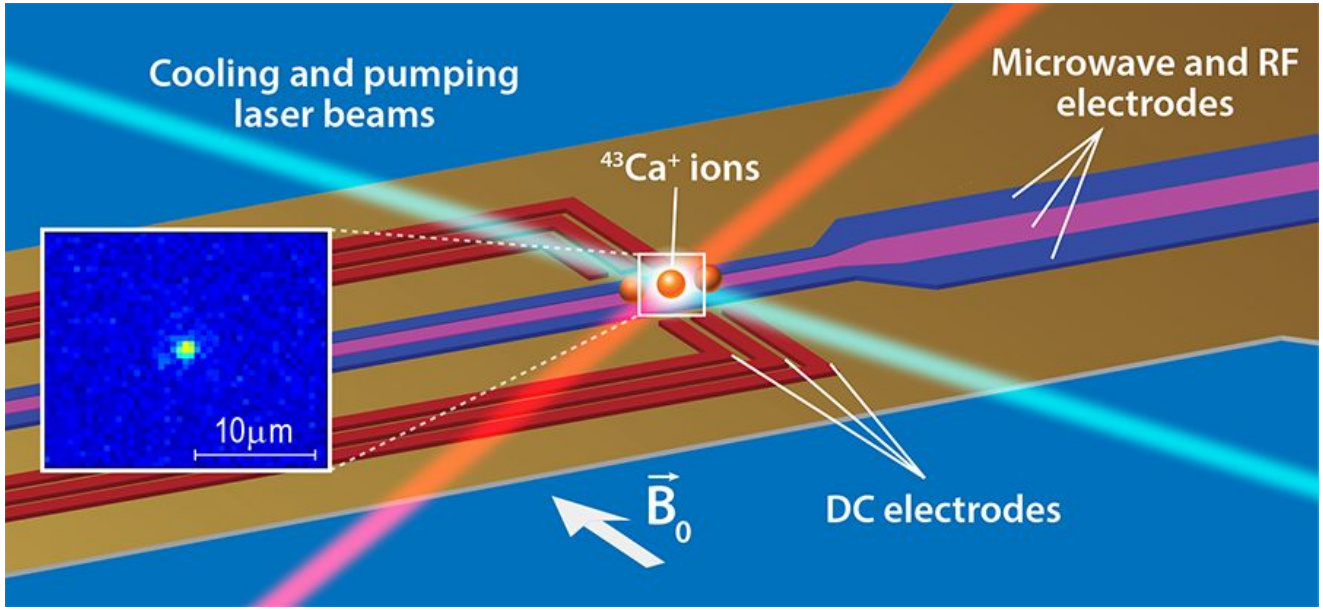
- Quantum phenomena are observable in many physical settings
- So there are many physical platforms we use to implement quantum computers
- Examples: superconducters, trapped ions, semiconductor defects, neutral atoms

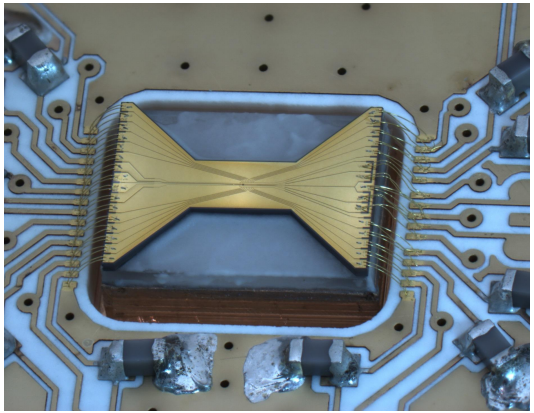
Superconducting Qubits



4 superconducting qubits

Trapped Ion Qubits





Trapped Ion vs. Superconducting Qubits

Trapped Ion

+ Longer coherence times (order of seconds)
+ All multi-qubit interactions supported within trap
+ Operate at 4K (or even closer to room temperature)
- Parallelization of certain instructions is difficult
- Scalability is questionable because of physical traps, adding more ions introduces more thermal and vibrational noise

Superconducting

+ Typically easier to fabricate
+ Tunable, as the qubits are essentially non-linear LC oscillators
+ Fast gate times
- Capacitive coupling requires qubits be neighbors to interact
- Dilution fridges are needed to cool the chips (15mK), space is very limited, scalability is questionable
- Shorter coherence times, crosstalk control noise

