Scalable Quantum Circuit Cutting in a Distributed System

Shuwen Kan

Computer and Information Science Department, Fordham University

Outline

- **Background**
- Motivation
- Solution Design
- Evaluation
- Future Work

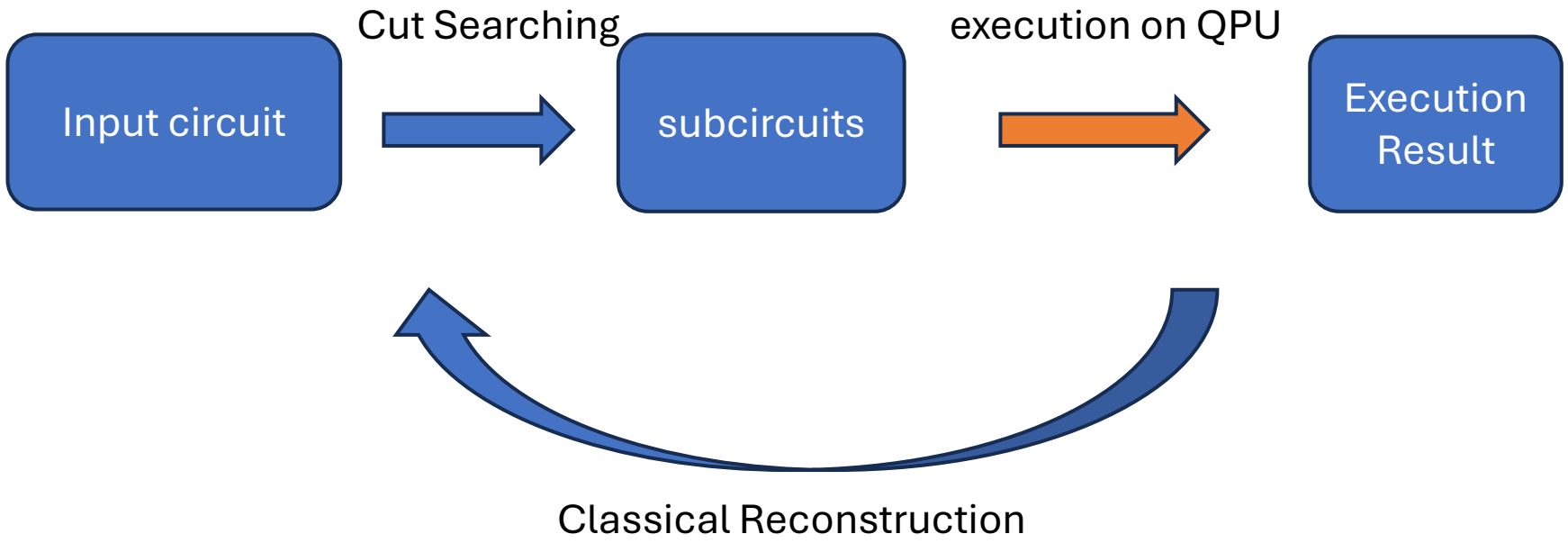
Quantum Computing in the NISQ Era

- Challenges:
  - Small: Limited qubit count.
  - Noisy: High error rate and limited coherence time

- Solution: Circuit Cutting
  - Enables evaluation of large circuits
  - Improves fidelity

Circuit Cutting: a hybrid approach

- Cut a large quantum circuit into smaller subcircuits, execute, and reconstruct the original result.



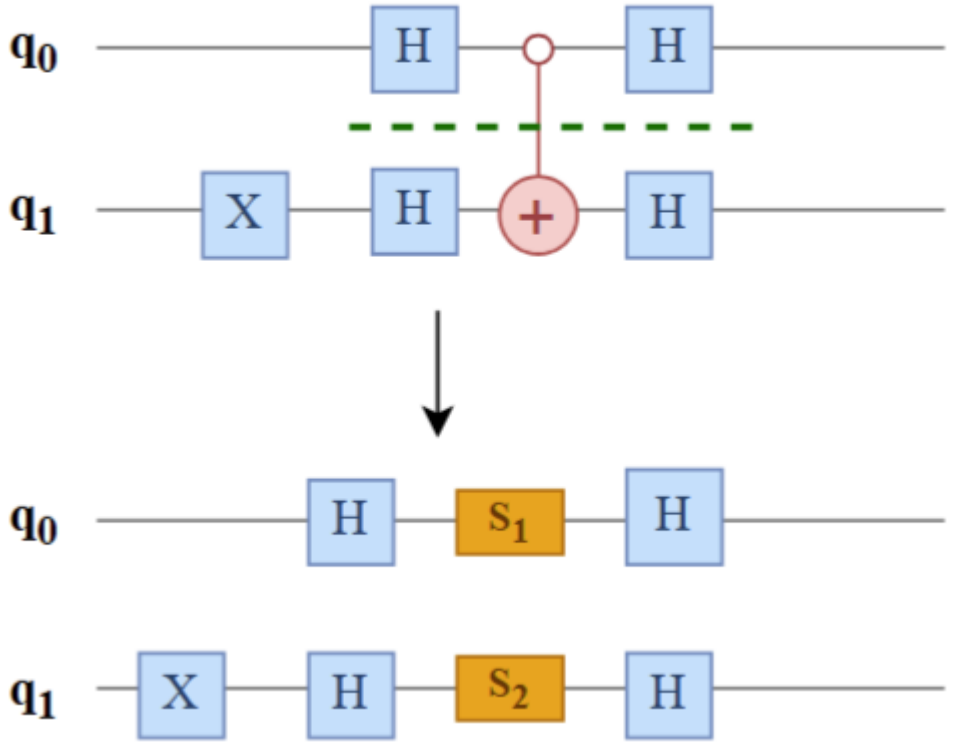
- Only need to execute the subcircuits on QPU, thereby reducing the demand for a highly powerful QPU.

Circuit Cutting: Gate Cutting

- Gate Cutting: replace two-qubit gate with local operations using Quasi-Probability Decomposition Simulation(QPD)

$$\mathcal{E} = \sum_{i=1}^{m} a_i \mathcal{F}_i, \quad \gamma = \left(\sum_i |a_i| \right)^2$$

- Where $\mathcal{E}$ is two-qubit gate and $\mathcal{F}$ is local gate
- Coefficient a corresponds to the reconstruction process and also the number of samples that need to be taken in QPD.



