量子计算理论基础与软件系统

期末项目要求

基本要求

- 单人作业，不允许组队
- 顶会论文理解 + 代码复现 + 实验报告
- 支持自行选择论文，需确认是否符合要求
- 提交项目报告及关键代码至学在浙大，截止时间：2026年1月16日23:59
- 项目报告需包含：问题分析、算法原理、技术实现、实验测试、心得体会

推荐论文

■ 参考于 HPCA 2025 发布的 Janus 3.0 量子软件框架，整合了多篇顶会成果：

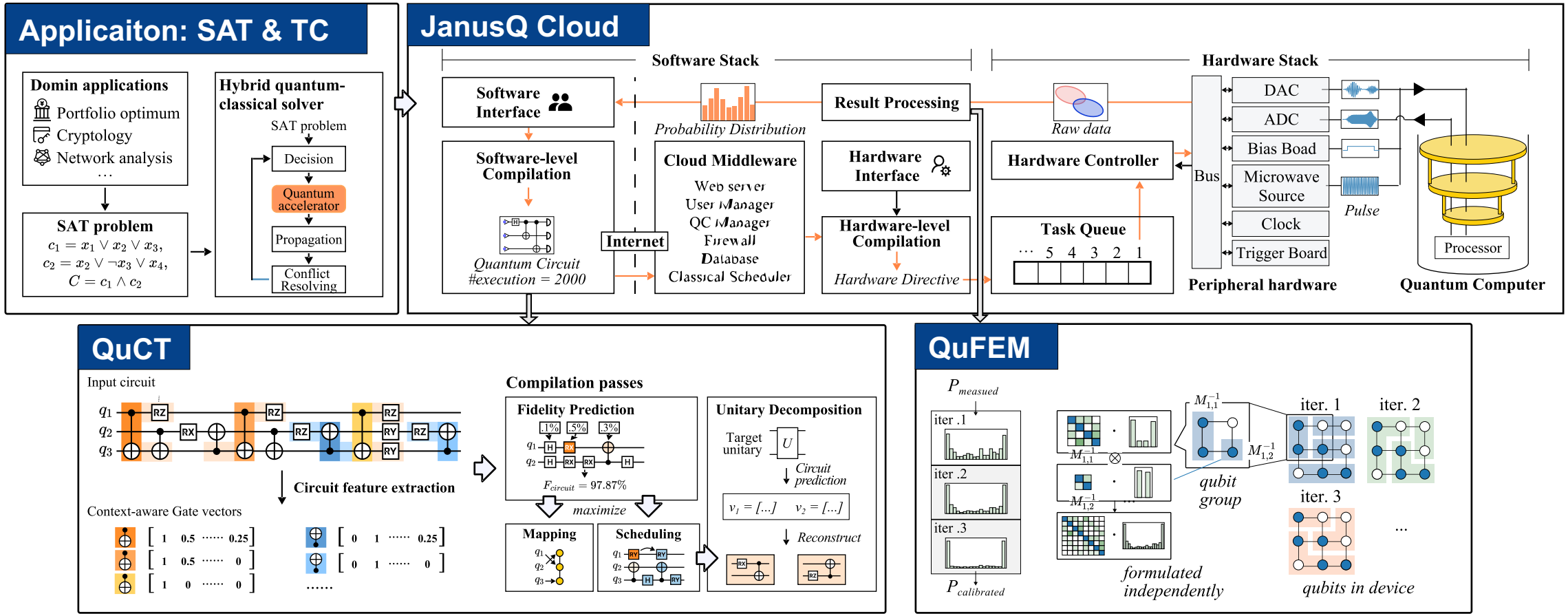
- [MICRO 2023] QuCT: A Framework for Analyzing Quantum Circuit by Extracting Contextual and Topological Features
- [ASPLOS 2024] MorphQPV: Exploiting Isomorphism in Quantum Programs to Facilitate Confident Verification
- [ASPLOS 2024] QuFEM: Fast and Accurate Quantum Readout Calibration Using the Finite Element Method
- [HPCA 2025] Choco-Q: Commute Hamiltonian-based QAOA for Constrained Binary Optimization