Physical Platforms

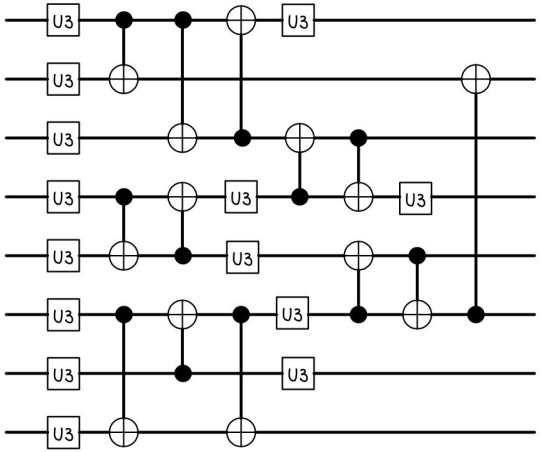
- It’s difficult to make quantum computers work, especially at large scales.
- Some difficulties are common to all platforms:
  - **Noise**: Qubits decohere quickly, and 2-qubit gates are very noisy
  - **Local Interactions**: Qubits must be nearby in order to interact
- Many important quantum algorithms require computation over many qubits

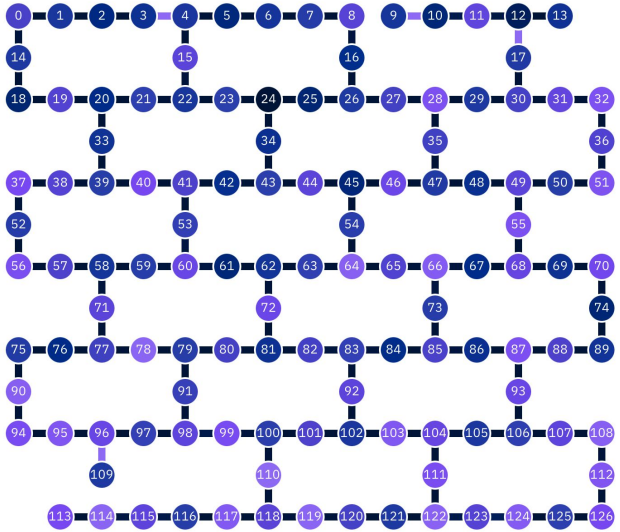
Benchmarks

- IBM’s newest quantum computer has 127 physical qubits [source]
- With current gate error rates, we need around 220 million qubits to break current RSA keys

Mapping Quantum Circuits to Hardware

- Placement
  - Equivalent to graph isomorphism problem
  - Pick an assignment of labels for each qubit in the physical topology such that the distance between qubits that interact in some program is minimized.

