bachelor's Thesis (Academic Year 2025)

Structure-Aware Suitability Analysis of Quantum Circuit Partitioning for Distributed Quantum Computing

Keio University
Undergraduate School of Media and Governance

Yuta Imahoko

January 2026

bachelor's Thesis Abstract   Academic Year 2025

Structure-Aware Suitability Analysis of Quantum Circuit Partitioning for Distributed Quantum Computing

Summary

Quantum computing is scaling rapidly, and distributed quantum computing (DQC)—in which circuits that no longer fit on a single quantum processor are executed cooperatively across multiple nodes—is becoming increasingly important. For quantum circuit partitioning, a core problem in DQC, a variety of approaches have been proposed, including random partitioning, graph-cut minimization (e.g., the Kernighan–Lin algorithm), hypergraph-based methods, and frequency-based heuristics. However, a systematic framework for determining *which method is suitable for which circuit structures* prior to execution remains insufficiently developed, and many prior studies have largely been limited to empirical comparisons on individual benchmarks. As a result, it remains difficult to obtain generalizable insights that explain the relative advantages and disadvantages of partitioning methods in terms of circuit structure.

The goal of this study is to quantitatively clarify the relationship between structural characteristics of quantum circuits and partitioning performance, and to establish decision indicators for selecting partitioning strategies according to circuit structure. To this end, we first design a set of structural indicators that can be computed directly from a circuit, independent of any particular partitioning outcome, in order to characterize circuits from the perspectives of "partitionability" and "communication-cost profiles." Specifically, we formulate indicators including edge-duplication measures, hot-qubit concentration, and Laplacian spectral features, enabling circuits to be positioned in a structural space and providing a foundation for subsequent method selection and explanation.

Next, we apply multiple representative partitioning algorithms (e.g., Kernighan–Lin and FM) under a unified experimental setting to multiple benchmark circuits (e.g., QASMBench and QFT), and measure communication-cost metrics such as the number of remote CNOTs (RCX). We then organize the relationship between circuit-structure indicators and partitioning performance—especially relative differences across methods—using regression/classification frameworks together with explainability analysis (SHAP), and translate the results into an explainable form that answers: "under what conditions does method A tend to outperform method B?" Through this process, we aim to provide a performance map over the structural space that offers practical guidance for method selection.

The outcomes of this study provide a foundation for moving beyond ad hoc, experience-based method selection, toward selecting partitioning strategies based on circuit structure. Moreover, by deepening the understanding of how circuit structure relates to communication cost, the results are expected to contribute to circuit-design guidelines (structural guidelines) that are better suited to distributed quantum computing.

Keywords

quantum computer, quantum compiler, distributed quantum computer, quantum circuit paritioning

Undergraduate School of Media and Governance
Keio University

Yuta Imahoko

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

As quantum computation scales up, it becomes increasingly plausible that the circuit width and required resources will no longer fit within a single quantum processor. In such cases, Distributed Quantum Computing (DQC), where multiple nodes cooperate to execute a circuit, becomes important. However, in DQC, two-qubit operations that span across nodes inevitably involve communication and synchronization, and thus tend to dominate the execution cost. Therefore, *quantum circuit partitioning*—assigning a circuit to multiple nodes—is a core problem in DQC.

1.2 Problem Setting

For quantum circuit partitioning, many algorithms have been proposed, including graph partitioning (e.g., Kernighan–Lin), hypergraph partitioning, and frequency- or hotspot-based methods. On the other hand, a systematic framework for judging **which partitioning method is suitable for what kind of quantum circuit**, based on circuit structure, remains underdeveloped. As a result, many prior studies have been limited to empirical evaluations on individual benchmarks, and have not sufficiently provided generalized explanations of **why method A performs relatively better than method B on the same circuit**.

Moreover, in practical use, it is unavoidable that previously unseen quantum circuits will be given as inputs, yet there is a lack of **quantitative descriptors of circuits** and a corresponding **basis for method selection**. This gap makes it difficult not only for users of partitioners but also for those who design and edit circuits to make informed decisions such as "how much communication cost should be expected" and "how structural changes are likely to increase or decrease the cost." This study focuses on this gap.

1.3 Research Objective

The objective of this study is to quantitatively clarify the relationship between structural characteristics of quantum circuits and partitioning performance, and to establish decision indicators for selecting partitioning strategies according to circuit structure. Here, "partitioning performance" is evaluated mainly by metrics related to communication cost. In particular, this study centers on Remote CNOT as the evaluation metric, while positioning the possibility that the communication-cost model itself may change in the future as a topic for discussion.

1.4 Research Approach

This study proceeds in the following steps. First, we design a set of structure indicators that can be computed from a circuit and do not depend on partitioning results, so that circuits can be embedded into a "structural space." Second, we apply multiple partitioning algorithms to multiple benchmark circuits, and collect partitioning-performance data by measuring communication-cost metrics. Third, we analyze the relationship between structure indicators and partitioning performance using SHAP, and organize the results in an explainable manner—namely, under what conditions method A is likely to outperform method B.

1.5 Contributions

The contributions of this study are as follows.

- We systematically define a set of structure indicators for explaining circuit structure, enabling circuits to be described as points in a structural space.
- We compare results across multiple circuits and multiple methods within a single unified framework, and quantitatively characterize relative differences in partitioning performance.
- Based on the structure indicators, we derive interpretable rules (a performance map) for method selection, providing outlooks for unseen circuits.

1.6 Organization of This Thesis

Chapter **??** summarizes the prerequisites needed to formalize the DQC model, communication cost, and the circuit-partitioning problem. Chapter 3 reviews representative families of partitioning methods and positions this study within prior work. Chapter 4 defines the circuit-structure indicators and constructs the structural space considered in this study. Chapter 5 describes the benchmark circuits, partitioning algorithms, evaluation metrics, and experimental conditions. Chapter 6 presents and organizes the measured partitioning-performance results. Chapter **??** provides analysis and discussion based on the structure indicators, offering explanations for performance differences. Chapter **??** discusses general rules, limitations, and future directions. Finally, Chapter 7 concludes the thesis.

Chapter 2

Preliminaries

2.1 Quantum Computing

2.1.1 Quantum Bit (Qubit)

State Space of a Qubit

A single qubit is described by the two–dimensional complex Hilbert space $\mathcal{H}_2$ with the computational (Z) basis
$$\{|0\rangle\,,\ |1\rangle\}.$$
Any pure state $|\psi\rangle \in \mathcal{H}_2$ is a linear combination

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \qquad \alpha, \beta \in C,$$

subject to the normalisation condition

$$\langle\psi|\psi\rangle = |\alpha|^2 + |\beta|^2 = 1.$$

A global phase factor $e^{i\gamma}$ $(\gamma \in R)$ leaves physical predictions invariant, so $|\psi\rangle$ and $e^{i\gamma}\,|\psi\rangle$ represent the same physical state.

Vector Representation

Using column-vector notation,

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, \qquad |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}, \qquad |\psi\rangle = \begin{bmatrix}\alpha\\\beta\end{bmatrix}.$$

This representation facilitates matrix-based analyses of quantum gates and circuits.

Superposition Examples

Quantum parallelism originates from superposition. Two canonical examples are

$$|+\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big), \qquad |-\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big),$$

both satisfying the normalisation condition. The states $|+\rangle$ and $|-\rangle$ are eigenstates of the Pauli operator X with eigenvalues $+1$ and -1, respectively; the eigenstates of Z are $|0\rangle$ and $|1\rangle$.

Measurement Probabilities

Measuring $|\psi\rangle$ in the computational (Z) basis yields outcome 0 with probability

$$P(0) = |\alpha|^2,$$

and outcome 1 with probability

$$P(1) = |\beta|^2.$$

Post-measurement, the state collapses to the corresponding basis vector:

$$|\psi\rangle \longrightarrow \begin{cases} |0\rangle & \text{with probability } P(0), \\ |1\rangle & \text{with probability } P(1). \end{cases}$$

A qubit is thus a normalised vector in a two-dimensional complex Hilbert space. Dirac' s bra–ket notation provides a concise, linear-algebraic description of its state, while the squared modulus of each amplitude yields the measurement probabilities. Superposition and the resulting probabilistic behaviour underpin the computational power of quantum information processing.

2.1.2 Bloch Sphere

A single qubit state can be generally expressed as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \quad \alpha, \beta \in C, \quad |\alpha|^2 + |\beta|^2 = 1, \tag{2.1}$$

where $|0\rangle$ and $|1\rangle$ are the computational basis states.

This state can be conveniently visualized using the **Bloch sphere** representation (as shown in Fig. 2.1). Taking into account the normalisation condition and the irrelevance of the global phase, any pure single-qubit state can be written as

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle\,, \tag{2.2}$$

where $0 \leq \theta \leq \pi$ and $0 \leq \phi < 2\pi$. In this parameterization, θ is the polar angle from the north pole (the $+z$ axis) and ϕ is the azimuth (longitude) on the surface of a unit sphere.
.

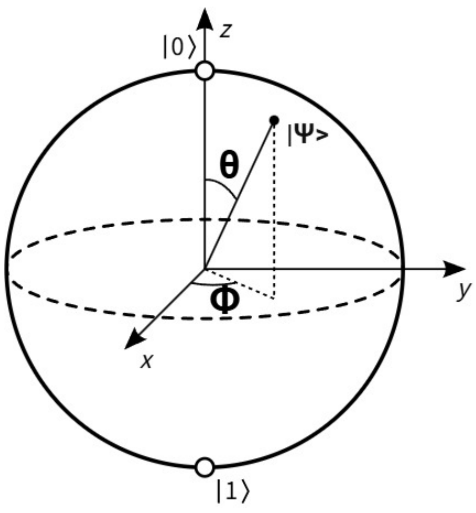


Figure 2.1: Bloch sphere representation of a single qubit.

Interpretation:

- $|0\rangle$ corresponds to the north pole ($\theta = 0$).
- $|1\rangle$ corresponds to the south pole ($\theta = \pi$).
- $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ is on the equator at $\phi = 0$.
- $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ is on the equator at $\phi = \pi$.
- $\frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle)$ is on the equator at $\phi = \frac{\pi}{2}$ (the positive y-axis).

Other commonly used bases are also easily represented:

- $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$: positive x-axis.
- $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$: negative x-axis.
- $|+i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle)$: positive y-axis.
- $|-i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i\,|1\rangle)$: negative y-axis.

The Bloch sphere not only visualizes qubit states but also helps understand quantum gate operations, which correspond to rotations of the state vector on the sphere. For example, the Pauli-X gate induces a π rotation about the x-axis, while the Pauli-Z gate induces a π rotation about the z-axis.

Measurement in quantum computing can be interpreted as projecting the qubit state onto an axis determined by the measurement basis on the Bloch sphere.

2.1.3 Multi-Qubit Systems

Tensor Product A composite quantum system of multiple qubits is described by the tensor product of the Hilbert spaces of each qubit. For two qubits, if the first qubit is in state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and the second in $|\phi\rangle = \gamma|0\rangle + \delta|1\rangle$, the joint state is

$$|\Psi\rangle = |\psi\rangle \otimes |\phi\rangle = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle, \tag{2.3}$$

where $|ab\rangle$ denotes $|a\rangle \otimes |b\rangle$.

The state space of an n-qubit system is 2^n-dimensional, i.e., $\mathcal{H}^{\otimes n}$. This exponential growth of the state space is a key quantum computational resource.