■ 相关链接：https://janusq.github.io/HPCA_2025_Tutorial/home

Janus (Taiyuan) Quantum Toolkit

ZHEJIANG UNIVERSITY

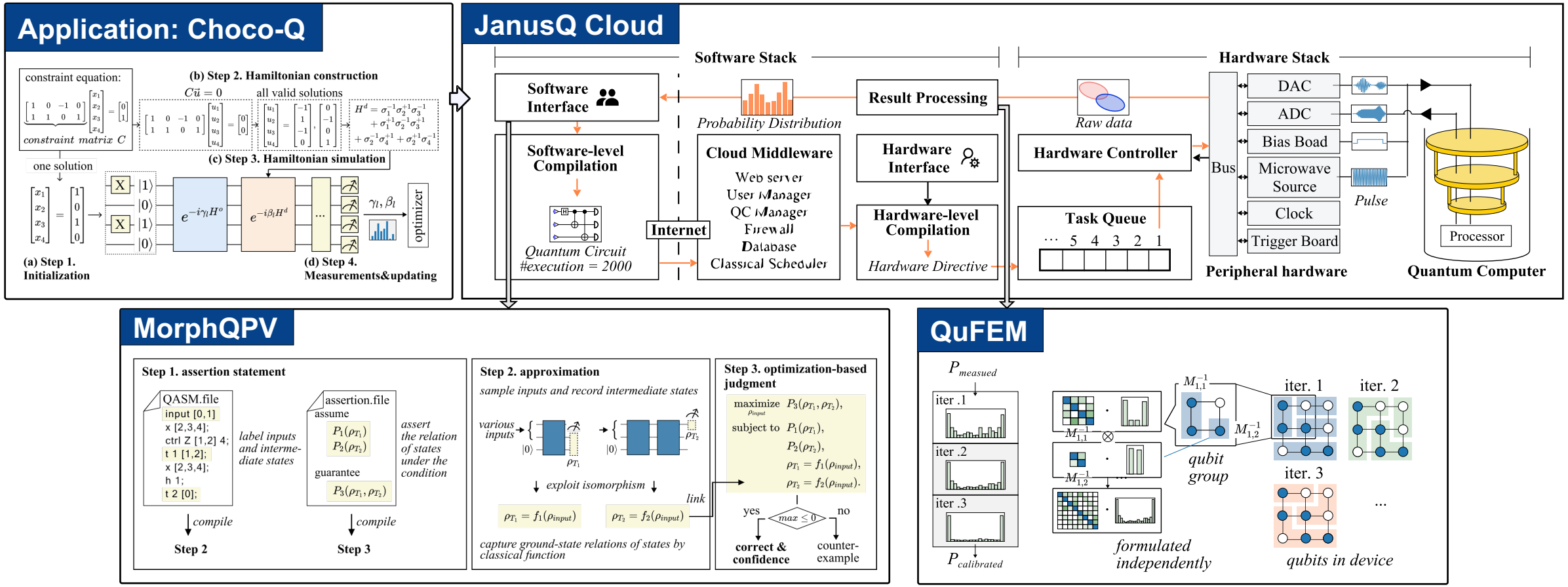
Janus 2.0: A Software Framework for Analyzing, Optimizing and Implementing Quantum Circuit