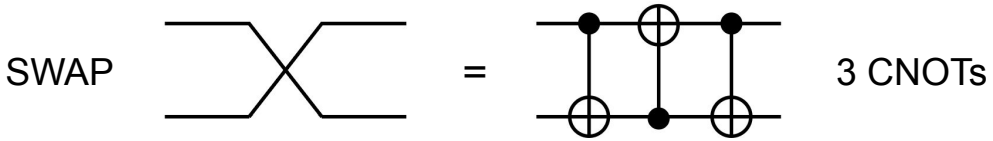


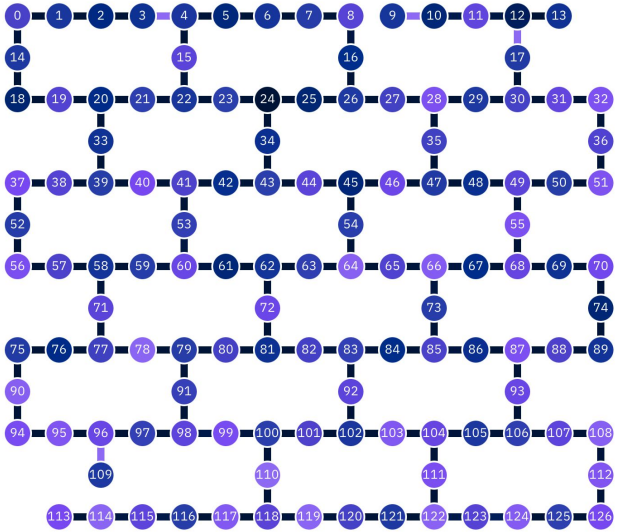


IBM Washington - 127 Qubit
Heavy Hex Topology

Mapping Quantum Circuits to Hardware

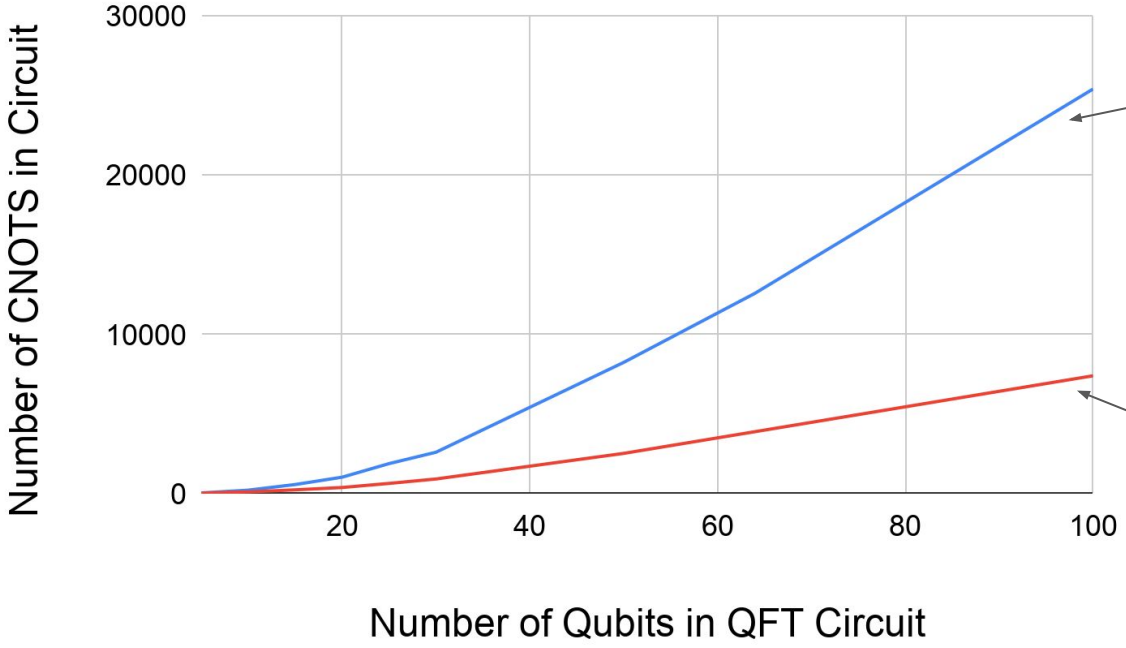
- Routing
  - Equivalent to token-swapping
  - Produce a list of SWAP operations between qubits in a physical topology so that each interaction in a program occurs along edges in the physical topology.

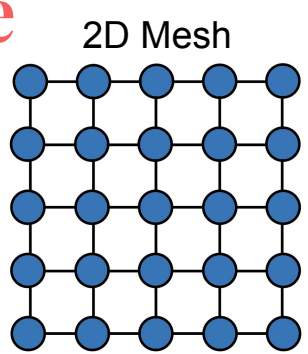


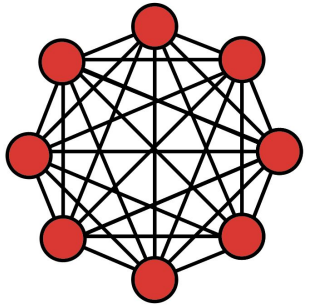


IBM Washington - 127 Qubit
Heavy Hex Topology

Mapping Quantum Circuits to Hardware







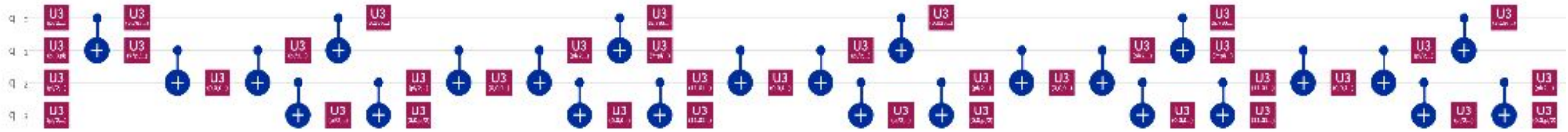
QFT Logical connectivity

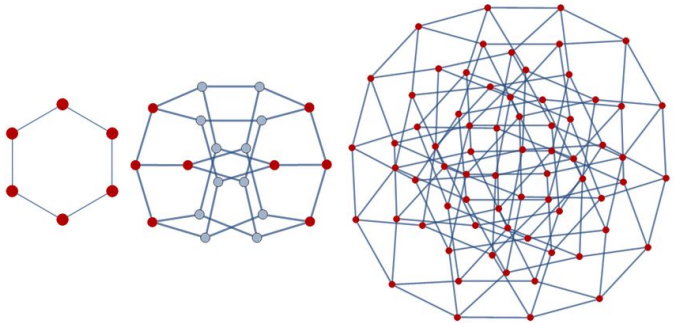
Techniques for Noise Mitigation

- Randomized compilation
  - Average out noise due to imprecise hardware
- Local algorithms
  - Design algorithms to be suited to hardware
- Circuit optimization
  - Shorten circuits