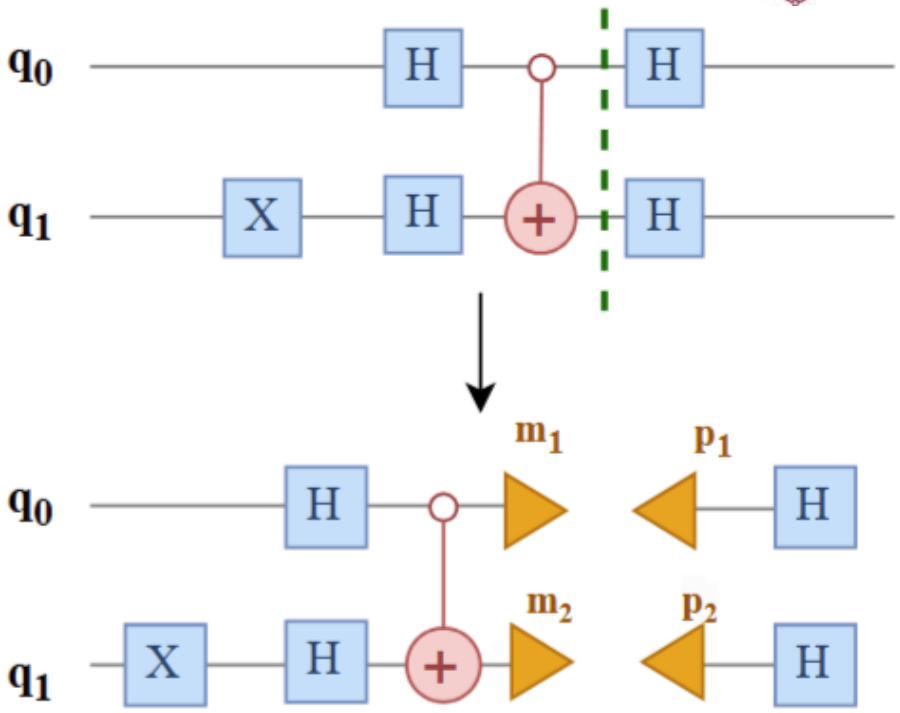
Circuit Cutting: Wire Cutting

- Wire Cutting: a measure-and-prepare channel

$$\mathbf{A} = \frac{A_1 + A_2 + A_3 + A_4}{2}$$

where

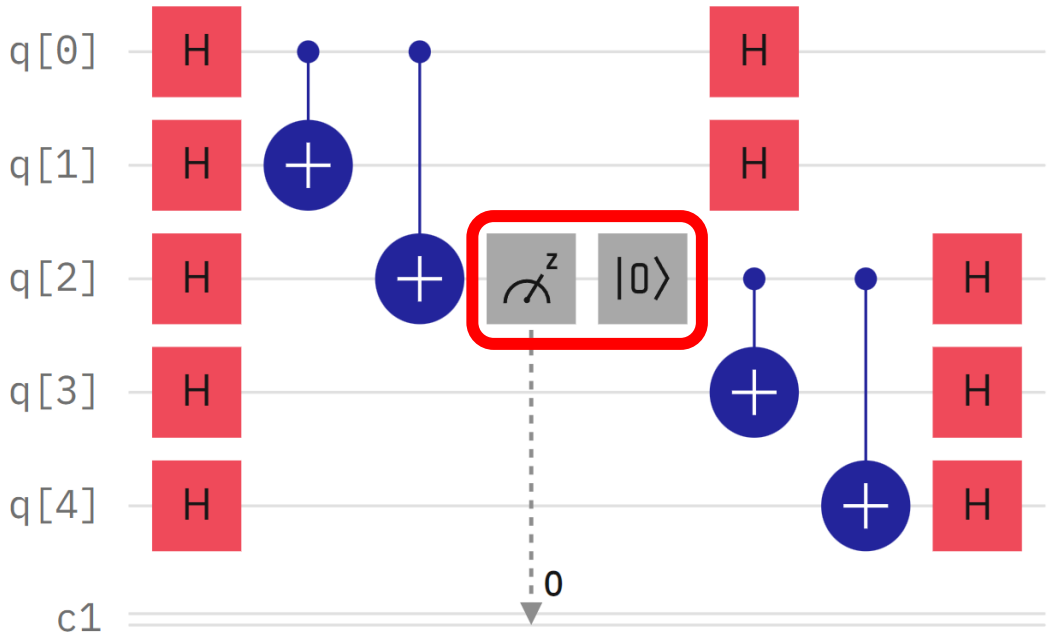
$$\begin{aligned} A_1 &= [Tr(\mathbf{A}I) + Tr(\mathbf{A}Z)] \, |0\rangle \langle 0| \\ A_2 &= [Tr(\mathbf{A}I) - Tr(\mathbf{A}Z)] \, |1\rangle \langle 1| \\ A_3 &= Tr(\mathbf{A}X)[2 \, |+\rangle \langle +| - |0\rangle \langle 0| - |1\rangle \langle 1|] \\ A_4 &= Tr(\mathbf{A}Y)[2 \, |+i\rangle \langle +i| - |0\rangle \langle 0| - |1\rangle \langle 1|] \end{aligned}$$

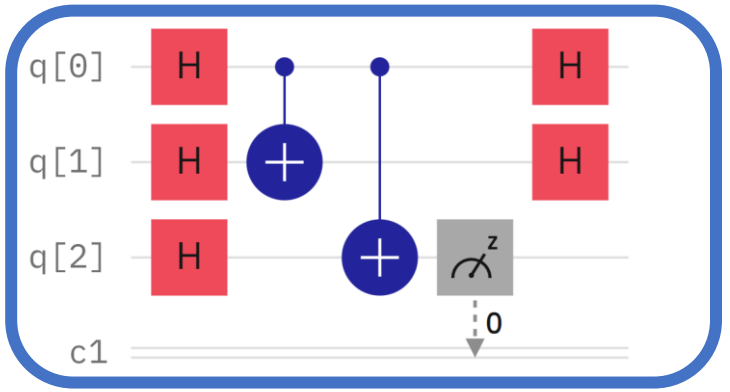


- Where each trace operator corresponds physically to measure the qubit in one of the Pauli bases.
- Each of the density matrices corresponds physically to initialize the qubit in one of the eigenstates.

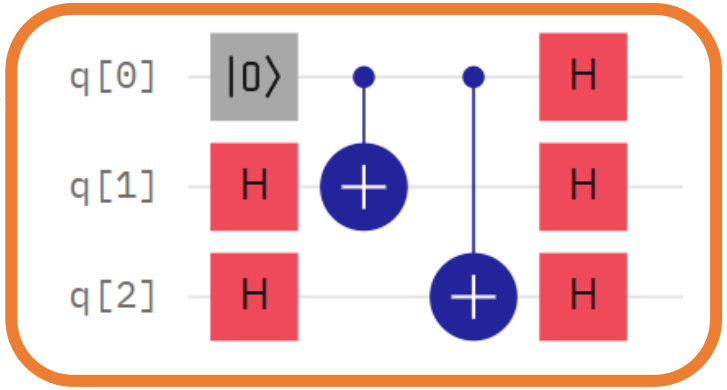
Circuit Cutting Overhead

- With each wire cut,
  - Quantum Cost: Multiple variations of subcircuits
  - Classical Cost: Exponential computation





To be measured in I, X, Y, Z basis

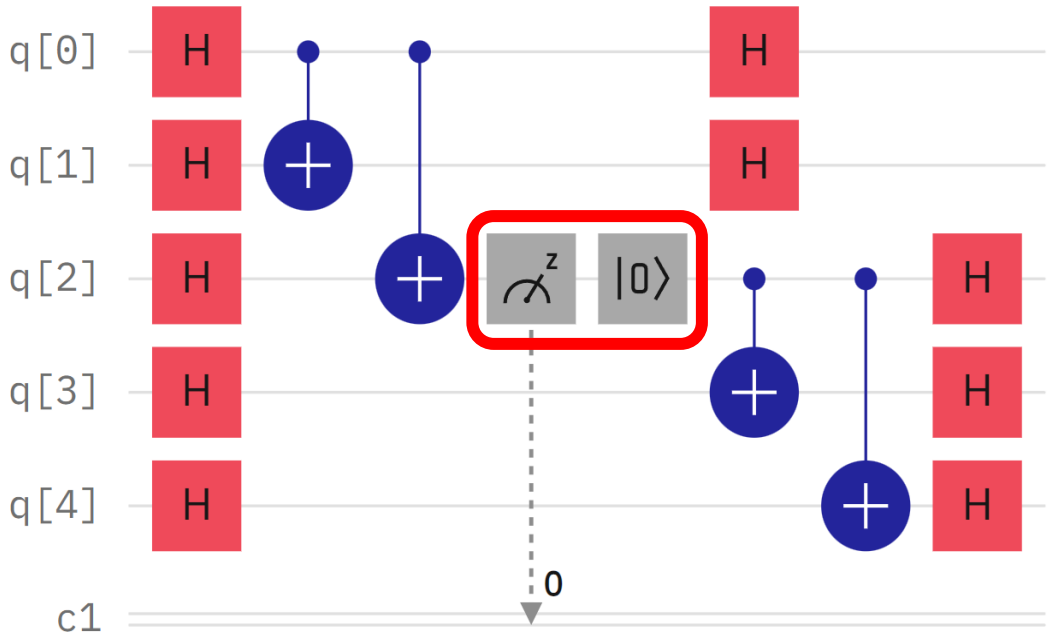


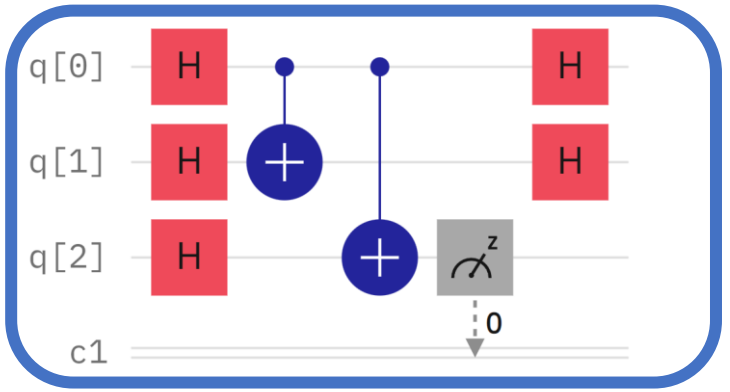
To be prepared in $|0\rangle$, $|1\rangle$, $|+\rangle$, $|i\rangle$ states

Reconstruction cost for i-th cut:

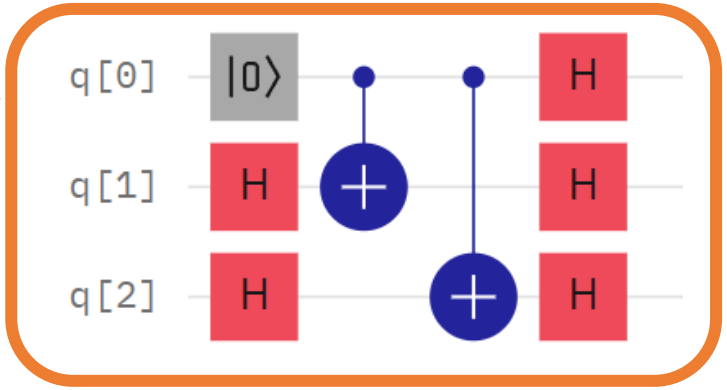
$$\bullet \sum_i c_i O_i \otimes \rho_i$$

- c_i is the coefficient
- O_i is the measurements in X,Y,Z basis
- ρ_i is the initialization of $|0\rangle$, $|1\rangle$, $|+\rangle$, $|i\rangle$ states





To be measured in I, X, Y, Z basis

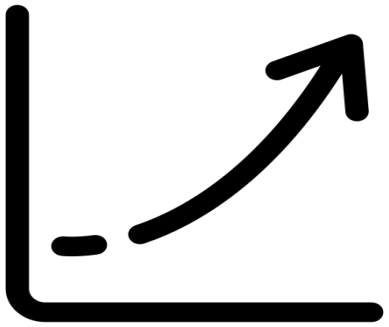


To be prepared in $|0\rangle$, $|1\rangle$, $|+\rangle$, $|i\rangle$ states

Outline

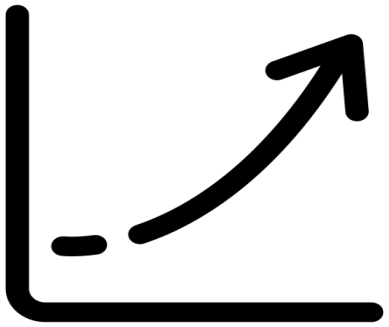
- Background
- **Motivation**
- Solution Design
- Evaluation
- Future Work

Motivation

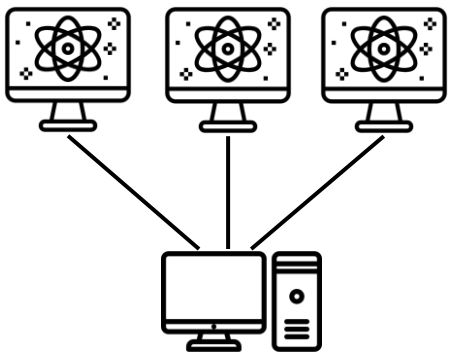


- Cost of circuit cutting is **exponential** in terms of number of cuts

Motivation



- Cost of circuit cutting is **exponential** in terms of number of cuts

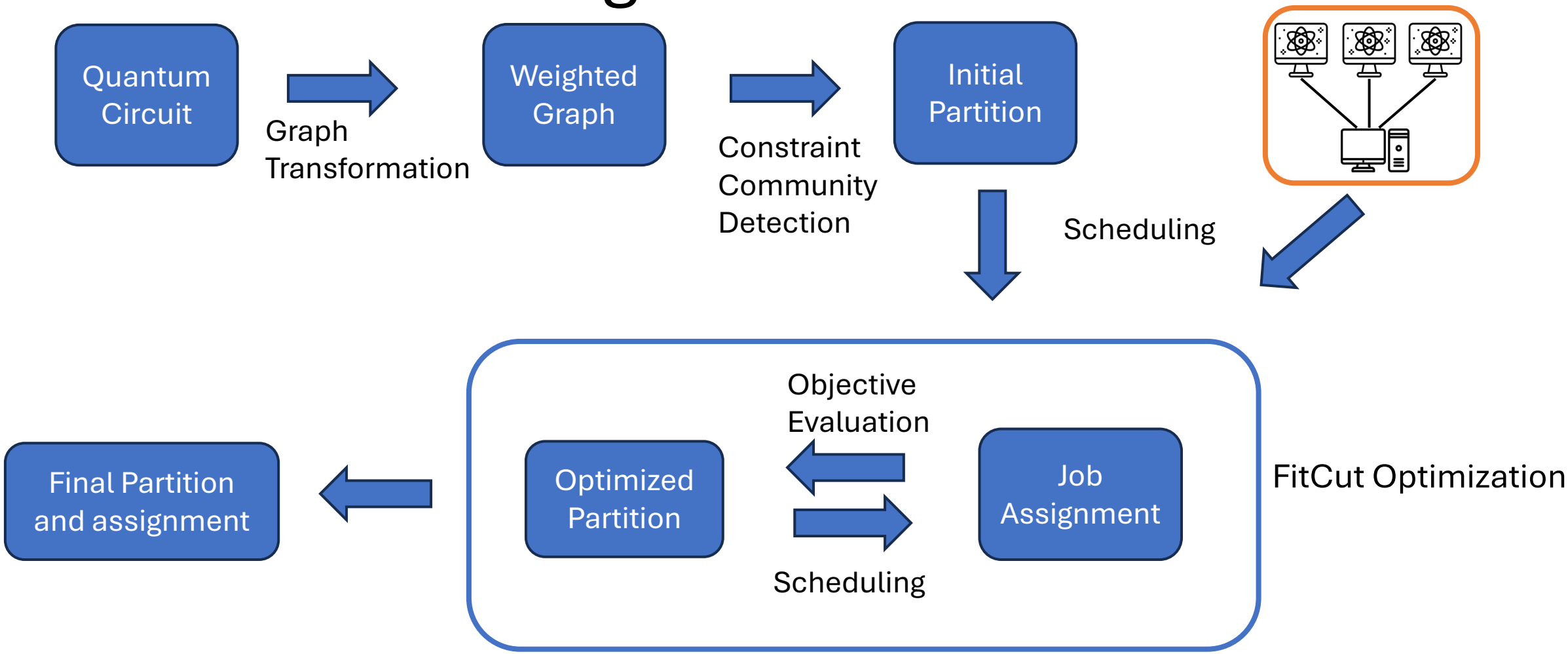


- Maximize the available quantum resource by a distributed system

Outline

- Background
- Motivation
- **Solution Design**
- Evaluation
- Future Work

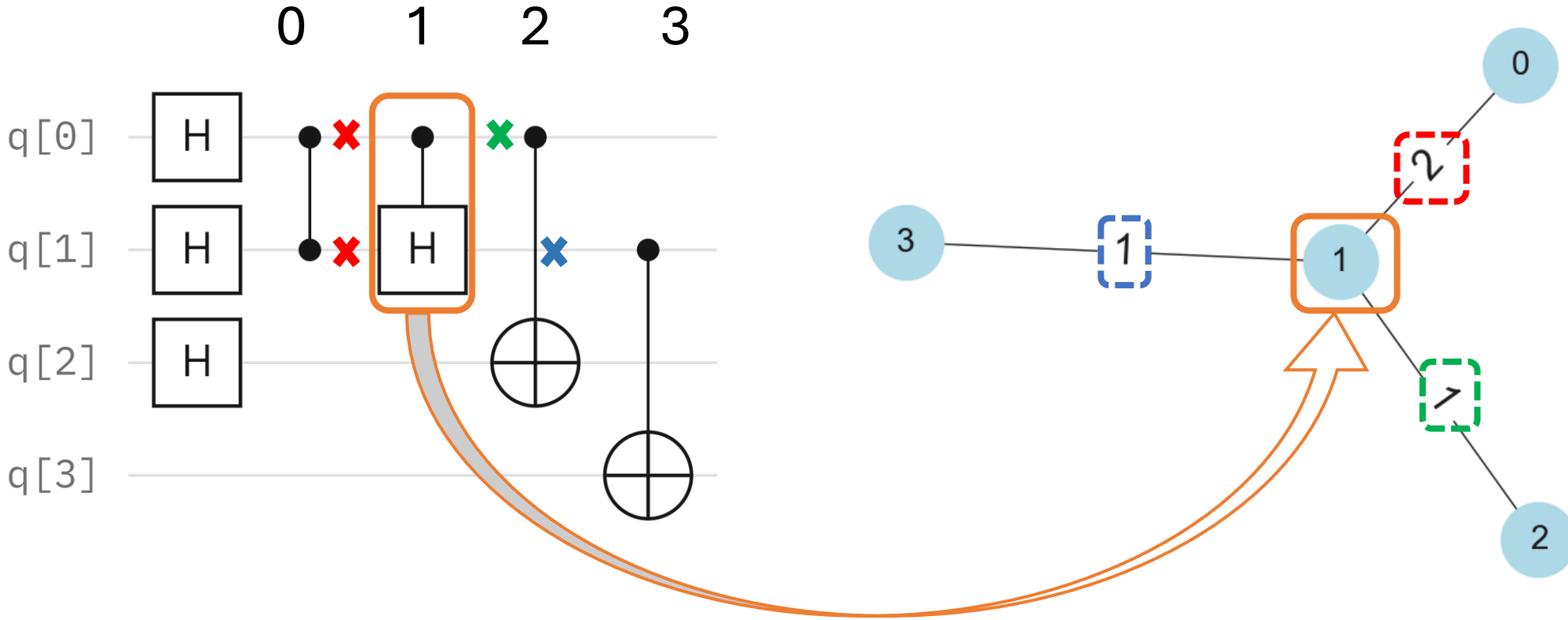
FitCut: Efficient circuit cutting and resource-aware scheduling