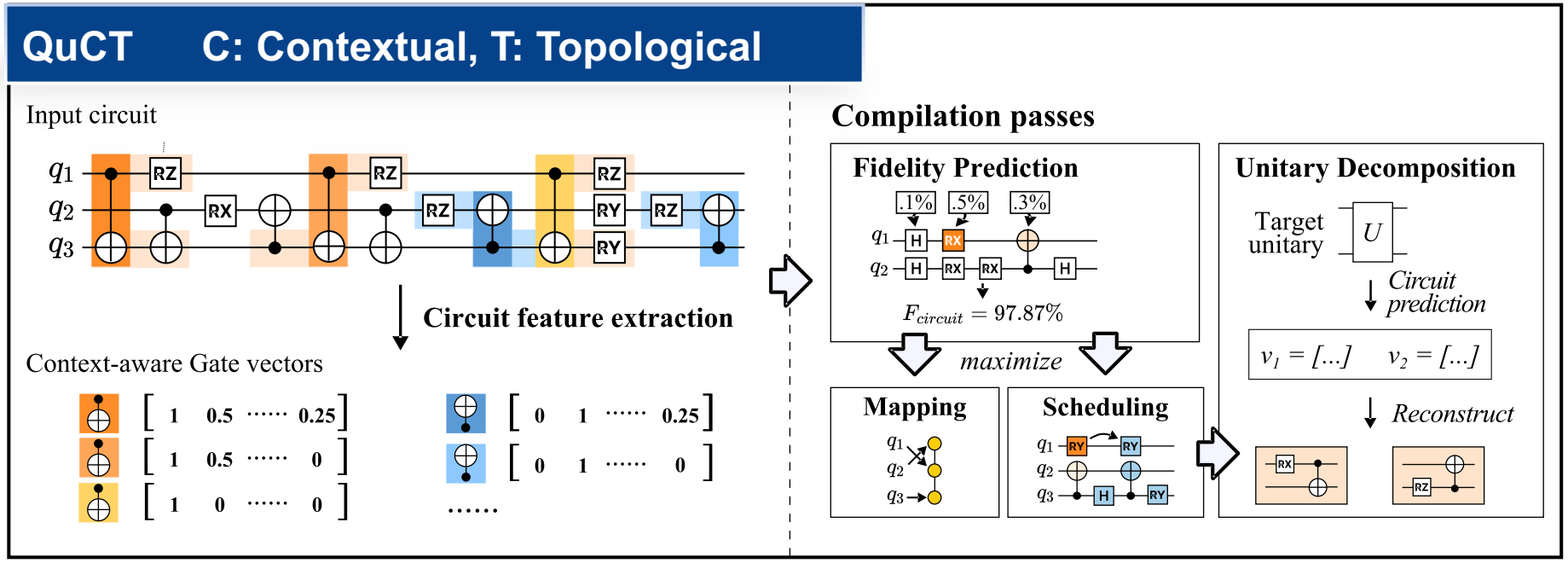
JanusQ 3.0 Overview

ZHEJIANG UNIVERSITY

Janus 3.0: A Software Framework for Analyzing, Optimizing, Verifying, and Implementing Quantum Circuit