Local Algorithms

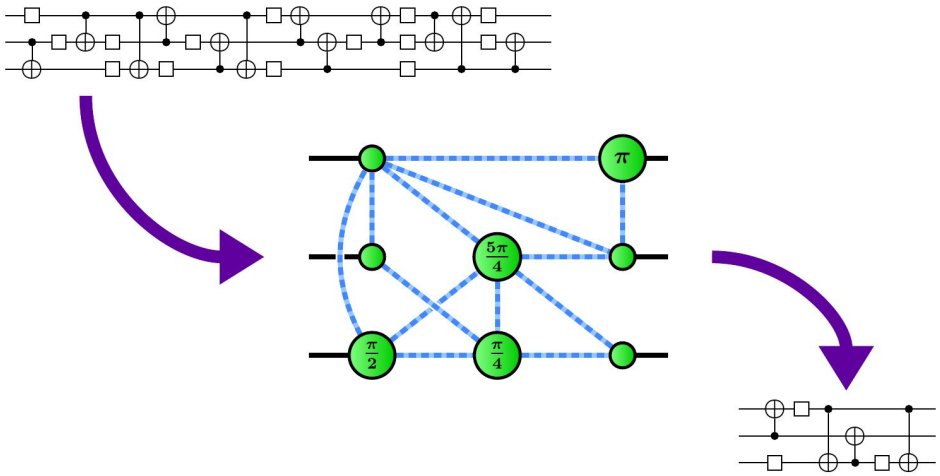
- Design algorithms that work well on limited hardware
- Alleviate or eliminate the need for mapping in some cases
- Examples include:
  - Transverse Field Ising Model circuit





Exact Circuit Optimization

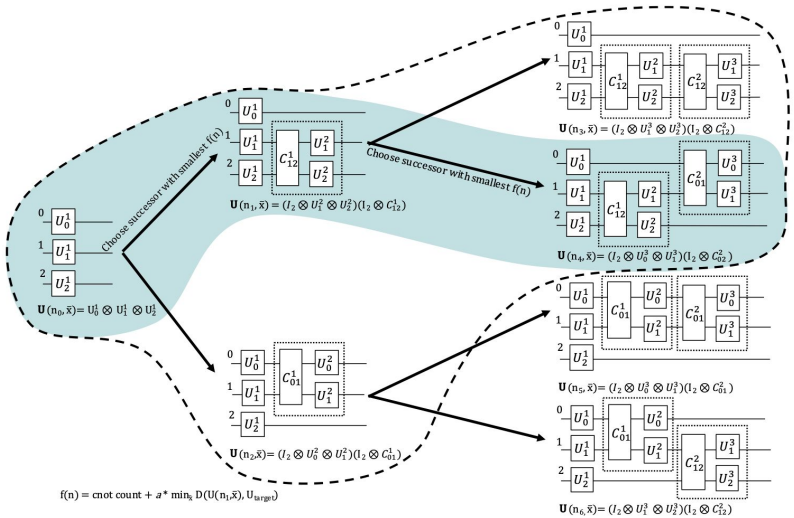
- Find circuits that implement the exact same unitary
- Lots of work being done in different representations of quantum programs - ZX calculus
- Pattern match, do peephole optimization, use substitution rules