Step 1: Circuit to Graph Transformation

Convert the circuit to a weighted graph where:

- Each node represents a two-qubit gate
- Each edge represents the circuit wire



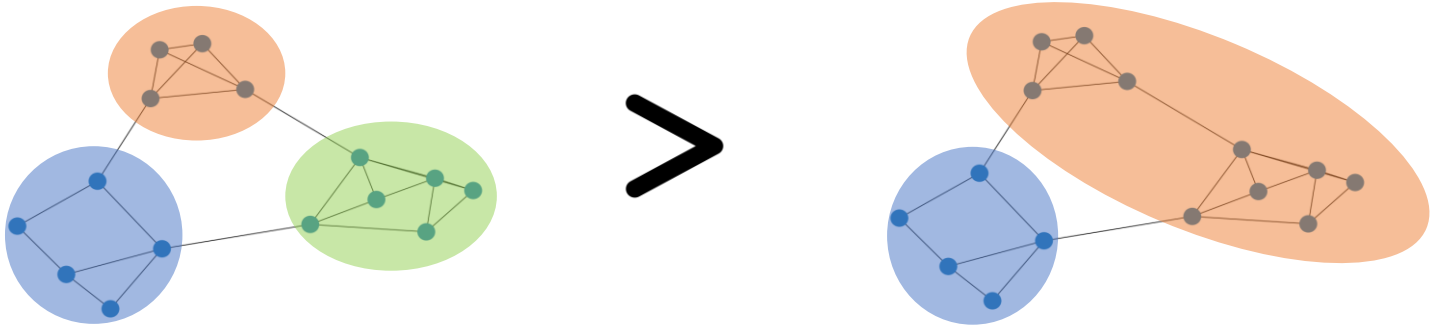
Step 2 : Modularity-based Community Detection

- Maximize modularity: $Q = \frac{1}{2m}\sum_{com}\sum_{i,j}(A_{i,j} - \frac{k_i k_j}{2m})$

actual edges within a community

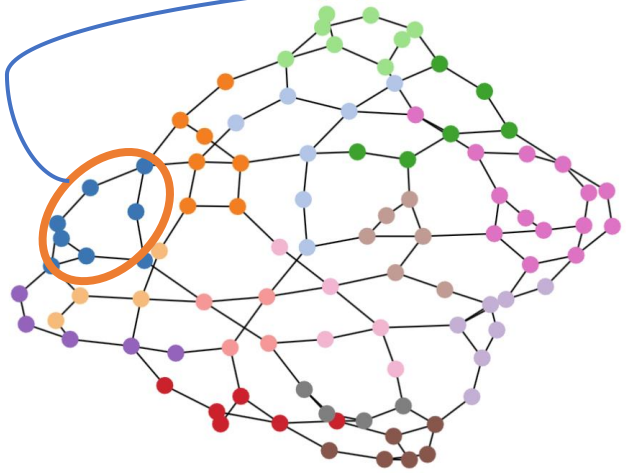
the expected number of edges in a random network

- Where:
  - m is number of edges
  - $A_{i,j}$ is adjacency matrix representing weight of edge(i,j)
  - k_i, k_j are the degrees of nodes i and j
- Modularity measures dense connections within communities and sparse connections between them.

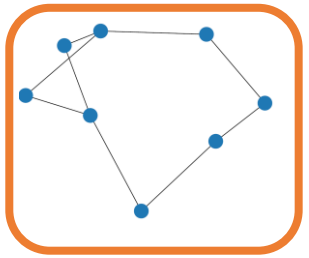


Step 2 : Constraint on Community Detection

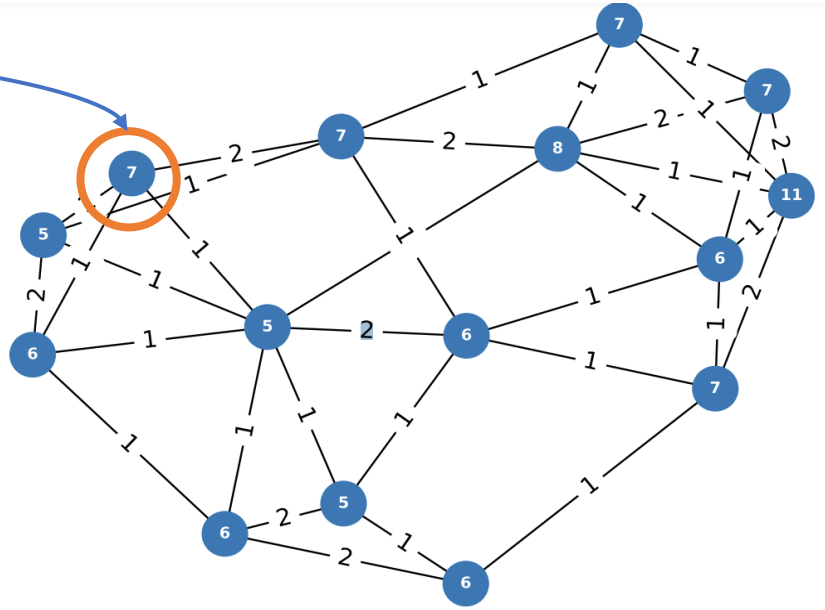
- Modularity-only solution will be our initial solution represented by a merged graph
  - Merge each community into a super node
  - Combine all edges between communities as a super edge



Example of a 7*8 supremacy circuit



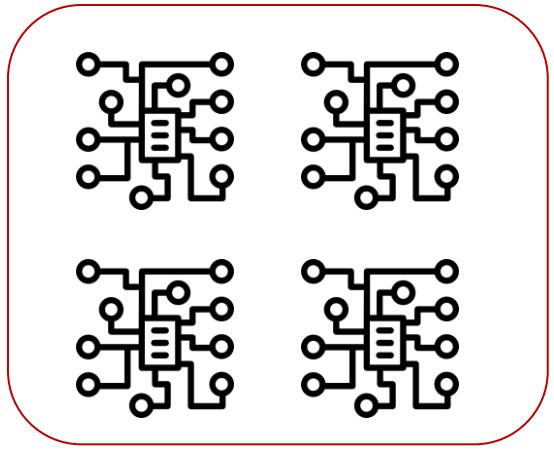
Community
=> 7-qubit Subcircuit



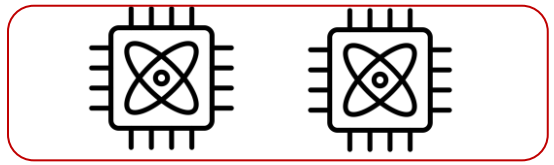
- Constraint: subcircuit size must be less than half of qubit counts on largest worker

Step 3: Resource-aware Scheduling

- Scheduling: assign subcircuits to different quantum workers in the system
  - Job: subcircuit with *d* depth and *w* width
  - Worker: quantum worker with *qc* qubits



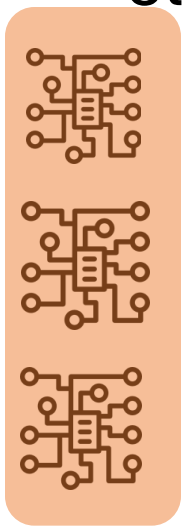
Job 1-3: (5-qubit,10), job 4: (10-qubit,10)



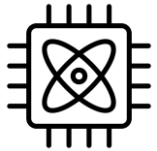
Worker 1: 5-qubit, worker 2: 10-qubit

Step 3: Resource-aware Scheduling

Initial Assignment: Assigning subcircuit to the quantum worker with closest qubit counts

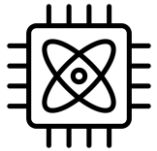


Job 1-3: (5-qubit,10)