Entanglement Not all multi-qubit states can be written as tensor products of single-qubit states. A state that cannot be so written is called an **entangled state**. Formally, a two-qubit state $|\Psi\rangle$ is entangled if there do not exist single-qubit states $|\psi\rangle, |\phi\rangle$ such that $|\Psi\rangle = |\psi\rangle \otimes |\phi\rangle$.

Entanglement is a uniquely quantum phenomenon and is fundamental to quantum computing and quantum information science.

Bell States A canonical example of two-qubit entangled states are the Bell states. The four Bell states form an orthonormal basis for the two-qubit Hilbert space:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.4}$$

$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \tag{2.5}$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \tag{2.6}$$

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \tag{2.7}$$

These states are maximally entangled and exhibit perfect quantum correlations. For example, if two parties share $|\Phi^+\rangle$, measurement of one qubit in the computational basis instantly determines the other's outcome.

For instance, $|\Phi^+\rangle$ cannot be factorized as a product of two single-qubit states, illustrating true quantum entanglement.

2.1.4 Quantum Gates

Quantum gates are the fundamental operations in quantum circuits, analogous to classical logic gates. However, unlike classical gates, quantum gates are represented by **unitary matrices** and act on quantum states via linear, reversible transformations.

Single-Qubit Gates

Pauli-X Gate (NOT Gate) The Pauli-X gate flips the basis states:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad X|0\rangle = |1\rangle, \quad X|1\rangle = |0\rangle. \tag{2.8}$$

Pauli-Y Gate The Y gate combines bit-flip and phase-flip operations:

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Y|0\rangle = i|1\rangle, \quad Y|1\rangle = -i|0\rangle. \tag{2.9}$$

Pauli-Z Gate The Z gate applies a phase flip to the $|1\rangle$ state:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad Z|0\rangle = |0\rangle, \quad Z|1\rangle = -|1\rangle. \tag{2.10}$$

Hadamard Gate (H) The Hadamard gate creates a superposition from a basis state:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{2.11}$$

This gate is essential for enabling quantum parallelism and appears in many quantum algorithms, including the Quantum Fourier Transform and Grover's search.

Multi-Qubit Gates

CNOT Gate (Controlled-NOT) The CNOT gate flips the target qubit if the control qubit is $|1\rangle$. It is a two-qubit gate with the matrix representation:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.12}$$

Its action is:

$$|00\rangle \mapsto |00\rangle, \quad |01\rangle \mapsto |01\rangle, \quad |10\rangle \mapsto |11\rangle, \quad |11\rangle \mapsto |10\rangle$$

CNOT is indispensable for creating entangled states. For example:

$$|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \xrightarrow{\text{CNOT}} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

CZ Gate (Controlled-Z) The CZ gate flips the phase of the target qubit when the control is $|1\rangle$:

$$\text{CZ} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \tag{2.13}$$

Unlike CNOT, CZ is symmetric in control and target and is often used in entanglement generation and measurement basis transformations.

Unitary Nature of Quantum Gates

All quantum gates are unitary matrices U, satisfying:

$$U^\dagger U = UU^\dagger = I.$$

As a result, quantum operations are always reversible, unlike most classical logic gates.

2.1.5 Quantum Circuits

The quantum circuit model is the standard formalism for describing quantum computations. A quantum circuit consists of a sequence of quantum gates applied to qubits over discrete time steps, typically arranged from left to right.

A quantum circuit is composed of the following elements:

- **Qubit wires**: Horizontal lines represent individual qubits, with their quantum states evolving over time.
- **Quantum gates**: Unitary operators acting on one or more qubits, such as Hadamard, Pauli gates, and CNOT.
- **Measurement**: A final operation converting quantum states to classical bits via probabilistic projection.

Example Circuit

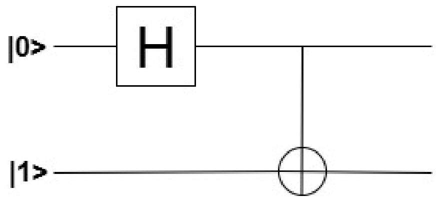


Figure 2.2: Bell state quatum circuit

This example shows a quantum circuit that generates a Bell state:

This corresponds to the following operations:

1. Apply the Hadamard gate to the first qubit to create a superposition.
2. Apply the CNOT gate to entangle the two qubits.

The resulting state is:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Temporal and Parallel Structure

Quantum circuits have a clear temporal ordering of gate operations:

- Gates acting on different qubits can be applied in parallel.
- The overall computation is a composition of unitary operations.

Mathematical Representation

The circuit's operation on an input state $|\psi_{\text{in}}\rangle$ can be expressed as:

$$|\psi_{\text{out}}\rangle = U_k \cdots U_2 U_1 |\psi_{\text{in}}\rangle,$$

where U_i are unitary matrices corresponding to the quantum gates in the circuit.

Significance and Applications

The quantum circuit model underlies the design of virtually all quantum algorithms, including Shor ' s algorithm, Grover ' s algorithm, and quantum error correction codes. It provides a clear visual and analytical framework and connects naturally to quantum hardware and compiler implementations.

Alternative Models of Quantum Computation

While the circuit model is the most widely used, other models of quantum computation also exist:

- **Measurement-Based Quantum Computing (MBQC)**: Uses a pre-prepared entangled resource state and performs computation via adaptive single-qubit measurements.

- **Topological Quantum Computing**: Encodes information in topological properties of anyonic quasiparticles, manipulating them via braiding.
- **Quantum Turing Machine**: A theoretical model analogous to the classical Turing machine; rarely used in practice but equivalent in computational power.

These models are computationally equivalent to the quantum circuit model but offer different trade-offs in terms of implementation and theoretical analysis.

2.1.6 Measurement

Quantum measurement is the fundamental mechanism through which classical information is extracted from a quantum system. Unlike unitary gate operations, measurement is inherently **probabilistic** and **irreversible**. It plays a critical role at the final stage of quantum computation, collapsing a quantum state into a classical bit value.

Basic Definition

Let a single-qubit state be

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle, \qquad \alpha, \beta \in C, \qquad |\alpha|^2 + |\beta|^2 = 1. \tag{2.14}$$

When measured in the computational (Z) basis $\{|0\rangle, |1\rangle\}$, the statistics and the post-measurement state are given formally below.

Projective Measurement Formalism

The measurement is described by the projectors

$$M_0 = |0\rangle\langle 0|, \qquad M_1 = |1\rangle\langle 1|, \qquad M_0 + M_1 = I. \tag{2.15}$$

The probability of outcome $m \in \{0, 1\}$ and the corresponding post-measurement state are

$$\Pr(m) = \langle\psi| M_m |\psi\rangle, \tag{2.16}$$

$$|\psi'\rangle = \frac{M_m |\psi\rangle}{\sqrt{\Pr(m)}}. \tag{2.17}$$

Equations (2.16)–(2.17) specialise to $\Pr(0) = |\alpha|^2$ and $\Pr(1) = |\beta|^2$, with $|\psi'\rangle = |0\rangle$ or $|1\rangle$, respectively.

Measurement in Other Bases

Measurement is not limited to the Z basis. For example,

X-basis: $\{|+\rangle, |-\rangle\}$, $\quad |\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$, $\qquad$ Y-basis: $\{|+i\rangle, |-i\rangle\}$, $\quad |\pm i\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm i |1\rangle)$.

To measure in an orthonormal basis $\{|u_0\rangle, |u_1\rangle\}$, apply a unitary U such that $U|0\rangle = |u_0\rangle$ and $U|1\rangle = |u_1\rangle$, then perform a Z-basis measurement:

$$\Pr(k) = |\langle u_k|\psi\rangle|^2 = \langle\psi| U |k\rangle\langle k| U^\dagger |\psi\rangle \quad (k = 0, 1). \tag{2.18}$$

Measurement in Multi-Qubit Systems

In an n-qubit system, partial (local) measurements are common. For the Bell state

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \tag{2.19}$$

measuring the first qubit in the Z basis yields outcome 0 (resp. 1) with probability $\frac{1}{2}$, after which the joint state collapses to $|00\rangle$ (resp. $|11\rangle$); the unmeasured qubit is thereby determined. This exemplifies entanglement correlations and the non-local constraints revealed by local measurements.

Irreversibility and Classical Information Extraction

Measurement is inherently irreversible: once an outcome is obtained, the superposition is destroyed and replaced by a classical bit value, and the pre-measurement amplitudes cannot be recovered from the post-measurement state alone. This aligns with the *no-cloning theorem* and the general non-unitary nature of measurement.

References: M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (10th Anniversary Ed.), Cambridge Univ. Press, 2010. Y. Shimada, *Quantum Computing: From Basic Algorithms to Quantum Machine Learning*, Kodansha, 2021.

2.2 Fault-Tolerant Quantum Computation (FTQC)

Motivation

Quantum systems are inherently fragile. Physical qubits are susceptible to decoherence, imperfect gate operations, *crosstalk*, and measurement errors. Unlike classical bits, quantum information cannot be duplicated to fight noise by naive redundancy (no-cloning), making it essential to detect and correct errors *during* computation within a systematic framework.[2, 20]

Definition of FTQC

Fault-tolerant quantum computation (FTQC) is a paradigm in which quantum algorithms are executed reliably in the presence of noise by combining quantum error-correcting (QEC) codes with fault-tolerant circuit constructions. Informally, FTQC ensures that local errors arising from gates, measurements, or idling are detected and corrected without spreading uncontrollably across the device.[10] This is achieved by encoding each logical qubit into many physical qubits and by using gate implementations (e.g., transversal gates, ancilla-assisted gadgets such as lattice surgery or magic-state distillation) that confine error propagation to correctable patterns.

Objectives of FTQC

1. Suppress logical error rates The logical error rate per operation, p_{L}, must be driven far below the physical error rate p_{phys}. With surface-code families, increasing the code distance can push p_{L} to extremely low levels (e.g., $\sim 10^{-12}$), depending on distance, decoder, and architectural overhead.[10] A practical design target is

$$p_{\mathrm{L}} \quad \frac{1}{N_{\mathrm{ops}}},$$

so that a computation with N_{ops} logical operations succeeds with high probability.

2. Contain errors locally A single physical fault (e.g., a bad CNOT) should not induce uncorrectable, correlated errors on multiple logical qubits. Fault-tolerant gate constructions are engineered to limit error spread, typically via transversal structures, verified ancillae, or surgery-based protocols, so that any resulting faults remain within the code ’ s correctable set.

3. Exploit the threshold theorem The *quantum accuracy threshold theorem* guarantees scalability when p_{phys} lies below a code- and architecture-dependent threshold, often quoted in the range 10^{-2}–10^{-3} for leading codes and decoders.[2, 10] Below threshold, increasing code distance reduces p_{L} exponentially (up to prefactors), enabling arbitrarily long computations at the cost of additional space–time resources.

2.3 Quantum Error Correction Codes

Logical Qubit Construction: Code Distance and Qubit Overhead

The performance and resource requirements of surface codes are primarily characterized by a parameter called the **code distance**, denoted as d.

Code Distance d The code distance d is defined as the minimum number of physical qubit errors required to cause a logical error. It determines the error-correcting power of the code: up to $\lfloor (d-1)/2 \rfloor$ errors can be corrected reliably.

Physical Qubit Overhead To encode a single logical qubit, the surface code requires approximately $O(d^2)$ physical qubits[10]. For instance, a distance-5 surface code might require 25–49 physical qubits, depending on the layout and inclusion of ancilla qubits for stabilizer measurement.