QuCT

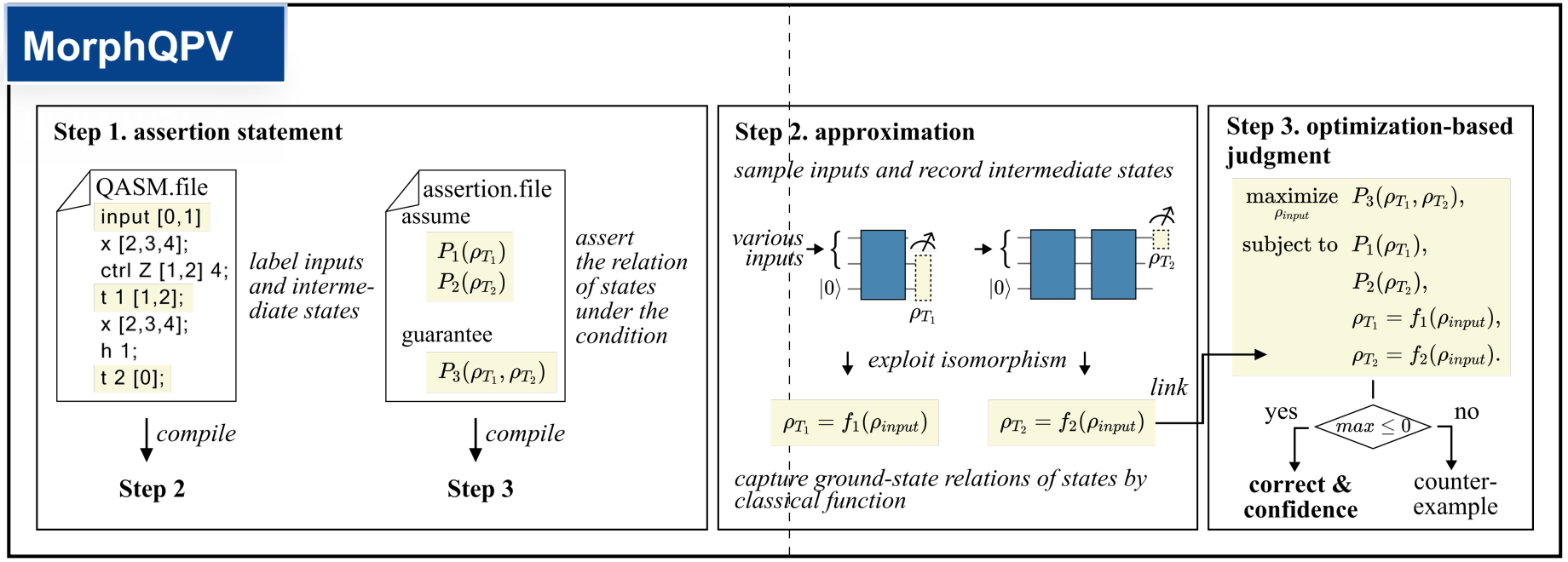


QuCT: Topology-aware quantum circuit optimizer (MICRO 2023)

- Why is topological information important in circuit analysis and optimization?
- How do we use topological information to analyze and optimize noise?
- How can we speed up the unitary decomposition with an upstream model?

Related paper: QuCT: A Framework for Analyzing Quantum Circuit by Extracting Contextual and Topological Features. **[MICRO 2023]**

MorphQPV

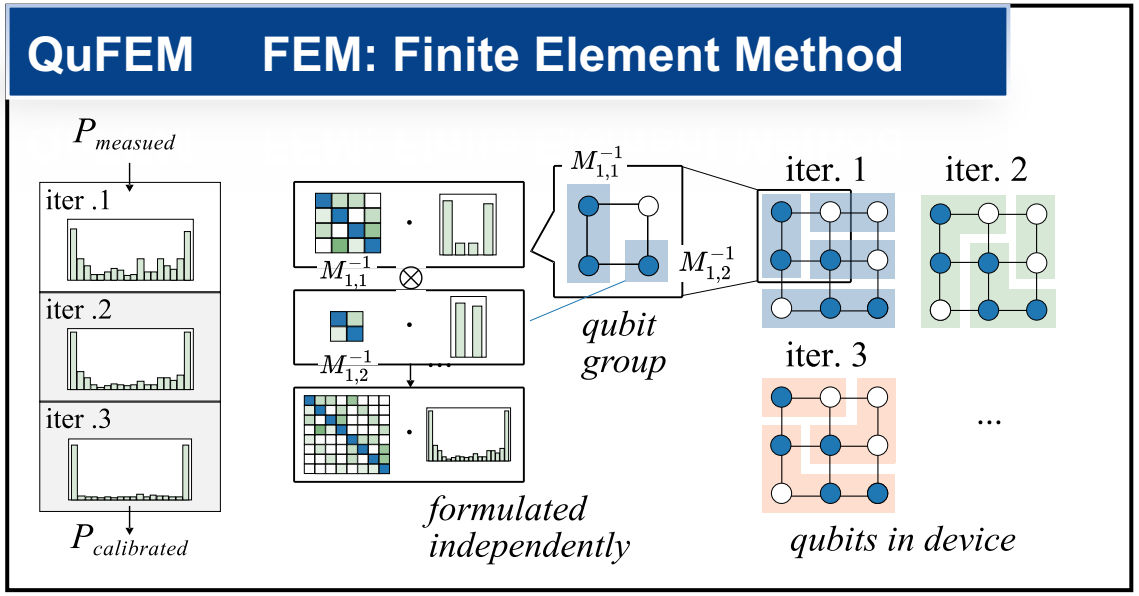


MorphQPV: Isomorphism-based quantum program verification (ASPLOS 2024)

- What is program verification in quantum circuit analysis? Why is it difficult?
- How do we use isomorphism relationship to verity quantum programs?
- How do we realize the assertion statement in quantum programs?