5-qubit worker 1: depth 30

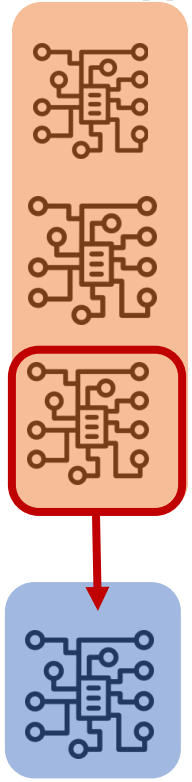
Job 4: (10-qubit,10)



10-qubit worker 2: depth 10

Step 3: Resource-aware Scheduling

Redistribute the jobs from overloaded worker to underutilized worker

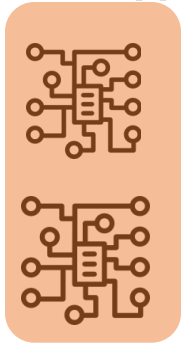


Job 1-3: (5-qubit,10) → 5-qubit worker 1: depth 30

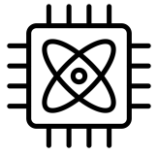
Job 4: (10-qubit,10) → 10-qubit worker 2: depth 10

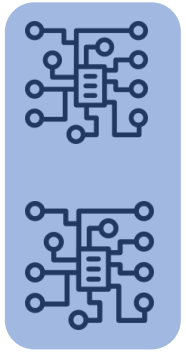
Step 3: Resource-aware Scheduling

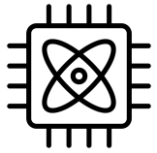
Redistribute the jobs from overloaded worker to underutilized worker



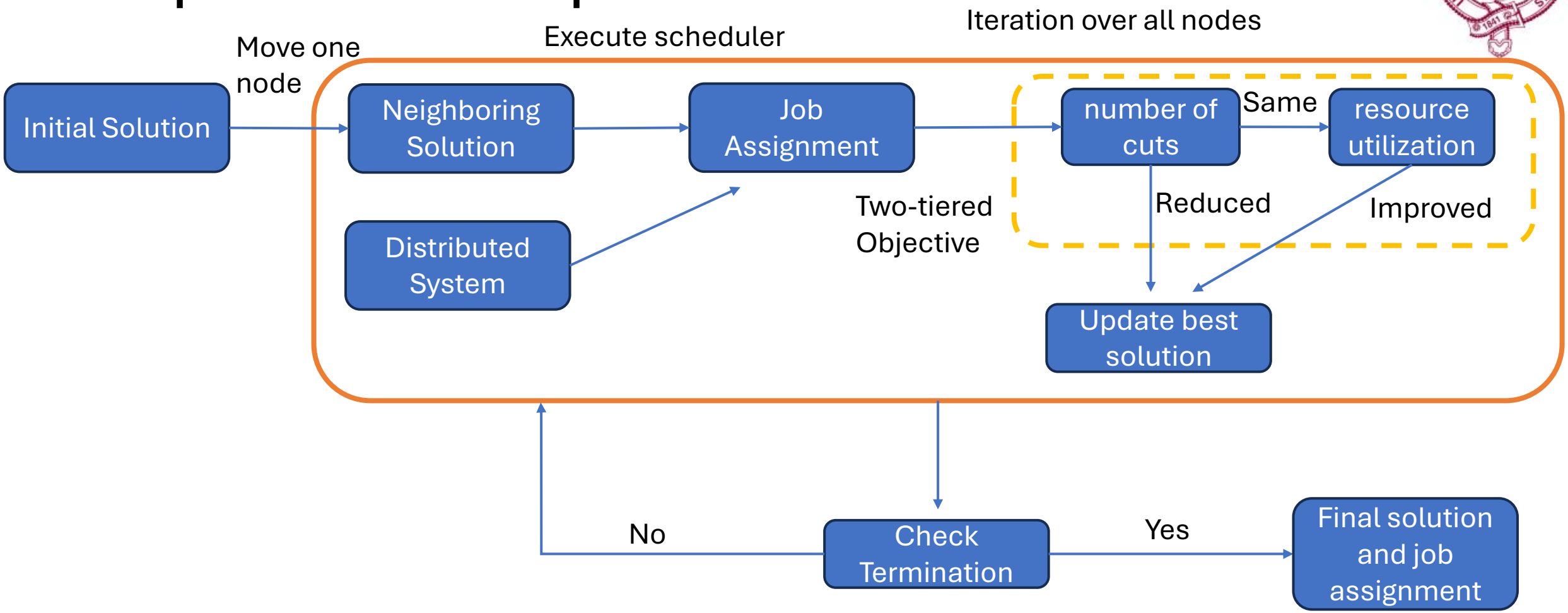
Depth-based resource utilization rate is calculated as:

Job 1,2: (5-qubit,10) →  5-qubit worker 1: depth 20 $\frac{\frac{5}{5}*20}{20} = 100\%,$



Job 3: (5-qubit,10)
Job 4: (10-qubit,10) →  10-qubit worker 2: depth 20 $\frac{\frac{10}{10}*10+\frac{5}{10}*10}{20} = 75\%$

Step 4: FITCUT optimization



- Termination Criteria: no improvements are made during one round of iteration
- This process is stochastic and affected by the order of nodes evaluated

Outline

- Background
- Motivation
- Solution Design
- **Evaluation**
- Future Work

Evaluation

Search Time: execution time for searching optimal solution

Number of Cuts: circuit cutting cost

System-wide Resource Utilization: quantifies the resource utilization in a heterogenous multi-worker system

Methodology

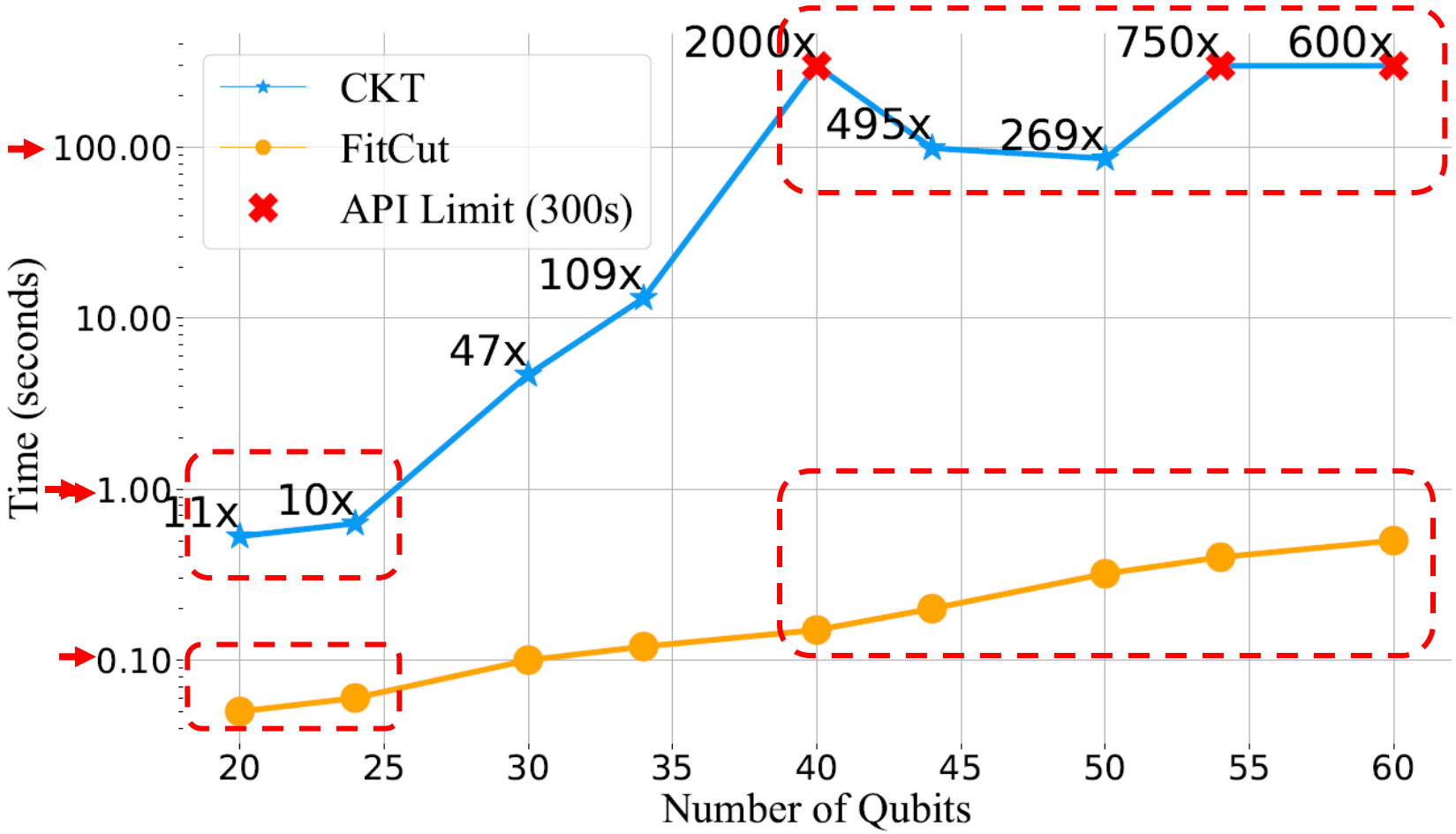
- Benchmarked on 4 types of quantum algorithms:
  - Adder
  - Bernstein-Vazirani
  - Hardware-efficient ansatz
  - Supremacy
- Circuit Size: 20-qubit to 100-qubit
- Constraints: 15-qubit QPU and 20-qubit QPU
- Result: FitCut is executed 50 trials.
  - Execution time is the average of 50 trials
  - Number of cuts is the range of results