As d increases, the logical error rate decreases exponentially, following an approximate scaling law of the form:

$$\epsilon_L \sim \alpha \left(\frac{\epsilon_P}{\epsilon_{\text{th}}} \right)^{(d+1)/2}$$

where ϵ_L is the logical error rate, ϵ_P is the physical error rate, ϵ_{th} is the threshold, and α is a constant depending on the circuit architecture.

2.3.1 Surface Codes

Quantum error correction (QEC) is a fundamental requirement for reliable quantum computation. Among various QEC codes, the surface code has emerged as a highly practical and robust scheme. It leverages a two-dimensional array of physical qubits arranged on a lattice, where each qubit interacts only with its immediate neighbors[10]. This local connectivity aligns well with current hardware constraints and enables comparatively high error thresholds.

The surface code encodes logical qubits within the stabilizer formalism on the lattice. As illustrated in Fig. 2.3, each logical qubit occupies a distinct planar *patch*. Data qubits reside on the edges of the lattice tiles (filled circles), while ancillary *syndrome* qubits (open circles) are associated with plaquettes. Blue and red plaquettes correspond to X- and Z-type stabilizers, respectively. Under periodic stabilizer measurements, Z-type stabilizers detect *bit-flip* (X) errors, and X-type stabilizers detect *phase-flip* (Z) errors.

The ability of the surface code to correct errors is parameterised by the *code distance* d. A larger d increases fault tolerance (longer minimum-weight logical operators), at the cost of additional qubits and cycles. In particular, a distance-d code can correct up to $\lfloor (d-1)/2 \rfloor$ arbitrary single-qubit errors. For a concrete illustration, Fig. 2.3 shows a distance-3 planar patch; such patches require $O(d^2)$ data and $O(d^2)$ ancilla qubits in total.

A key advantage of the surface code lies in its modular variants, such as patch-based, defect-based, and twist-based architectures. Here we focus on the patch-based approach due to its compatibility with tile-based layouts and simplified control.

In Fig. 2.4, we depict representative syndrome-extraction circuits for X- and Z-type stabilizers. Each round entangles data qubits with a neighbouring syndrome qubit using CNOT gates, followed by the measurement of the syndrome qubit. The Z-type stabilizer circuit (bottom) directly measures Z-parity to detect X errors. The X-type stabilizer circuit (top) is implemented via a basis change (Hadamards) to convert an X-parity check into a Z-parity measurement on the ancilla, thereby detecting Z errors. The resulting syndrome outcomes identify and localise physical faults without disturbing the encoded logical information.

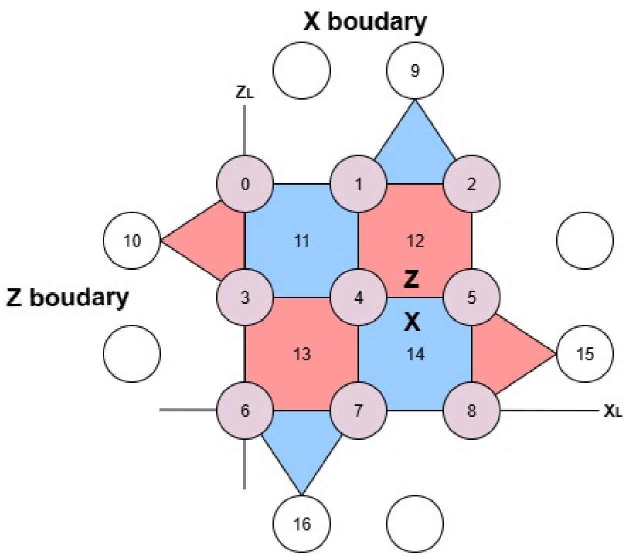


Figure 2.3: Example of a distance-3 patch-based surface code.

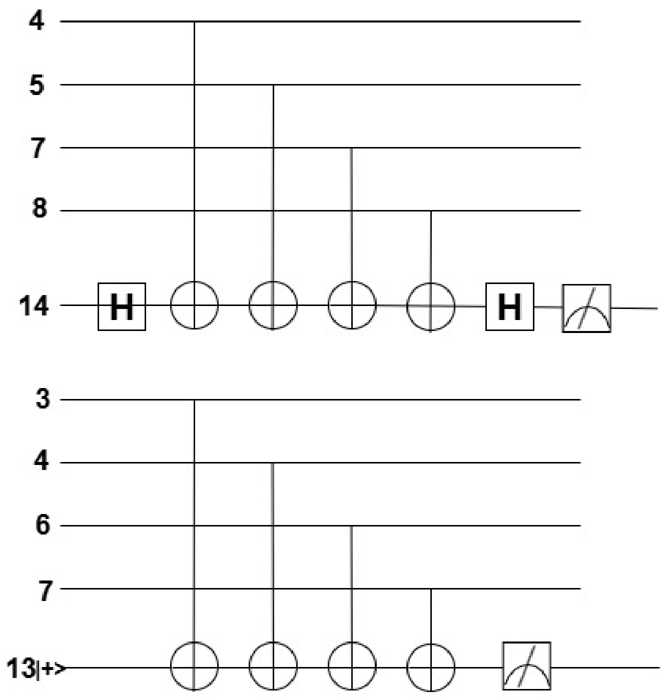


Figure 2.4: Syndrome extraction circuits: X (top) and Z (bottom) stabilizer measurements.

subsectionLattice Surgery

Lattice surgery is a central technique for implementing non-local logical operations in fault-tolerant quantum computation based on the surface code[15]. Unlike conventional gate-based approaches, lattice surgery realises interactions by *merging* and *splitting* neighbouring patch boundaries to measure joint logical parities (e.g., $Z_L \otimes Z_L$ or $X_L \otimes X_L$) while preserving strictly local connectivity within a tile-based surface-code architecture.

Logical CNOT via Lattice Surgery

A typical application of lattice surgery is the realization of a logical CNOT gate between two logical qubits, known as the control and target. A standard construction uses an ancilla patch initialised in $|+\rangle_L$ and performs two parity checks: first a rough merge to measure $Z_L^{(C)} Z_L^{(A)}$, then a smooth merge to measure $X_L^{(A)} X_L^{(T)}$. The classical outcomes determine Pauli-frame updates, yielding a net operation equivalent to $\mathrm{CNOT}_{C \to T}$. A high-level sequence is:

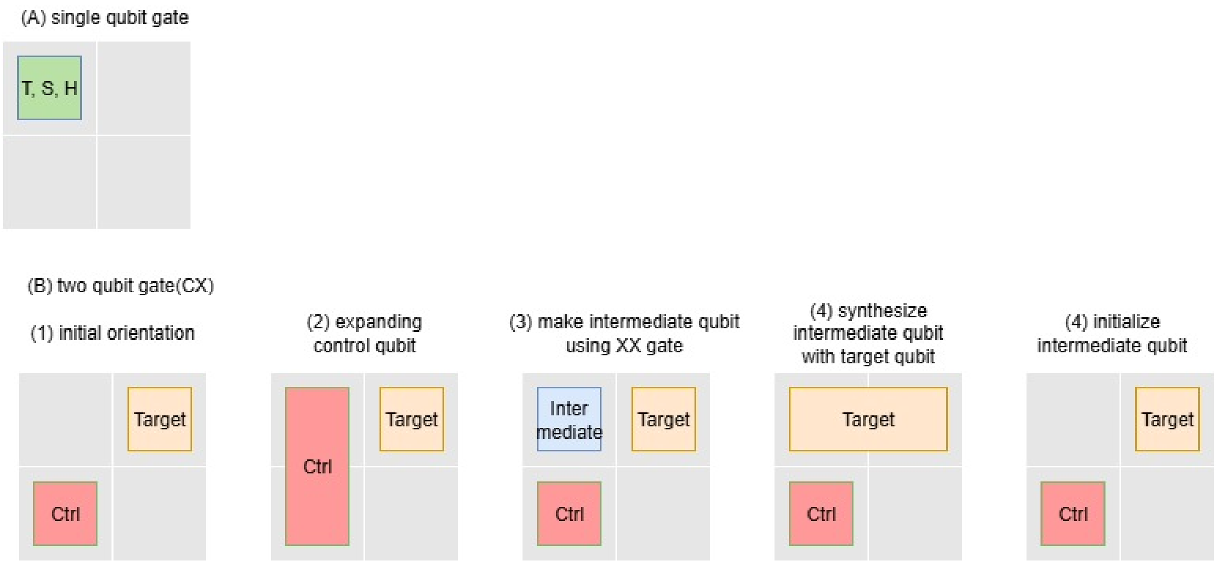

(A) single qubit gate
T, S, H
(B) two qubit gate(CX)
(1) initial orientation
Target
Ctrl
(2) expanding control qubit
Ctrl
Target
(3) make intermediate qubit using XX gate
Inter mediate
Target
Ctrl
(4) synthesize intermediate qubit with target qubit
Target
Ctrl
(4) initialize intermediate qubit
Target
Ctrl

Figure 2.5: Lattice surgery sequence for implementing a logical two-qubit (CNOT) gate using patch manipulations.

1. Place an ancilla patch between the control (C) and target (T) and initialise it in $|+\rangle_L$.
2. Perform a *rough merge* between C and the ancilla to measure the parity $Z_L^{(C)} Z_L^{(A)}$, then split to restore separate patches.
3. Perform a *smooth merge* between the ancilla and T to measure the parity $X_L^{(A)} X_L^{(T)}$, then split.
4. Measure the ancilla (e.g., in Z_L) and record both parity outcomes.

5. Apply the corresponding Pauli-frame updates to C/T; the composite effect is a logical CNOT from C to T.

This procedure enables fault-tolerant interaction between distant logical qubits while maintaining locality; each merge/split consumes $O(d)$ stabiliser-measurement rounds for code distance d, so the overall depth scales as $O(d)$ without requiring transversal two-qubit gates.

Scheduling Ancilla Patches and Error Management

The core resource in lattice surgery is the ancilla patch. In systems involving concurrent logical operations, the spatial and temporal scheduling of ancilla patches must avoid collisions[15]. Each ancilla follows a cycle of initialization, interaction, and measurement, necessitating pipeline-aware scheduling and retry logic for error recovery.

Error correction within lattice surgery is governed by the code distance d of the surface code. A code of distance d can correct up to $\lfloor (d-1)/2 \rfloor$ errors, which imposes a trade-off between resource overhead and logical fidelity. Furthermore, each measurement step in the surgery protocol must be synchronized with stabilizer measurements, which also require specific timing and physical gate sequences.

2.4 Distributed Quantum Computing (DQC)

Distributed Quantum Computing (DQC): A Scalable Alternative

Distributed quantum computing (DQC) offers an alternative paradigm: instead of building a single massive quantum processor, computation is distributed across multiple smaller quantum nodes that are interconnected via quantum and classical communication links.[2]

This architecture brings several potential advantages:

- **Modularity and fabrication scalability**: Each node can be optimized and fabricated independently.
- **Resource locality**: Some quantum operations can be performed entirely within a node, reducing noise.
- **Physical separation**: Enables heterogeneous platforms and separates noise sources.

However, DQC introduces new architectural and algorithmic challenges, primarily associated with **non-local quantum operations** across nodes.

Communication Cost and Remote CNOT

The core challenge in DQC lies in executing quantum gates between qubits that reside on different physical nodes. The most fundamental example is the **remote CNOT gate**, which must act on two qubits located at distinct sites.[20]

Implementing remote CNOT requires:

- Generation of entanglement (e.g., Bell pairs or EPR pairs) between remote nodes
- Classical communication for measurement-based correction
- Synchronization and buffering to manage decoherence during communication

Each remote CNOT typically incurs non-negligible time and fidelity cost. Therefore, the **total number of remote CNOTs** is a key metric in evaluating the performance of distributed quantum circuits.