Related paper: MorphQPV: Exploiting Isomorphism in Quantum Programs to Facilitate Confident Verification. **[ASPLOS 2024]**

QuFEM

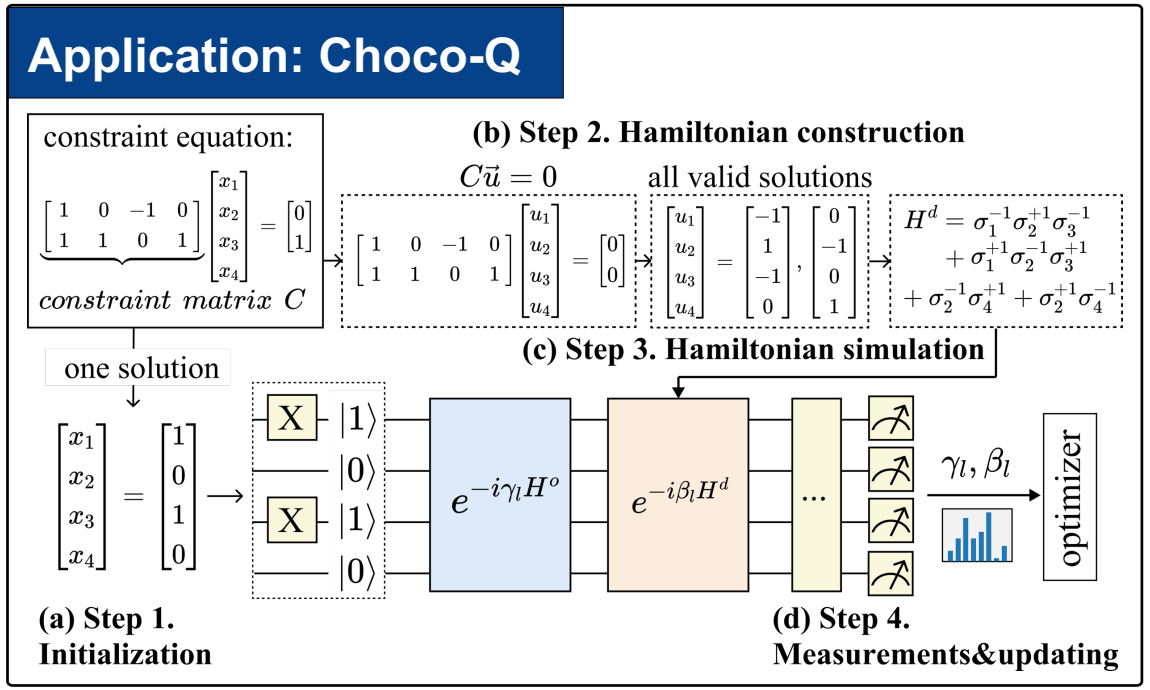


QuFEM: Readout Calibration based on finite element method (ASPLOS 2024)

- What is readout error and the difficulty of calibrating it?
- What is the finite element method?
- How do we mitigate the readout noise using the finite element method?

Related papers: QuFEM: Fast and Accurate Quantum Readout Calibration Using the Finite Element Method. **[ASPLOS 2024]**

Choco-Q



Choco-Q: Constrained Binary Optimization (HPCA 2025)

- What is constrained binary optimization?
- How do we use commute Hamiltonian to squeeze the searching space of QAOA?

Related papers:

Choco-Q: Commute Hamiltonian-based QAOA for Constrained Binary Optimization. **[HPCA 2025]**

Installing JanusQ

https://github.com/JanusQ/JanusQ

Installing Methods

Language: Python 3, C++

Available platforms: Linux, Mac, Windows

HyQSAT has not been tested on Windows.

Hardware requirements:

- Classical computer: ≥ 16 GB Memory, Intel i5 10 Gen

Hardware used in the experiment

- A server with two AMD EPYC 2.25GHz 64-core CPUs and 1.6TB DDR 5 memory is preferable.
- Superconducting, ion-trapped quantum computer and D-Wave quantum annealer.

Data collected from the quantum hardware are put into the framework.

Installation:

From Docker (recommend)

From Source code

Wheel is provided in https://github.com/JanusQ/JanusQ/tree/main/dist. But HyQSAT is disabled.