Experiment Settings:

- Qiskit Addon Cutting 0.6.0: Automates the process of finding optimal circuit cuts.
  - Uses an optimization solver for a Mixed-Integer Programming (MIP) model.
  - Imposes a 300-second time limit if the solution space cannot be fully explored.
- Software dependencies:
  - IBM Qiskit 1.02, Networkx 3.3
  - Qiskit Addon Cutting 0.6.0, IBM ILOG CPLEX Optimization Studio 22.1.1.0
- Hardware: AMD Ryzen 7 6800H processor running at 3.2 GHz.

Search Time and Number of Cuts Comparison

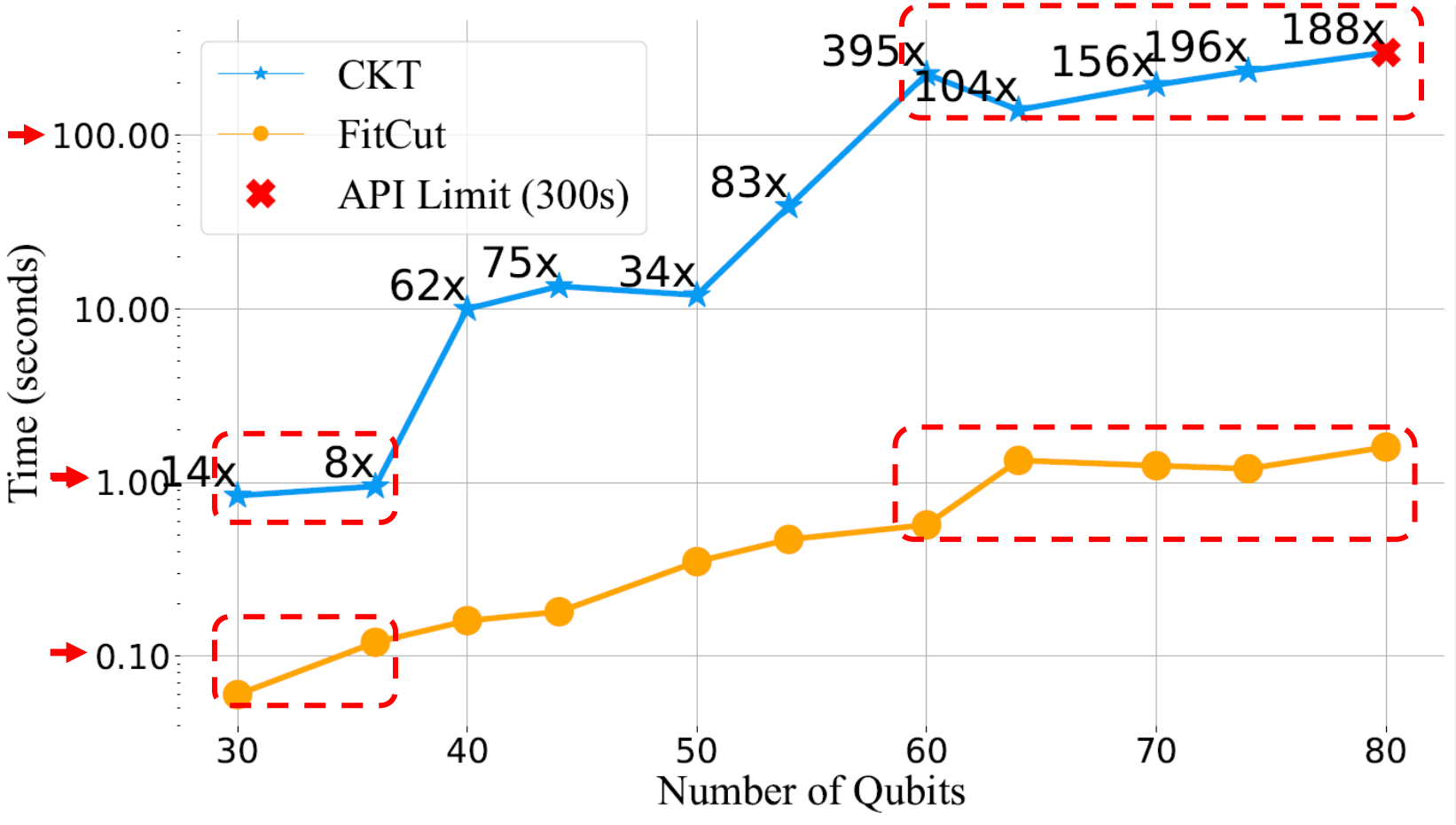
- Adder Circuit, 15-qubit worker



Number of Cuts		
Width	CKT	FitCut
20	2	2
30	4	4
40	6*	6
50	6	6
54	8*	8
60	NA*	8

Search Time and Number of Cuts Comparison

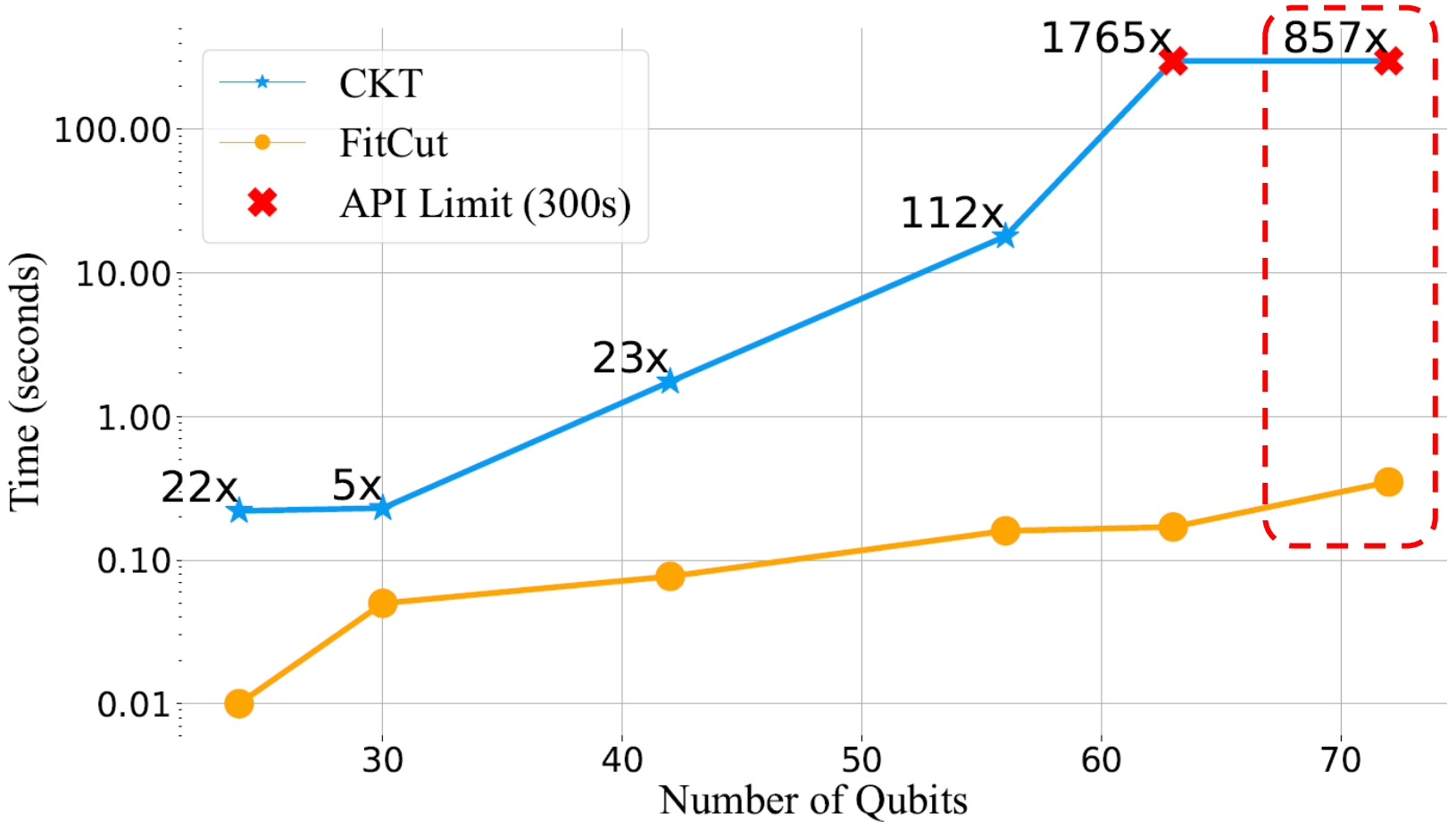
- Adder Circuit, 20-qubit worker



Number of Cuts		
Width	CKT	FitCut
30	2	2
40	4	4
50	4	4
60	6	6
70	6	6
80	NA*	8

Search Time and Number of Cuts Comparison

- Supremacy Circuit, 20-qubit worker



Number of Cuts		
Width	CKT	FitCut
24	4	[4,8]
30	5	[6,9]
42	10	[10,15]
56	15	[16,22]
64	20*	[20,26]
72	27*	[24,30]