Quantum Links Quantum communication enables the distribution of entanglement between remote nodes. This is achieved through the generation and transmission of entangled qubit pairs (e.g., Bell pairs), which serve as the foundation for non-local quantum operations.

However, quantum links face significant physical limitations:

- **High loss rates**: Photonic qubits transmitted through optical fiber suffer from attenuation.
- **Low entanglement generation rate**: Probabilistic nature of entanglement protocols leads to delays.
- **Short coherence time**: Qubits may decohere before communication completes.
- **Hardware complexity**: Requires quantum memories, optical interfaces, and possibly quantum repeaters.

These constraints make quantum communication a major bottleneck in DQC, and motivate architectural designs that minimize inter-node quantum interactions[20].

2.4.1 Non-Local Quantum Operations

A central challenge in distributed quantum computing is the implementation of quantum gates between qubits that reside on separate physical nodes. Among these, the remote controlled-NOT (CNOT) gate is particularly critical, as it underpins the construction of entangled states and the execution of universal quantum logic across distributed systems[33].

Several protocols have been proposed to realize remote CNOT operations, each offering different trade-offs in terms of communication cost, latency, and hardware requirements. This section presents three representative approaches[2, 5].

Gate Teleportation-Based Remote CNOT

Gate teleportation provides a method for implementing a remote CNOT without transferring the qubits themselves. Instead, the operation is effectively ”teleported” using pre-shared entanglement and classical communication.

The general idea is as follows:

- A Bell pair (EPR pair) is pre-generated and shared between the two nodes.

- The sender performs a Bell-state measurement involving the control qubit and one half of the EPR pair.
- The measurement result is sent to the receiver via a classical channel.
- The receiver applies Pauli corrections on the target qubit based on the received result.

This technique has the advantage of separating the non-unitary, noisy communication phase from the coherent quantum gate logic. It supports asynchronous operation and is well-suited to modular architectures, though it requires high-fidelity entanglement and precise timing for correction steps.

Quantum Teleportation-Based Remote CNOT

An alternative approach is to use quantum teleportation to move one or both qubits to the same physical node, where a local CNOT gate can then be applied.

The typical steps are as follows.

- Teleport the control or target qubit from its original node to the node where the other resides.
- Perform the CNOT operation locally.
- Optionally, teleport the moved qubit back to its original location.

This method is conceptually straightforward and compatible with existing teleportation protocols. However, it requires multiple teleportation steps and hence incurs additional communication and correction overheads. Moreover, relocating qubit states can disrupt data locality and complicate scheduling.

Data Qubit Swapping-Based Remote CNOT

A third strategy involves physically relocating the relevant qubits through a series of SWAP operations so that both qubits end up on the same node. The CNOT gate is then performed locally, and the qubits are returned to their original locations if necessary.

Key properties of this method include the following.

- Avoids the need for shared entanglement by relying only on unitary operations.
- It is compatible with static routing schemes and circuit-level scheduling.
- However, it may require a large number of SWAP operations if the nodes are far apart, increasing both the circuit depth and the accumulation of errors.

2.4.2 Quantum Compiler Overview

The quantum software stack mirrors classical computing in its hierarchical transformation from high-level algorithms to hardware-executable instructions. Following the model in [19], the compiler is structured as a layered flow, where each stage performs key translation and optimisation tasks.

Figure 2.6 presents the design flow used in this thesis, tracing how a quantum program progresses through multiple compilation layers to reach physical execution.

A **Quantum Program**, written in a high-level framework such as Qiskit or Scaffold, first enters the **Front-End**, which converts it into a technology-independent intermediate representation (e.g., QIR). This IR preserves logical structure while enabling formal analyses and program transformations.

The **Technology-Independent Optimiser** then applies hardware-agnostic rewrites —such as gate cancellation, commutation-based reordering, and circuit-depth reduction— many adapted from classical compiler techniques.

The program is subsequently lowered to QASM. The **Technology-Dependent Optimiser** then adapts the circuit to a concrete device model (connectivity, native-gate set, calibrated error rates). This stage performs qubit mapping (logical-to-physical assignment), insertion of routing operations (e.g., SWAP synthesis), and gate decomposition into native operations for the target platform (e.g., superconducting or ion-trap systems).

The final output—at a pulse/control level (e.g., a QPOL-style physical-operations layer)—is provided to a simulator or real hardware backend for execution using calibrated control sequences.

This architecture highlights the compiler ’ s dual role as translator and coordinator across abstraction layers. Although fault tolerance is not yet addressed in this NISQ-oriented flow, the next section extends it toward FTQC support and considers distributed systems.

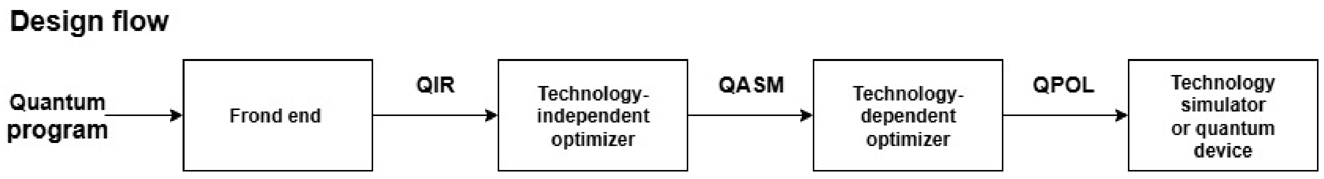

Design flow
Quantum program
Frond end
QIR
Technology-independent optimizer
QASM
Technology-dependent optimizer
QPOL
Technology simulator or quantum device

Figure 2.6: The design flow for quantum layered software architecture

2.4.3 Taxonomy of Compiler Tasks in FT-DQC

Design Objective. This taxonomy defines the functional decomposition of compilers that must deliver not only correctness and performance, but also fault tolerance in distributed quantum environments. Each category corresponds to a subsystem that preserves logical correctness under device noise, inter-node latency, and error-prone communication primitives.

Fault-tolerant distributed quantum computation (FT-DQC) introduces substantial complexity beyond NISQ-era compilation. In contrast to near-term flows that largely target depth reduction and hardware compliance, FT-DQC compilers must explicitly incorporate error-correction logic, manage fragile entangled resources across networked nodes, and ensure conformance to fault-tolerant protocols such as surface codes, magic-state distillation, and lattice surgery.

Thus, the compiler becomes a *correctness enforcer* across abstraction levels: every transformation, mapping, and scheduling decision must account for logical fidelity under realistic noise and communication constraints.

To address these challenges, we adopt the taxonomy proposed in [20], which organises the key subsystems a compiler must provide to meet FT-DQC requirements.

- **Compilation Algorithm**: High-level methods for translating and transforming programs into fault-tolerant logical circuits suitable for distributed execution (e.g., module extraction, IR-level rewrites, and FT circuit templates).
- **Partitioning Techniques**: Methods for dividing circuits across nodes while respecting FT locality (code patches, error domains) and minimising inter-node operations (e.g., Remote CNOTs, teleportations) and their induced error propagation.
- **Resource Allocation and Optimisation**: Policies for placing logical qubits and ancilla patches, provisioning EPR pairs, and budgeting space–time resources (code distance, cycle counts), with objectives such as latency and makespan reduction under FT constraints.
- **Communication and Control**: Abstractions and schedules for inter-node quantum communication and coordinated non-local gates (logical teleportation, lattice surgery), including runtime protocols and classical feedback to maintain Pauli frames and synchronisation.

This structured view treats the compiler as a modular composition of interacting subsystems, each with clear optimisation goals and correctness obligations under fault-tolerant, distributed execution.

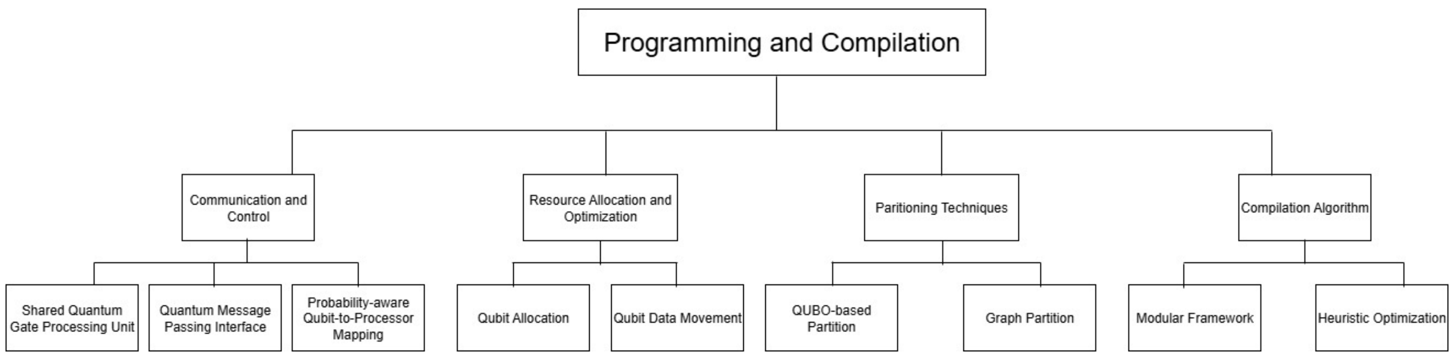

Programming and Compilation
Communication and Control
Resource Allocation and Optimization
Paritioning Techniques
Compilation Algorithm
Shared Quantum Gate Processing Unit
Quantum Message Passing Interface
Probability-aware Qubit-to-Processor Mapping
Qubit Allocation
Qubit Data Movement
QUBO-based Partition
Graph Partition
Modular Framework
Heuristic Optimization

Figure 2.7: Taxonomy of Compiler Functions in FT-DQC. Each layer corresponds to a function essential for maintaining fault tolerance across distributed quantum nodes. Adapted from [20].

(1) Compilation Algorithm

This category focuses on strategies for efficiently constructing and transforming logical quantum circuits. These algorithms serve as the backbone of the compilation process, defining how the quantum algorithm is expressed and optimised prior to hardware realisation.

- **Modular Framework**: Decompose large circuits into subprograms/routines, compile them in isolation with FT templates (e.g., verified ancillae, distillation blocks), and integrate them to limit error spread across fault domains.
- **Heuristic Optimisation**: Employ scalable heuristics (pattern extraction, peephole, template matching) to approximate optimal rewrites when exact optimisation is intractable.

(2) Partitioning Techniques

Partitioning in FT-DQC must minimise communication *and* respect FT locality (stabiliser extraction boundaries, distillation logistics, patch movement).

- **QUBO-based Partitioning**: Encode objectives/constraints (Remote CNOT count, teleportation budget, inter-node concurrency limits) as a quadratic unconstrained binary optimisation problem.
- **Graph Partitioning**: Apply edge-cut or hypergraph partitioning on gate/interaction graphs to cluster tightly coupled logical qubits while aligning boundaries with FT modules.

(3) Resource Allocation and Optimisation

This subsystem manages physical/logical resources under FT constraints.

- **Qubit Allocation**: Map logical qubits to nodes/patches considering code distance, local error rates, and decoder performance; provision ancillae for repeated syndrome extraction.
- **Qubit Data Movement**: Track and coordinate logical state movement (teleportation, braiding, lattice surgery), accounting for success probabilities and Pauli-frame updates.

(4) Communication and Control

Compiler components must orchestrate inter-node communication and non-local operations via robust abstractions and runtime control.