Install from Docker (Preferred)

Step 1. download docker-desktop

https://www.docker.com/products/docker-desktop/

Windows Subsystem for Linux (WSL) are required for windows.

Step 2. pull JanusQ image

```
docker pull janusq/janusq3:latest
```

Step 3. start image

```
docker run -itd -p 8888:22 -p 9999:23 --name tutorial janusq/janusq3
```

Step 4. Use SSH/VSCode/PyCharm to connect the docker

```
ssh root@localhost -p 8888
```

The password of the docker is "root".

Step 5. Or use Jupyter Lab to connect the docker

```
http://localhost:9999/lab
```

Docker page of JanusQ:
https://hub.docker.com/r/janusq/janusq3

Install from Source Code

Step 1. clone the source code

```
git clone git@github.com:JanusQ/JanusQ.git
```

Step 2. install requirements (virtual environment is preferred)

```
cd ./JanusQ
conda create -n janusq python=3.10
conda activate janusq
pip install -r requirements.txt
```

Step 3. compile HyQSAT

```
cd ./janusq/application/hyqsat/solver
cmake .
make install
```

Implemented based on C++.

Step 3. install requirements for Choco-Q (Linux with CPU)

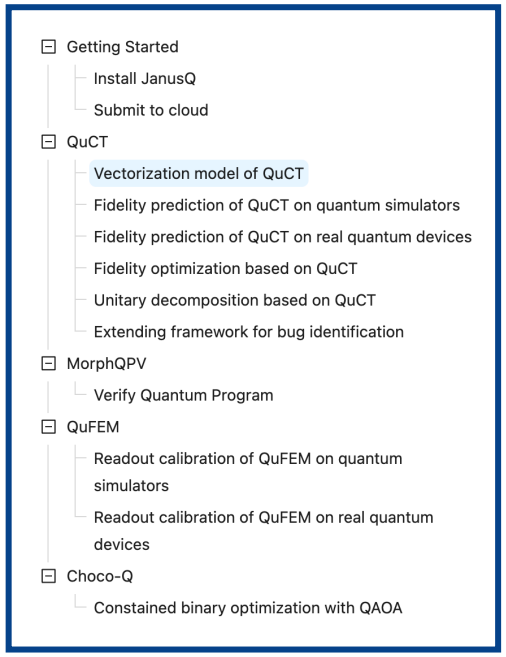
```
cd ./janusq/application/chocoq
conda env create -f environment_linux_cpu.yml
```

Change the dependency file according to your platform

Page of github:
https://github.com/JanusQ/JanusQ

Document and Example

On: **JanusQ/examples** or **https://janusq.github.io/HPCA_2025_Tutorial/demo**



Include 12 examples

Vectorization Model of QuCT

Author: Siwei Tan

Date: 7/4/2024

Based on paper "QuCT: A Framework for Analyzing Quantum Circuit by Extracting Contextual and Topological Features" (MICRO 2023)

In the current Noisy Intermediate-Scale Quantum era, quantum circuit analysis is an essential technique for designing high-performance quantum programs. Current analysis methods exhibit either accuracy limitations or high computational complexity for obtaining precise results. To reduce this tradeoff, we propose QuCT, a unified framework for extracting, analyzing, and optimizing quantum circuits. The main innovation of QuCT is to vectorize each gate with each element, quantitatively describing the degree of the interaction with neighboring gates. Extending from the vectorization model, we can develop multiple downstream models for fidelity prediction and unitary decomposition, etc. In this tutorial, we introduce the APIs of the vectorization model in QuCT.

In [1]:

```
%matplotlib inline

import os
os.chdir("../..")
import logging
logging.basicConfig(level=logging.WARN)

import random
import numpy as np

from janusq.analysis.vectorization import RandomwalkModel, extract_device
from janusq.objects.random_circuit import random_circuits, random_circuit
from janusq.objects.backend import GridBackend
```

Vectorization Flow

Below is the workflow to vectorize a gate in the quantum circuit. The gate is vectorized by two steps. The first step runs random walks to extract circuit features in the neighbor of the gates. the second step use a table comparison to generate the gate vector.