Takeaways

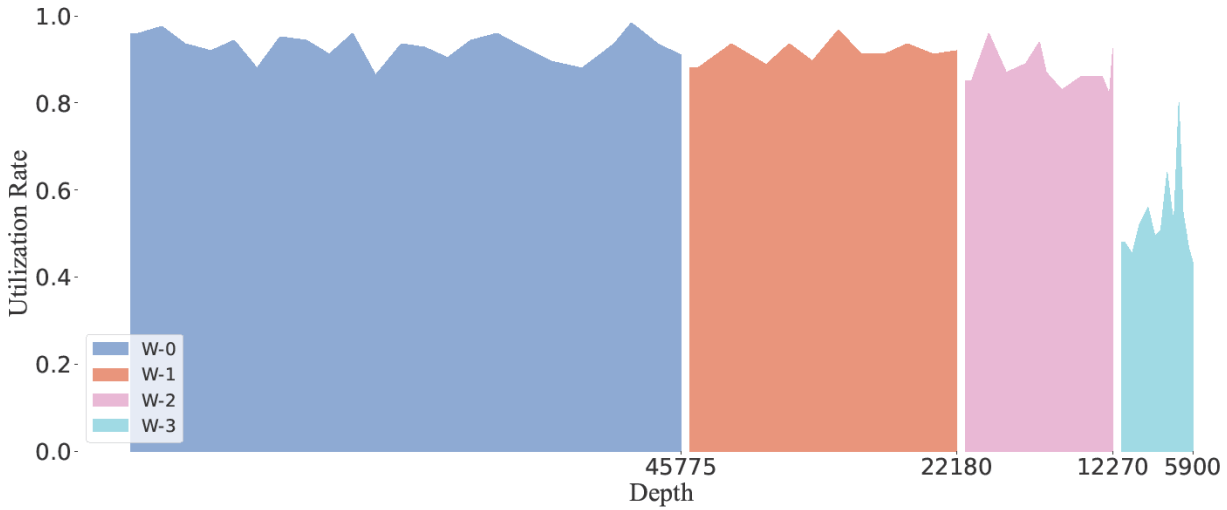
- Search time:
  - FitCut achieves **3**x to **2000**x speedup compared to CKT
  - Larger circuits experience more significant speedup.

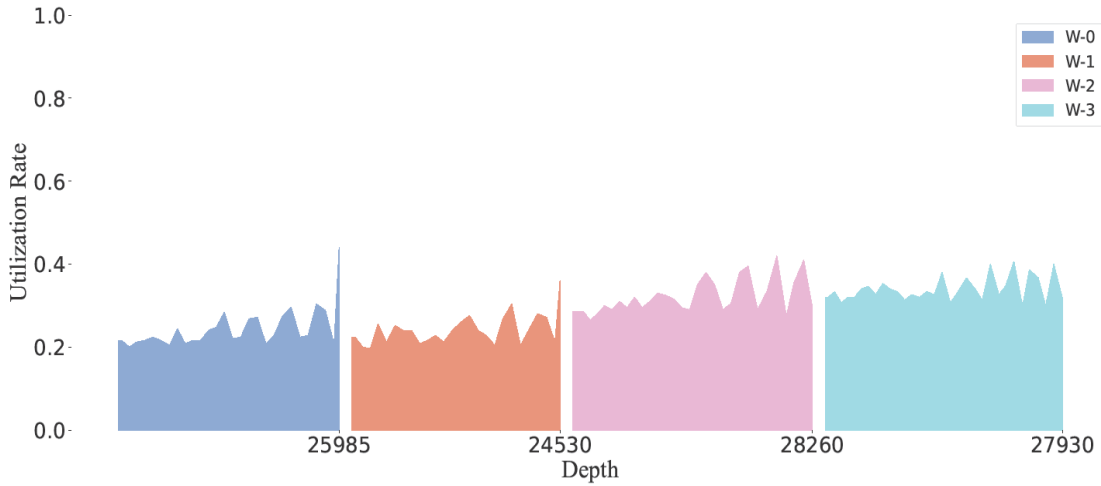
- Number of cuts:
  - For structured circuits (e.g. adder, BV, HWEA):
    - FitCut constantly finds optimal solution.
    - FitCut succeeds when CKT fails within 300s limit.
  - For random circuits (e.g. supremacy):
    - FitCut's results show more variability but still outperforms CKT in multiple trials.
    - Fitcut is able to find better result than CKT within 300s for larger circuit.

System-wide Resource Utilization Comparison

FitCut VS Modularity-only solution:

- Distributed system with 4 workers : [25-qubit,25-qubit,20-qubit,15-qubit]
- The Utilization rate:
  - FitCut: 0.93 vs Modularity-only: 0.32
- System-wide depth is reduced by 19.3%



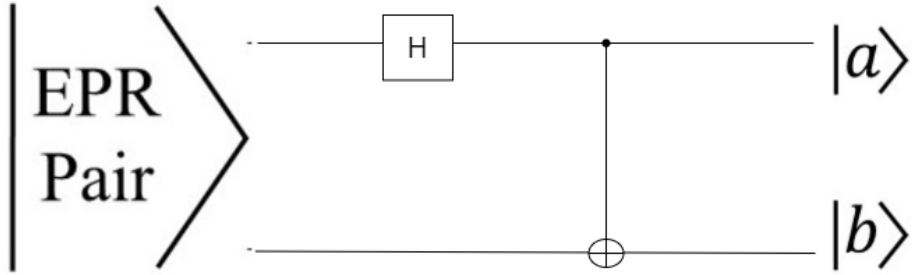


Outline

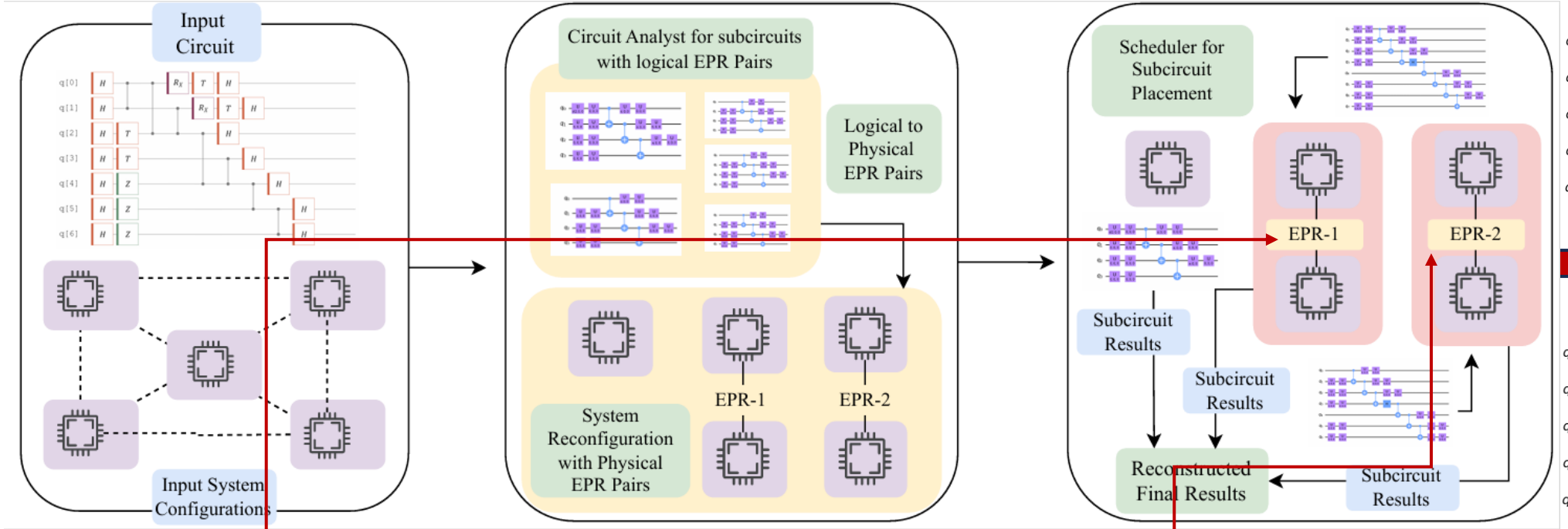
- Background
- Motivation
- Solution Design
- Evaluation
- **Future Work**