- **Shared Quantum Gate Processing Unit**: Centralise scheduling of remote gates across nodes to reduce synchronisation overhead and enforce FT gate semantics via classical feedback.
- **Quantum Message Passing Interface**: Provide a programming abstraction for remote operations (logical teleportation, lattice surgery) with protocol-aware fallbacks and retries.

- **Probability-aware Qubit-to-Processor Mapping**: Incorporate stochastic success models (e.g., teleportation, entanglement generation) into mapping/scheduling to maximise end-to-end fidelity.

Chapter 3

Related Work

This chapter reviews the research lineage of quantum circuit partitioning and quantum circuit cutting in Distributed Quantum Computing (DQC), and clarifies the problem setting adopted in this thesis (static partitioning; RCX as a representative communication metric) as well as the remaining gaps in the literature. While DQC extends the executable scale of quantum circuits by distributing them across multiple nodes, inter-node operations introduce communication resources as a dominant constraint. Accordingly, prior studies have discussed (i) modeling communication costs, (ii) partitioning/allocation/cutting methods to reduce such costs, and (iii) neighboring tasks such as placement and scheduling [2, 20].

This thesis focuses on static circuit partitioning and aims to systematically organize the relationship between circuit structure and partitioning performance. Therefore, this chapter proceeds as follows: (i) execution models and communication-cost views in DQC, (ii) neighboring tasks (allocation/placement and scheduling), (iii) the historical development of circuit partitioning and circuit cutting, (iv) the positioning of the partitioners considered in this thesis, and (v) the remaining gap in structure-based systematization.

3.1 Execution Models and Communication Costs in DQC

3.1.1 Cost Models for Remote Gates and Teleportation

In DQC, two-qubit gates can be classified into **local** operations that are completed within a single node and **remote** operations that span different nodes. Remote operations typically involve consumption of entanglement resources, classical communication, and synchronization (waiting for measurement outcomes), and thus tend to dominate runtime and resource usage [2, 5, 33].

Representative metrics for communication cost include the following [2]:

- **RCX (Remote CNOT count)**: counts the number of two-qubit operations that cross node boundaries.
- **ebit**: counts the number of consumed EPR pairs (1 EPR pair $\equiv$ 1 ebit).
- **Classical communication, latency, and bandwidth occupation**: expressed as bits, round-trip delay, link utilization, etc.

- **Depth overhead**: additional circuit depth induced by communication procedures and synchronization.

In this thesis, we adopt RCX as a representative metric by projecting the composite cost onto the "number of boundary-crossing two-qubit operations," and perform a relative comparison among partitioners. RCX directly reflects how partitioning changes the communication boundary, and can also serve as a baseline axis when extending the evaluation to other cost models (e.g., ebits or latency).

3.1.2 System-Level Views of DQCE: Interconnect, Scheduling, and Distributed Arithmetic

Communication cost in DQC is not determined solely by the total count of remote operations, but also by the system context that transforms those operations into execution time and resource occupation. In DQCE-style (Distributed Quantum Circuit Execution) studies, the target is an end-to-end system that includes the interconnect topology, switching structure, entanglement generation/distribution capability, and runtime scheduling policies. For example, Q-Fly proposes an optical interconnect inspired by high-radix network designs and evaluates how architectural choices and entanglement-generation resources shape the bottlenecks of distributed execution. Under such a system view, minimizing "how many times a boundary is crossed" remains meaningful, but it does not uniquely determine latency or throughput: temporal concentration of remote procedures, contention on links, and limited parallelism in EPR generation/synchronization can dominate the effective cost even when RCX is similar [24].

A complementary lineage that makes the architecture dependence explicit is the work on quantum multicomputers and distributed arithmetic by Van Meter and collaborators. Van Meter's Ph.D. thesis develops an architectural perspective for distributed-memory quantum multicomputers, emphasizing that the dominant cost depends on node capacity, I/O capability, network characteristics, and the chosen protocol for realizing nonlocal interactions. Building on this view, distributed-memory arithmetic studies analyze how arithmetic subroutines (e.g., adders and modular arithmetic components) interact with communication constraints and how performance depends on the remote-operation implementation and system parameters. These system-oriented results support the premise of this thesis: partitioning performance differences are ultimately meaningful only relative to an execution model and resource constraints, and therefore a clear cost view must precede method comparison [30, 32].

3.1.3 Communication Bottlenecks and Resource Constraints (Bandwidth and Parallelism)

Communication cost is influenced not only by the total amount but also by temporal concentration (congestion). If many remote operations occur simultaneously, link bandwidth, EPR generation capability, and synchronization constraints can increase the overall execution time. Therefore, even when the total RCX is similar, the temporal distribution and parallelism of remote operations can lead to different execution efficiency [2, 5].

This thesis primarily targets static partitioning and uses RCX as the main evaluation metric. However, in later chapters (structure-indicator design), we also introduce indicators that reflect temporal structure (e.g., two-qubit-gate parallelism and layer-wise active sets), so that the framework can be extended to models that explicitly incorporate bandwidth and synchronization constraints.

3.2 Neighboring Tasks Beyond Partitioning

3.2.1 Relationship Between Partitioning and Qubit Allocation / Placement

Partitioning divides the set of logical qubits across multiple nodes and primarily determines where communication (boundary edges) arises. In contrast, *allocation* assigns the resulting clusters to physical nodes, and can significantly affect communication cost through network topology (e.g., connectivity, distance, heterogeneous bandwidth) [2, 5].

Moreover, in Fault-Tolerant Quantum Computing (FTQC) with lattice surgery, *placement* within each node—placing logical qubits (patches) on a 2D plane—affects the cost of two-qubit operations themselves through adjacency and workspace constraints [20, 10, 15]. Hence, partitioning acts as a coarse-grained design decision, while allocation/placement serve as fine-grained design tasks that determine feasibility and local costs. From a software-stack perspective, these tasks are organized at different abstraction layers as well [19].

3.2.2 Scheduling and Dynamic Reconfiguration

To mitigate communication congestion and synchronization waiting, *scheduling*—ordering and parallelizing operations—is also important. Since interaction patterns can change over time, the literature has discussed *dynamic reconfiguration* that updates placements/assignments over phases of the circuit [2, 20, 19].

However, dynamic reconfiguration introduces a new cost: the cost of changing the configuration itself. In NISQ-style circuit models, moving qubits is often realized by sequences of nearest-neighbor SWAPs, which can substantially increase the number of two-qubit gates and the circuit depth. In FTQC with lattice surgery, patch motion/resizing/code deformation and additional procedures to secure workspace consume time/space resources and error budget. Therefore, dynamic reconfiguration is not always beneficial and should be evaluated as a trade-off between "remote operations reduced by reconfiguration" and "operations/waits introduced by reconfiguration" [15].

This thesis primarily aims to clarify the correspondence between structure indicators and method suitability under static partitioning. Nevertheless, since some implemented approaches (e.g., HQA-type methods) may incorporate phase-wise reassignment, we treat them as "partitioning methods with limited dynamic elements" within the unified evaluation framework.

3.2.3 Software-Layer Perspectives and Domestic Workshop References

DQC systems require not only partitioning and allocation but also protocol-level control and software abstractions that coordinate entanglement generation, classical feed-forward, and rule-based execution. As an example of a software-layer viewpoint, Satoh's thesis ("cocori" in the lab context) studies a programming-language approach for RuleSet-based quantum repeaters, highlighting how execution rules and control abstractions can be represented and managed at the software level. Such work complements the partitioning literature by clarifying what assumptions about control, scheduling, and resource management are embedded in an "execution model" [25].

In addition, short workshop papers in domestic venues such as IPSJ research meetings (e.g., the Quantum Software SIG) document early-stage ideas and system prototypes that may not appear as full journal articles. Citing such records improves traceability of the local research lineage and provides context for system assumptions used in later partitioning and execution studies [26].

3.3 Research Lineage of Quantum Circuit Partitioning and Circuit Cutting

3.3.1 Introducing Classical Partitioning Algorithms (Graph Partitioning and Local Refinement)

Circuit partitioning is commonly formulated by representing interaction structure as a graph and minimizing the cut (boundary). Within this formulation, classical graph partitioning algorithms have been reused as basic building blocks. Typical examples include the Kernighan–Lin method for local refinement in bisection [13] and the Fiduccia–Mattheyses (FM) method as a faster refinement heuristic [8].

3.3.2 Hypergraph-Based Partitioning and Tool Utilization

Beyond weighted interaction graphs, hypergraph representations have been adopted to more directly express shared structures and multi-way dependencies in circuits [3]. From an implementation standpoint, there is a practical line of work that leverages existing hypergraph partitioners (e.g., KaHyPar) for optimization [6].

3.3.3 Circuit Cutting (Wire Cut / Gate Cut) and Statistical Reconstruction

As an alternative direction, circuit cutting replaces quantum communication with classical post-processing to evaluate large circuits using smaller devices. CutQC partitions a circuit into fragments that satisfy hardware constraints and combines fragment results on the classical side [28]. MLFT treats the reconstruction from fragments as maximum-likelihood estimation and provides statistically consistent inference [23].

3.4 Partitioners Considered in This Thesis

In this thesis, we select multiple representative approaches from the literature and implement/evaluate them under a unified setting. This section briefly positions each method and clarifies the references. The optimization formulations, pseudocode, and implementation details are described later in Chapter 5 (partitioning algorithms section).

3.4.1 Kernighan–Lin Method

The Kernighan–Lin (KL) method is a classical heuristic that locally improves the cut weight for graph bisection [13]. For multi-way partitioning ($P > 2$), it is commonly constructed via recursive bisection.

3.4.2 FM-Type Methods (Local Refinement)

The Fiduccia–Mattheyses (FM) method is a fast local refinement algorithm based on vertex moves [8]. In quantum circuit partitioning, FM-style refinement can be used as a component for both graph and hypergraph formulations. When using hypergraph representations, a practical implementation choice is to employ an existing partitioner such as KaHyPar [6].

3.4.3 HQA (Hungarian-Based Assignment)

HQA is a framework that improves mappings for modular (multi-node) execution using assignment optimization based on the Hungarian algorithm [7].

3.5 Open Gap: Systematizing the Relationship Between Circuit Structure and Partitioning Performance

3.5.1 Structure-Indicator-Based Comparison, Prediction, and Explainability

A dominant paradigm in prior work is to propose a specific partitioning (or cutting) method and compare communication cost and execution resources on benchmark circuits. However, a systematic methodology that generalizes *which circuit-structural differences* lead to *which method advantages*, and that enables *a priori* method selection for unseen circuits, remains insufficiently developed [28, 23, 3].

To address this gap, this thesis designs circuit-structure indicators and quantitatively analyzes their correspondence with partitioning performance (in particular, relative differences between methods), aiming to provide a framework that simultaneously supports comparison, prediction, and explainability.

3.5.2 Remote-Gate Realization: Telegate and Teledata

A "remote two-qubit gate" is an abstraction whose concrete realization depends on the system protocol. In the distributed-memory quantum multicomputer perspective, two representative realizations are commonly contrasted: **telegate** (teleported gate) and **teledata** (data teleportation) [30, 32].

Telegate (teleported gate). Telegate realizes a nonlocal two-qubit gate (e.g., remote CNOT) via pre-shared EPR pairs, measurements, and classical feed-forward. Under this view, RCX is close to "the number of remote-gate invocations." However, the wall-clock cost can vary substantially with EPR generation time, measurement/feedback latency, synchronization waits, and link contention governed by bandwidth and parallelism constraints. Therefore, identical RCX does not imply identical latency or throughput [2, 5].