Exact circuit optimization with ZX-Calculus

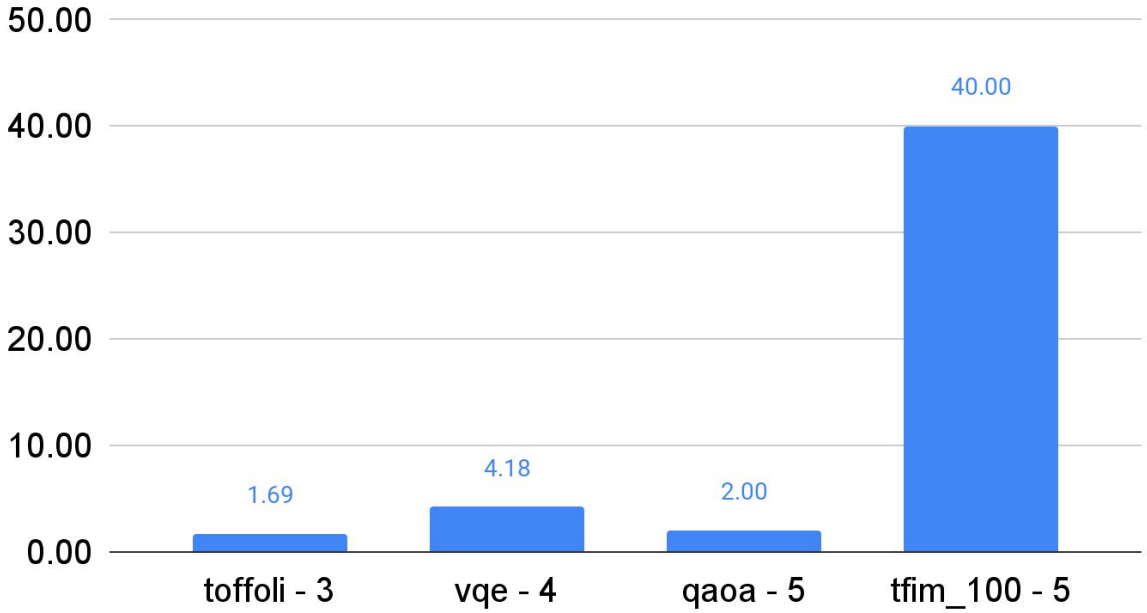
Approximate Circuit Optimization: Unitary Synthesis

- Find shorter circuit that implements unitary within some small distance of a target unitary
- Use heuristic (A*) to search space of possible solutions
- Distance metric is typically the Hilbert-Schmidt norm
- Distance is chosen to be less than expected error due to noise



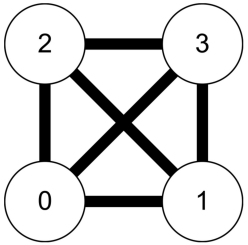
QSearch/LEAP Synthesis Algorithm

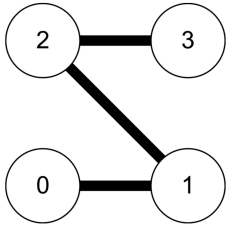
Unitary Synthesis



Improvement in CNOT Count of Circuits Mapped to Linear Topologies with QSearch vs. Qiskit (*higher is better*)

Reference Circuit

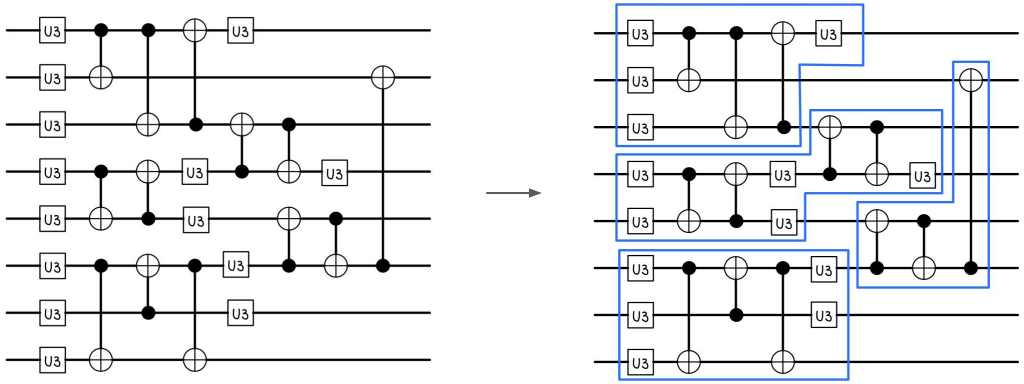




Synthesized Circuit

Unitary Synthesis

- Runtime scales exponentially with the number of qubits
- Large circuits must be partitioned into manageable subcircuits
- Circuit partitioning goal: form subcircuits with as many multi-qubit gates as possible - these tend to reduce the most
- Total distance is bounded by the sum of distances for each subcircuit



Open Questions

- Is there a way to know a priori if there is a shorter quantum circuit within some distance epsilon to some input quantum circuit or unitary?
- Can we derive lower bounds on the number of gates needed to implement some unitary given a restricted gate set?
- Given a unitary, can we predict which subtopology will result in the shortest synthesized circuit?