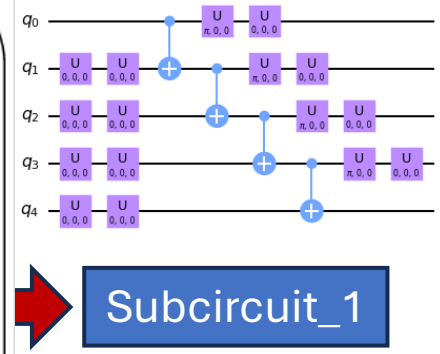
Einstein-Podolsky-Rosen (EPR) pair

- Entangled states(Bell states) of two qubits: are commonly used in quantum communication to enable remote gate operations in multi-node quantum systems.

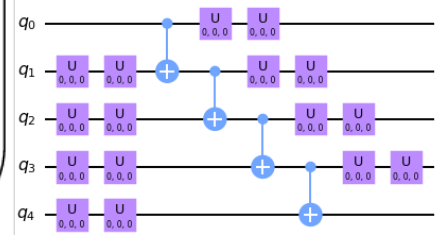


EC2S Multi-Node Quantum System:



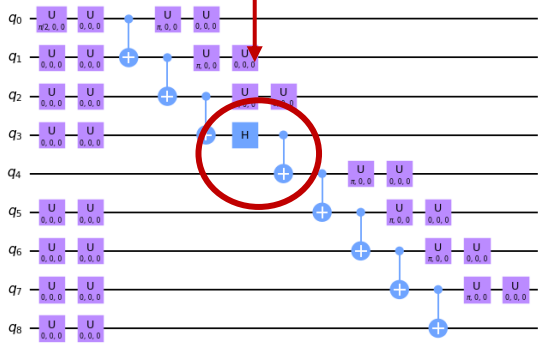


Subcircuit_1

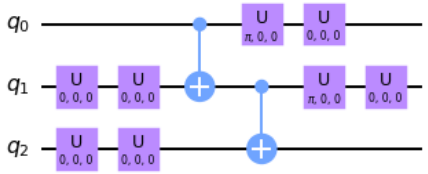


Subcircuit_2

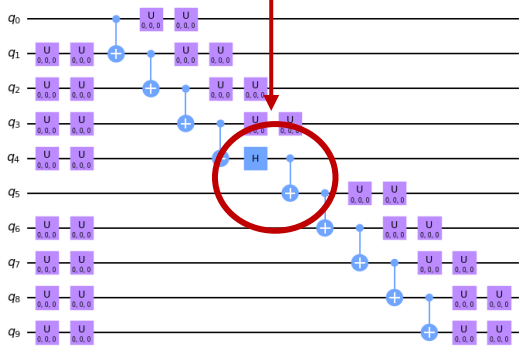
Subcircuit_3



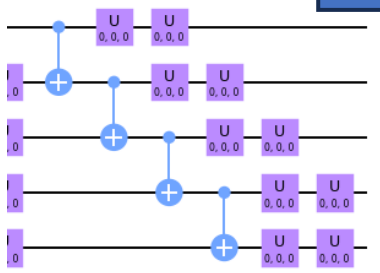
Subcircuit_0



Subcircuit_1



Subcircuit_2



Subcircuit_4

Multi-Node

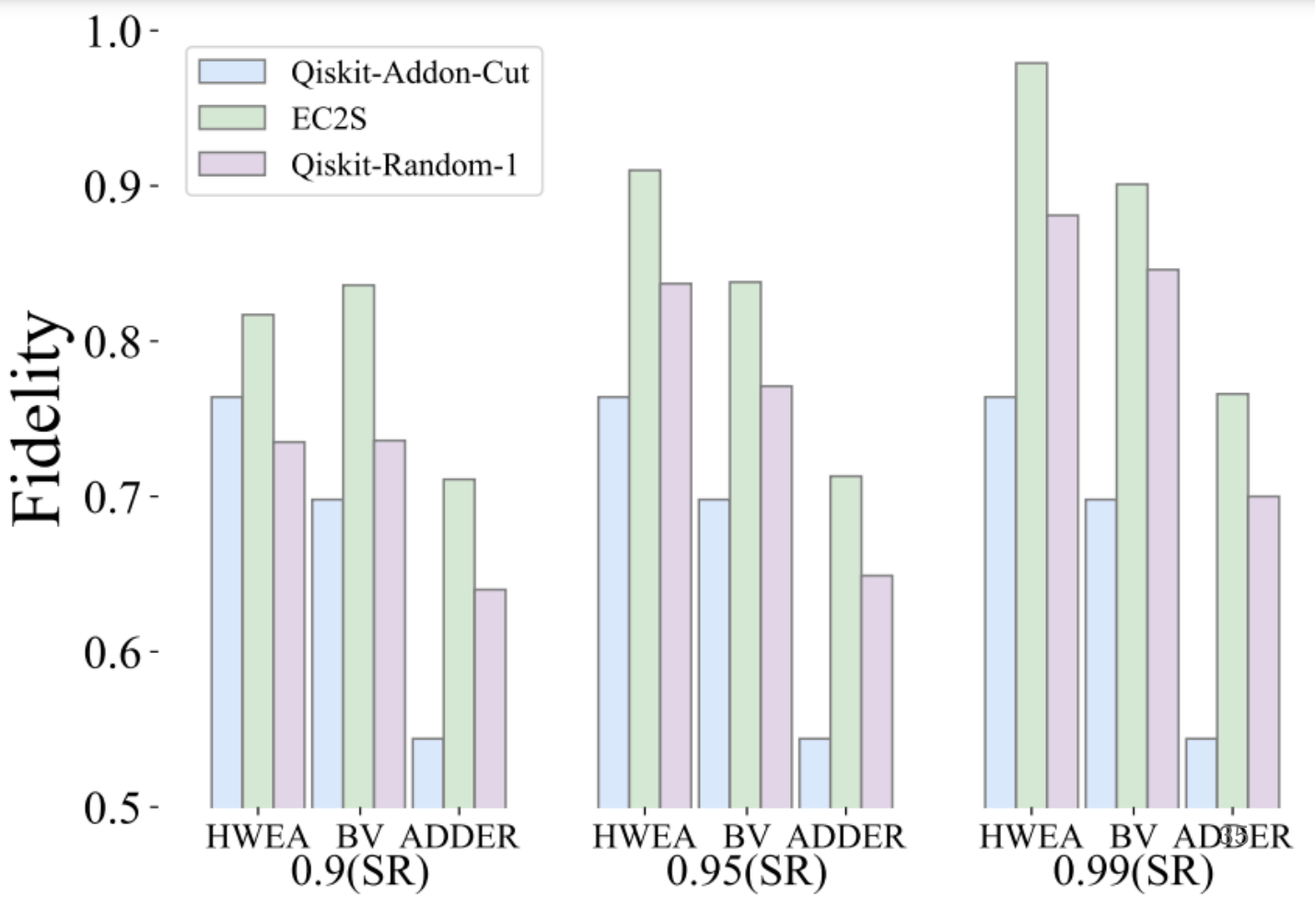
- 4-worker system randomly selecting emulator backends from a pool consisting of IBM Auckland, IBM Toronto, IBM Sydney, and IBM Montreal.

SR(Success rate) = 0.9:

- Fidelity improvements of 5.3%, 12.8%, and 16.7% for HWEA, BV, and ADDER

SR = 0.99:

- Fidelity improvements of 16.2%, 6.5%, and 5.5% for HWEA, BV, and ADDER.



Q & A
Thank you!