Teledata (data teleportation). Teledata teleports the data qubit itself to another node and then executes the operation locally. This shifts the accounting from "remote gates" to "data transfers," making the correspondence between RCX and the underlying communication procedures less direct. Moreover, when teleportation is decomposed into phases (e.g., entanglement generation/pre-distribution versus consumption), parallelism and inventorying of EPR resources can further decouple a simple RCX count from effective execution time [32].

Implication for using RCX as a proxy. RCX is a concise proxy for "how much interaction crosses partition boundaries," but its translation into latency and resource occupancy is affected by (i) the telegate/teledata choice, (ii) whether EPR generation is the dominant bottleneck, (iii) the number of transceiver qubits and the network parallelism, and (iv) temporal concentration and contention of remote procedures. This thesis adopts RCX to unify comparisons under static partitioning, while later chapters explicitly incorporate temporal-structure indicators to keep the framework extendable to bandwidth- and synchronization-aware cost models.

3.5.3 Cat-Entangler / Cat-Disentangler as Building Blocks for Nonlocal Control

At a finer granularity, nonlocal operations can be expressed as compositions of primitive procedures. Yimsiriwattana and Lomonaco introduced **cat-entangler** and **cat-disentangler**, showing that distributing control qubits via cat states (generalized GHZ states) enables systematic constructions of nonlocal controlled operations (including nonlocal CNOT) and teleportation-based procedures [33].

From this viewpoint, "one remote CNOT" (one unit counted by RCX) may correspond to a multi-stage protocol: (1) cat-state preparation, (2) local operations conditioned on distributed control, and (3) cat-state disentangling. Importantly, introducing cat-based primitives does not necessarily reduce the logical CNOT count of the circuit; rather, it provides an alternative realization of nonlocal control that can introduce additional local entangling operations, measurements, and feed-forward. Conversely, in situations where multiple nonlocal interactions can be amortized over a shared cat state (e.g., distributed fan-out or multi-target control), cat-based constructions can reduce the number of expensive cross-node procedures and alleviate bottlenecks related to EPR generation, synchronization, and link occupation.

However, such reductions are not purely a consequence of the original circuit structure; they arise from implementation freedom in the execution model (e.g., whether cat-state resources are available and efficiently prepared). Therefore, while RCX remains a useful proxy, it does not explicitly represent cat-state preparation/disentangling overhead, EPR-generation limits, synchronization latency, or parallelism constraints. This limitation becomes more pronounced when execution models allow protocol-level transformations that change the effective "remote-procedure count" without changing the high-level algorithm.

Chapter 4

Design of Quantum Circuit Structural Indicators

In this chapter, we define structural indicators computed from a circuit in order to explain why communication costs differ across partitioning methods from the perspective of circuit structure. The goal is to provide a foundation for comparing and predicting which partitioning methods tend to be advantageous based on the circuit ' s intrinsic structure, without relying on heuristics tied to a specific benchmark or a specific partitioning method.

The structural indicators used in this study can be broadly organized into the following groups. Terminology for circuit statistics such as gate density and entanglement variance is organized in QASMBench [14]. Modularity follows the classical definition for evaluating community structure [21]. Betweenness centrality is also classically defined as a centrality measure [11]. Laplacian eigenvalues (in particular λ_2) are positioned in spectral graph theory as indicators of partitionability [9, 4].

4.1 Grouping of the Indicators

- Density group (D, GateDensity, M, m, $|H_t|$, $\bar{h}$, CircuitWidth): represents whether two-qubit gates are sparse or dense, and the scale of interactions in terms of counts, types, totals, etc.
- Skewness group (EntVar, Rdup, Hmax, Var_ent, $R_{\text{long}}(\tau)$): represents how strongly interactions concentrate on particular qubits (or particular pairs), including the strength of repeated use of the same pair (including temporal persistence).
- Parallelism group (Pavg, Pmax, Peff): represents the number of independent two-qubit gates executable within the same layer (time step).
- Spectral / partitionability group ($S_{\text{spec}}^{(P)}$, $\bar{\lambda}_2^{(\text{rec})}$, Qmod, Bk): represents " natural cut boundaries " and community structure in the interaction graph, and how likely connections remain after partitioning.
- Balance / capacity group (Bsize, CVC, Savg, Roverload, Rhyper, Q_{node}): represents post-partition cluster size imbalance, overload of hot clusters, boundary-edge ratios, capacity violations, and related effects.

Below, we first define the notation in a unified manner (Section 4.1), and then formalize the indicators group by group.

Table 4.1: Quick reference of structural indicators used in this chapter

Symbol	Variable name	One-line description
CircuitWidth	Circuit width indicator	Number of qubits actually used in the circuit (n_{active}).
D	Two-qubit coupling density	Defined as $D = m/\binom{n}{2}$; represents the density of used pairs.
M	Total edge weight	Defined as $M = \sum_{i<j} w_{ij}$; represents the total number of two-qubit gates (sum of weights).
m	Number of interaction pair types	Defined as $m = \lvert E \rvert$; represents the number of distinct interacting pairs.
GateDensity	Gate density	Defined as $(G_{1q} + 2G_{2q})/(\text{CircuitDepth} \times \text{CircuitWidth})$.
$\lvert H_t \rvert$	Number of simultaneously active qubits	Number of qubits active simultaneously in layer t.
$\bar{h}$	Mean number of active qubits	Mean $\bar{h} = \frac{1}{T}\sum_t \lvert H_t \rvert$.
EntVar	Entanglement variance	Measures imbalance of two-qubit gate load across qubits.
R_{dup}	Duplication ratio (repetition ratio)	Defined as $\text{R}_{\text{dup}} = M/\lvert E \rvert$; represents the strength of pair repetition.
$H_{\max}$	Hot-qubit concentration	Defined as $H_{\max} = \max_i p_i$; represents dominance of the busiest qubit.
Var_{ent}	Temporal entanglement variance	Variance $\text{Var}_{\text{ent}} = \frac{1}{T}\sum_t (E_t - \bar{E})^2$.
$R_{\text{long}}(\tau)$	Long-lived hot ratio	Fraction of M accounted for by the total weight of long-lived hot edges.
P_{avg}	Average parallelism	Defined as $P_{\text{avg}} = \frac{1}{T}\sum_t g_t$.
$P_{\max}$	Maximum parallelism	Defined as $P_{\max} = \max_t g_t$.
P_{eff}	Effective parallelism	Defined as $P_{\text{eff}} = M/T_{2\text{q}}$.
Q_{mod}	Modularity	Measures how well “ dense within clusters / sparse between clusters ” holds.
$\bar{\lambda}_2^{(\text{rec})}$	Recursive mean λ_2	Defined as $\bar{\lambda}_2^{(\text{rec})} = \frac{1}{N_{\text{sub}}}\sum_h \lambda_2^{(h)}$.
$S_{\text{spec}}^{(P)}$	Spectral sum for P-partition	Defined as $S_{\text{spec}}^{(P)} = \frac{1}{P-1}\sum_{k=2}^{P} \lambda_k$.
B_k	Bridge concentration	Defined as $B_k = \sum_{i \in H_k^{(b)}} \tilde{b}_i$ (concentration on top nodes).

(Continued on next page)

Symbol	Variable name	One-line description
Q_{node}	Node capacity	Under equal-capacity assumption, $Q_{\text{node}} = n/P$.
R_{hyper}	Capacity violation ratio	Time average of $\max\{0, \lvert H_t\rvert - Q_{\text{node}}\}/\lvert H_t\rvert$.
S_{avg}	Average surface ratio	Weighted mean $S_{\text{avg}} = \sum_k \frac{\lvert C_k\rvert}{n} S_k$.
CV_C	Coefficient of variation	Dispersion indicator $\mathrm{CV}_C = \sigma_C/\mu_C$.
B_{size}	Capacity consistency	Measures how well cluster sizes align with Q_{node}.
R_{overload}	Overload ratio	Fraction of internal load accounted for by capacity-violating clusters ($\lvert C_k\rvert > Q_{\text{node}}$).

1 Definition of Symbols

We represent a quantum circuit as a weighted undirected graph consisting of a set of qubits and a set of two-qubit gates. Let the set of qubits (circuit width) be

$$Q = \{q_1, q_2, \ldots, q_n\}, \qquad n = |Q|$$

where qubit indices start from 1.

Let the set of two-qubit gates be

$$G_2 = \{g_1, g_2, \ldots, g_{m_{2q}}\}, \qquad m_{2q} = |G_2|$$

and regard each g_ℓ as acting on some qubit pair (u_ℓ, v_ℓ).

Define the number of two-qubit gates acting between q_i and q_j as the edge weight

$$w_{ij} > 0 \quad (1 \le i < j \le n).$$

This represents how many times that qubit pair interacts.

Define the weighted degree of each node (qubit) q_i as

$$d_i = \sum_{j \ne i} w_{ij}.$$

This represents the total number of two-qubit gates involving q_i (how “busy” that qubit is).

Define the total edge weight of the entire graph as

$$M = \sum_{1 \le i < j \le n} w_{ij} = \frac{1}{2} \sum_{i=1}^{n} d_i.$$

If the weights are gate counts, then M is the total number of two-qubit gates in the circuit.

Let the edge set of the complete graph be

$$E_{\text{full}} = \{(i,j) \mid 1 \le i < j \le n\}, \qquad |E_{\text{full}}| = \binom{n}{2}.$$

Define the set of pairs on which at least one two-qubit gate acts as

$$E = \{(i,j) \mid w_{ij} > 0,\ 1 \le i < j \le n\}, \qquad |E| = m.$$

Here, m is the number of distinct interacting pair types.

Mini example (reused below) Let $n = 4$ and suppose the weights are

$$w_{12} = 5, \quad w_{13} = 1, \quad w_{24} = 1, \quad w_{34} = 1, \quad \text{and 0 otherwise.}$$

Then

$$M = 5 + 1 + 1 + 1 = 8, \quad |E| = 4,$$

and

$$d_1 = w_{12} + w_{13} = 6,\ d_2 = w_{12} + w_{24} = 6,\ d_3 = w_{13} + w_{34} = 2,\ d_4 = w_{24} + w_{34} = 2.$$

2 Density Group (Sparsity/Density, Types, Totals, Scale)

2.1 Circuit Width

Define the number of qubits actually used in the circuit as

$$\text{CircuitWidth} = n_{\text{active}}.$$

Intuition In distributed execution, this is the most basic scale indicator describing how many qubits the circuit uses in total. If the number of nodes P is fixed, then $Q_{\text{node}} = n/P$ is determined, and **the larger the circuit width, the less likely the circuit fits within a single node, and the more likely partitioning becomes necessary**.

Simple example If $n_{\text{active}} = 16$ is handled with $P = 4$ nodes, then $Q_{\text{node}} = 4$. If $n_{\text{active}} = 32$, then even with the same P, one needs $Q_{\text{node}} = 8$; if node capacity is fixed, this may cause " infeasibility / increased communication. "

Rule of thumb (high/low)

- Rather than being " good/bad, " this is simply a **scale** parameter.

2.2 Density of Two-Qubit Couplings

Define

$$D = \frac{m}{\binom{n}{2}}.$$

Intuition This indicates, among all qubit pairs that are theoretically possible, how many are actually used:

$$D \text{ large} \Rightarrow \text{“ many pairs interact (high density) ”},$$

$$D \text{ small} \Rightarrow \text{“ only limited pairs are used (sparse) ”}.$$

The denser the circuit, the more likely cut boundaries appear no matter where one partitions, and the harder it is to substantially reduce RCX.

Simple example In the mini example, $n = 4$ so $\binom{n}{2} = 6$, and $m = |E| = 4$, hence

$$D = \frac{4}{6} \approx 0.667.$$

Rule of thumb (high/low)

- $D \ll 1$ **(low)**: sparse; natural partition boundaries are likely to exist, and cut-based methods tend to work well.
- $D \to 1$ **(high)**: nearly all pairs interact; any partition tends to incur heavy boundaries, leaving little room for improvement.

2.3 Gate Density

Let the number of single-qubit gates be G_{1q}, the number of two-qubit gates be G_{2q}, and the circuit depth be CircuitDepth. Define

$$\text{GateDensity} = \frac{G_{1q} + 2G_{2q}}{\text{CircuitDepth} \times \text{CircuitWidth}}.$$

Intuition This measures **how much computation is packed into a short time span**. Even if the total number of gates is the same, a smaller depth yields a higher density, which tends to strengthen simultaneous communication and resource demands.

Simple example If CircuitWidth $= 10$, CircuitDepth $= 20$, $G_{1q} = 100$, $G_{2q} = 50$, then

$$\text{GateDensity} = \frac{100 + 2 \cdot 50}{20 \cdot 10} = 1.$$

Rule of thumb (high/low)

- **Low**: more slack in the time dimension, weaker simultaneous demand.
- **High**: stronger congestion; simultaneous cross-partition interaction demands (contention) tend to increase after partitioning.

2.4 Layer-wise Hyperedges

Layer decomposition in the implementation (greedy / ASAP) In this study, layers t ($t = 1, \dots, T$) are defined by a greedy (ASAP) decomposition. Specifically, we scan the circuit instruction list from the beginning, track for each qubit q the most recent layer index $t_{\text{last}}(q)$ in which it was used, and place each gate into the earliest executable layer given by $\max_q t_{\text{last}}(q) + 1$. Within the same layer, gates sharing the same qubit do not coexist. Hereafter, $H_t,\ g_t,\ T,\ T_{2\text{q}}$ are defined based on this layer decomposition.

Let the set of active qubits in layer t be

$$H_t = \{\, i \mid \text{in layer } t,\ \text{qubit } i \text{ participates in a gate} \,\},$$

and define

$$\bar{h} = \frac{1}{T}\sum_{t=1}^{T} |H_t|.$$

Intuition $|H_t|$ is the size of the set of qubits that are simultaneously " active " at that time step. The larger $\bar{h}$ is, the more qubits are active simultaneously on average, and the more likely " simultaneous cross-partition interactions " occur after partitioning.

Simple example If $T = 4$ and $|H_t| = (2, 2, 6, 2)$, then

$$\bar{h} = \frac{2+2+6+2}{4} = 3.$$

Rule of thumb (high/low)

- **Low**: few simultaneously active qubits; many phases can be completed within a single node.
- **High**: many simultaneously active qubits; simultaneous multi-node requirements tend to increase.

3 Skewness Group (Concentration, Repetition, Temporal Persistence)

3.1 Entanglement Variance

Let $G_{2q,i}$ be the number of two-qubit gates acting on qubit q_i, and define the mean

$$\bar{G}_{2q} = \frac{1}{n}\sum_{i=1}^{n} G_{2q,i}.$$

Then define

$$\text{EntVar} = \frac{\log\left(\sum_{i=1}^{n}(G_{2q,i} - \bar{G}_{2q})^2 + 1\right)}{n}.$$

Intuition This measures **how imbalanced the two-qubit gate load is across qubits**:

$$\text{EntVar large} \Rightarrow \text{“ hot qubits exist / load concentrates on a subset ”},$$

$$\text{EntVar small} \Rightarrow \text{“ load is uniformly distributed ”}.$$

When concentration is strong, communication can be greatly reduced if the hot region is contained within a single node; however, **RCX can increase sharply if the hot region spans multiple nodes**.

Simple example In the mini example, if we approximate $G_{2q,i} \approx d_i = (6, 6, 2, 2)$, then

$$\bar{G}_{2q} = 4, \qquad \sum_i (G_{2q,i} - \bar{G}_{2q})^2 = 16,$$

so

$$\text{EntVar} = \frac{\log(17)}{4},$$

which is larger than the uniform case $(4, 4, 4, 4)$.

Rule of thumb (high/low) Although this indicator is normalized, it still depends on circuit scale; thus it is primarily used for **relative comparisons**. Intuitively,

- **Low (uniform in a favorable sense)**: weak hotspots; differences among strategies that “ protect specific points ” are less likely to appear.
- **High (strong concentration)**: strong hotspots; hotspot-prioritizing methods tend to be relatively advantageous.

3.2 Duplication Ratio (Edge Duplication Ratio)

Define

$$\mathrm{R}_{\text{dup}} = \frac{M}{|E|} = \frac{\sum_{1 \le i < j \le n} w_{ij}}{|E|}.$$

What are M and $|E|$ here?

$$M = \sum_{1 \le i < j \le n} w_{ij}$$

is the total edge weight of the graph, i.e., the **total number of two-qubit gates executed in the entire circuit**. Also,

$$|E| = \big|\{(i, j) \mid w_{ij} > 0\}\big|$$

is the **number of distinct qubit pairs (pair types) that interact at least once**.

Intuition Because

$$\mathrm{R_{dup}} = \frac{\text{total two-qubit gates}}{\text{number of interacting pair types}},$$

it represents **how many times, on average, a single pair type appears**:

$$\mathrm{R_{dup}} \text{ large} \Rightarrow \text{“ strong repetition of a small number of pairs ”},$$

$$\mathrm{R_{dup}} \text{ small} \Rightarrow \text{“ spread across many pairs (mostly one-off) ”}.$$

The stronger the repetition, the more RCX can be reduced if those repeated pairs are kept within the same node.

Simple example In the mini example, $M = 8$ and $|E| = 4$, hence

$$\mathrm{R_{dup}} = \frac{8}{4} = 2$$

(i.e., each pair appears 2 times on average).

Rule of thumb (high/low)

- $\mathrm{R_{dup}} \approx 1$ **(low)**: nearly all pairs occur once; there are few frequency-wise “ pairs to protect, ” so differences are less likely to appear.
- $\mathrm{R_{dup}} \gg 1$ **(high)**: strong repetition of a small number of pairs; frequency-based / hotspot-prioritizing methods tend to work well.

3.3 Hot-Qubit Concentration

Let

$$p_i = \frac{d_i}{\sum_{j=1}^{n} d_j} = \frac{d_i}{2M}, \qquad \sum_{i=1}^{n} p_i = 1,$$

and define

$$H_{\max} = \max_{1 \le i \le n} p_i, \qquad \frac{1}{n} \le H_{\max} \le 1.$$

Intuition This represents **the fraction of all two-qubit gates borne by the busiest qubit**:

$$H_{\max} \text{ large} \Rightarrow \text{“ one (or a small number of) qubits dominates ”}.$$

In this case, assignments that let that qubit span multiple nodes can sharply increase RCX.

Simple example In the mini example, $\sum_i d_i = 16$, so

$$p = (6/16, 6/16, 2/16, 2/16),$$

and thus $H_{\max} = 6/16 = 0.375$.

Rule of thumb (high/low)

- **The lower bound is** $1/n$ (for perfectly uniform load, $H_{\max} \approx 1/n$).
- $H_{\max}$ **close to** $1/n$ **(low)**: weak hotspots; differences due to protecting specific qubits are less likely to appear.
- **Large** $H_{\max}$ **(high)**: strong hotspots; protecting hot qubits (fixed placement or frequency prioritization) tends to be effective.

3.4 Temporal Entanglement Variance

Let E_t be the entanglement amount at layers $t = 1, \dots, T$, and define

$$\bar{E} = \frac{1}{T}\sum_{t=1}^{T} E_t, \qquad \mathrm{Var}_{\mathrm{ent}} = \frac{1}{T}\sum_{t=1}^{T}(E_t - \bar{E})^2.$$

Intuition This measures whether entanglement is **temporally skewed (bursty)**. The more concentrated it is in specific phases, the harder it is for static partitioning to maintain an " always good " placement.

What are E_t and T? Decompose the circuit into layers along the depth (time) dimension and index them as

$$t = 1, 2, \dots, T,$$

where T is the total number of layers.

E_t represents the " entanglement amount at time t. " In this study, as the simplest choice, we set

$$E_t := \text{the number of two-qubit gates } g_t \text{ in layer } t$$

(or the total weight if weighted). Thus, E_t is a time series describing how much two-qubit interaction occurs at time t.

Why use $T = 4$ in the example? To simplify the explanation, we assume $T = 4$ layers in the example. This only means the circuit has four time slices (layers); there is no special meaning attached to the gate placement at $t = 1, 2, 3$. It simply refers to " layer 1, " " layer 2, " and so on.

Example 1: Why $E = (0, 0, 10, 0)$ is " concentrated " $E = (0, 0, 10, 0)$ means

$$(E_1, E_2, E_3, E_4) = (0, 0, 10, 0).$$

This corresponds to

- 0 two-qubit gates in layer 1 (mostly local operations),
- 0 in layer 2 as well,
- suddenly 10 in layer 3 (a strong single peak),
- 0 in layer 4,

so deviations from the mean are large and the variance becomes large.

Indeed,

$$\bar{E} = \frac{0+0+10+0}{4} = 2.5,$$

and

$$\begin{aligned}\mathrm{Var}_{\mathrm{ent}} &= \frac{1}{4}\Big[(0-2.5)^2 + (0-2.5)^2 + (10-2.5)^2 + (0-2.5)^2\Big] \\ &= \frac{1}{4}\Big[6.25 + 6.25 + 56.25 + 6.25\Big] = \frac{75}{4} = 18.75,\end{aligned}$$

which is large.

Example 2: Why $E = (2, 3, 2, 3)$ is " smooth "

$$(E_1, E_2, E_3, E_4) = (2, 3, 2, 3)$$

has about 2–3 gates in each layer, with no single layer standing out. Hence deviations from the mean are small and the variance is small.

$$\begin{aligned}\bar{E} &= \frac{2+3+2+3}{4} = 2.5, \\ \mathrm{Var}_{\mathrm{ent}} &= \frac{1}{4}\Big[(2-2.5)^2 + (3-2.5)^2 + (2-2.5)^2 + (3-2.5)^2\Big] \\ &= \frac{1}{4}\Big[0.25 + 0.25 + 0.25 + 0.25\Big] = 0.25,\end{aligned}$$

which is indeed small.

Summary (what this indicator captures)

$$\begin{aligned}\mathrm{Var}_{\mathrm{ent}}\ \text{large} &\Rightarrow \text{“two-qubit interactions concentrate in specific phases”} \\ &\Rightarrow \text{multi-node communication / resource contention is likely at peaks,} \\ \mathrm{Var}_{\mathrm{ent}}\ \text{small} &\Rightarrow \text{“two-qubit interactions are temporally dispersed”} \\ &\Rightarrow \text{static partitioning is less likely to break down.}\end{aligned}$$

Rule of thumb (high/low)

- **Low**: temporally uniform; the static-partitioning assumption is less likely to break.
- **High**: temporally concentrated; communication and resource contention are likely at peaks.

3.5 Edge Lifetimes and the Long-Lived Hot-Edge Ratio

First, $t_{\mathrm{first}}(i,j)$ and $t_{\mathrm{last}}(i,j)$ After decomposing the circuit into layers $t = 1, \ldots, T$, for an edge (qubit pair) (i,j), let $t_{\mathrm{first}}(i,j)$ be the **layer where the corresponding two-qubit gate first appears** and $t_{\mathrm{last}}(i,j)$ be the **layer where it appears last**. Thus,

$$L_{ij} = t_{\mathrm{last}}(i,j) - t_{\mathrm{first}}(i,j)$$

represents how widely in time that pair appears across the circuit.

What is the threshold τ? τ is a **lifetime (time-span) threshold**. Edges satisfying

$$L_{ij} \geq \tau$$

are regarded as **long-lived**. For example, if $\tau = 10$, then a pair whose first and last occurrences are at least 10 layers apart is classified as long-lived.

What is the threshold w_{th}? w_{th} is a **weight (occurrence count) threshold**. Edges satisfying

$$w_{ij} \geq w_{\text{th}}$$

are regarded as **hot (frequent)**. For example, if $w_{\text{th}} = 5$, then a pair appearing at least five times is classified as hot.

What does $E_{\text{long}}(\tau)$ collect?

$$E_{\text{long}}(\tau) = \{(i,j) \in E \mid L_{ij} \geq \tau,\ w_{ij} \geq w_{\text{th}}\}$$

is the set of pairs that are both **temporally long-lived and frequent**. $R_{\text{long}}(\tau)$ measures the fraction of the total two-qubit gate volume M accounted for by the total weight of such " long-lived hot " edges:

$$R_{\text{long}}(\tau) = \frac{\sum_{(i,j)\in E_{\text{long}}(\tau)} w_{ij}}{M}.$$

Concrete example (with explicit computation) Consider a circuit with $T = 6$ layers, where the appearances of two-qubit gates (pairs) are as follows:

t	1	2	3	4	5	6
appearing pair	$(1,2)$	$(1,2)$	$(3,4)$	$(1,2)$	$(5,6)$	$(1,2)$

Then

$$\begin{aligned} w_{12} &= 4, & t_{\text{first}}(1,2) &= 1, & t_{\text{last}}(1,2) &= 6, & L_{12} &= 5, \\ w_{34} &= 1, & t_{\text{first}}(3,4) &= 3, & t_{\text{last}}(3,4) &= 3, & L_{34} &= 0, \\ w_{56} &= 1, & t_{\text{first}}(5,6) &= 5, & t_{\text{last}}(5,6) &= 5, & L_{56} &= 0, \end{aligned}$$

and the total weight is

$$M = w_{12} + w_{34} + w_{56} = 4 + 1 + 1 = 6.$$

If we set the thresholds as

$$\tau = 4, \qquad w_{\text{th}} = 3,$$

then

$$(1,2): \ L_{12} = 5 \geq 4 \text{ and } w_{12} = 4 \geq 3 \Rightarrow \text{included},$$

$$(3,4): \ L_{34} = 0 < 4 \Rightarrow \text{excluded}, \qquad (5,6): \ L_{56} = 0 < 4 \Rightarrow \text{excluded}.$$

Hence,

$$E_{\text{long}}(\tau) = \{(1,2)\},$$

and therefore

$$R_{\text{long}}(\tau) = \frac{w_{12}}{M} = \frac{4}{6} \approx 0.667,$$

meaning that " long-lived hot edges account for about 66.7% of all two-qubit gates. "

Making the meaning of high/low more concrete

- **Small** $R_{\text{long}}(\tau)$: few long-lived hot edges. Important interactions tend to localize in short intervals, so static partitioning is less likely to break down.
- **Large** $R_{\text{long}}(\tau)$: important pairs persist over a wide range of the circuit. Once a placement is fixed, some interval is likely to become disadvantageous (static partitioning more readily hits its limits).

Remark: how to choose τ and w_{th} (e.g., heuristics) Because absolute values vary by circuit, comparisons are more stable when using relative criteria rather than fixed constants. For example:

- $\tau = \alpha T$ (e.g., $\alpha = 0.5$ for " appears over at least half the depth "),
- choose w_{th} by an upper percentile (e.g., top 10% by weight).

4 Parallelism Group (Simultaneous Executability of Two-Qubit Gates)

4.1 Two-Qubit Gate Parallelism

Let g_t be the number of two-qubit gates in layer t, and define

$$P_{\text{avg}} = \frac{1}{T}\sum_{t=1}^{T} g_t, \qquad P_{\text{max}} = \max_t g_t.$$

Further, let T_{2q} be the number of layers that contain at least one two-qubit gate, and define

$$P_{\text{eff}} = \frac{M}{T_{2q}}.$$

Intuition This measures **how simultaneously two-qubit gates run** (the strength of simultaneous communication demand). In particular, circuits with large P_{max} have sharp peaks and are likely to become execution bottlenecks.

Simple example If $g = (0, 5, 5, 0)$, then $P_{\text{max}} = 5$ and the peak is large. If $g = (2, 2, 3, 3)$, then $P_{\text{max}} = 3$ and the peak is mitigated.

Rule of thumb (high/low)

- **High** P_{max}: strong peak simultaneous demand; communication contention is likely after partitioning.
- **Low** P_{max}: weak simultaneous demand; scheduling remains easier after partitioning.

5 Spectral / Partitionability Group (Natural Boundaries, Community Structure)

Cluster partitioning in the implementation (community detection) The cluster partition $(C_1, \dots, C_K)$ used in this section is not the output of a partitioning algorithm. Instead, it is an analysis-only partition obtained by applying a greedy community detection method based on modularity maximization to the two-qubit interaction graph G. The number of clusters K is determined automatically by the algorithm. Accordingly, Q_{mod} here is interpreted as an indicator measuring the strength of community structure inherent in the circuit.

5.1 Modularity (Number of Clusters and Modularity)

Assume a cluster partition $(C_1, \dots, C_K)$ is given. Let $\delta(C_i, C_j) = 1$ if in the same cluster and 0 otherwise, and define

$$Q_{\mathrm{mod}} = \frac{1}{2M} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(w_{ij} - \frac{d_i d_j}{2M} \right) \delta(C_i, C_j).$$

Intuition This represents how strongly the property “**dense within clusters and sparse between clusters**” holds:

Q_{mod} large $\Rightarrow$ “natural module boundaries exist that are cheap to cut.”

Hence, partitioning that assigns clusters to nodes tends to be effective.

Simple example Let $C_1 = \{1, 2\}$ mean “vertices 1 and 2 (qubits 1 and 2) are in the same cluster,” and $C_2 = \{3, 4\}$ mean “vertices 3 and 4 are in the same cluster.” In the mini example, if we take $C_1 = \{1, 2\}, C_2 = \{3, 4\}$, the heavy edge $w_{12} = 5$ lies inside a cluster and only light connections remain across the boundary, so Q_{mod} tends to be relatively large for this partition.

Rule of thumb (high/low)

- $Q_{\mathrm{mod}} \approx 0$ **(low)**: weak cluster structure and few natural boundaries; method differences are less likely to appear.
- **Large Q_{mod} (high)**: clear cluster structure; graph-cut / cluster-oriented methods tend to be advantageous.

5.2 Spectral Indicators (Laplacian Eigenvalues)

Let the degree matrix be $\mathbf{D} = \mathrm{diag}(d_1, \dots, d_n)$ and the weighted adjacency matrix be $W = (w_{ij})$, and define

$$L = \mathbf{D} - W.$$

Order the eigenvalues as

$$0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n.$$

Further define

$$\bar{\lambda}_2^{(\mathrm{rec})} = \frac{1}{N_{\mathrm{sub}}} \sum_{h=1}^{N_{\mathrm{sub}}} \lambda_2^{(h)}, \qquad S_{\mathrm{spec}}^{(P)} = \frac{1}{P-1} \sum_{k=2}^{P} \lambda_k.$$

Intuition λ_2 is the **algebraic connectivity** and a spectral indicator of **how easy it is to bipartition**:

$$\lambda_2 \text{ small} \Rightarrow \text{“ weakly connected boundary exists and is easy to cut, ”}$$

$$\lambda_2 \text{ large} \Rightarrow \text{“ any cut tends to be heavy and hard. ”}$$

$\bar{\lambda}_2^{(\mathrm{rec})}$ represents the average difficulty of cutting under recursive partitioning, and $S_{\mathrm{spec}}^{(P)}$ represents the partitioning difficulty when assuming a P-way partition.

Simple example (intuition) A chain (linear) graph is easy to split by cutting near the center, so λ_2 tends to be small. In contrast, a dense and tightly integrated graph tends to have a large λ_2.

Rule of thumb (high/low) Because eigenvalues depend on scale and weights, **absolute comparisons** across circuits are generally avoided. However, the interpretation is clear:

the smaller λ_2 is, the easier it is to partition; the larger it is, the harder it is.

5.3 Centrality of Bridge Qubits

Let b_i be the betweenness centrality of vertex i, and define

$$\tilde{b}_i = \frac{b_i}{\sum_{j=1}^{n} b_j}, \qquad B_k = \sum_{i \in H_k^{(b)}} \tilde{b}_i,$$

where $H_k^{(b)}$ is the set of the top k vertices with large $\tilde{b}_i$.

Intuition This measures whether **inter-module bridging is concentrated on a small number of qubits**:

$$B_k \text{ large} \Rightarrow \text{“ the top } k \text{ vertices become communication bottlenecks. ”}$$

In that case, the placement of those few qubits tends to dominate partitioning performance, and this becomes a factor whereby **methods that can protect them are more likely to win (yield smaller RCX)**.

Simple example (intuition) In a dumbbell-shaped graph where two dense cliques are connected by a thin corridor, the corridor endpoints have high betweenness centrality; the top k nodes capture much of it, making B_k large.

Rule of thumb (high/low)

- **Small** B_k: bridges are dispersed; differences due to handling specific vertices are less likely to appear.
- **Large** B_k: bridges are concentrated; the placement of specific qubits is more likely to decide outcomes.

6 Balance / Capacity Group (Capacity Consistency, Boundary Ratio, Overload, Violations)

6.1 Equal-Capacity Node Assumption and Node Capacity

Let P be the number of nodes in the distributed system, and define the number of logical qubits each node can hold (capacity) as

$$Q_{\text{node}} = \frac{n}{P}$$

(assuming all nodes have equal capacity).

Intuition As P increases, Q_{node} decreases, so **the same circuit becomes harder to fit**. Therefore, capacity-dependent indicators must be **compared with P fixed**.

Simple example If $n = 32$ and $P = 4$, then $Q_{\text{node}} = 8$. If $P = 8$, then $Q_{\text{node}} = 4$, making " simultaneously active sets " and " hot clusters " more likely to exceed capacity.

Rule of thumb Q_{node} itself is a condition rather than a goodness indicator, but for the same circuit, increasing P tends to increase communication.

6.2 Capacity Violation Ratio of Layer Hyperedges

Define the capacity violation ratio as

$$R_{\text{hyper}} = \frac{1}{T} \sum_{t=1}^{T} \frac{\max\{0,\ |H_t| - Q_{\text{node}}\}}{|H_t|}.$$

Intuition This measures **the fraction of layers where the simultaneously active set does not fit within a single node**:

$R_{\text{hyper}} > 0 \Rightarrow$ " at that moment, any static partition necessarily requires multiple nodes. "

That is, R_{hyper} plays a role close to a **structural lower bound** on communication.

Simple example If $Q_{\text{node}} = 4$ and a layer has $|H_t| = 6$, then

$$\frac{\max\{0, 6-4\}}{6} = \frac{2}{6}$$

is added. The more such layers there are, the larger R_{hyper} becomes.

Rule of thumb (high/low)

- $R_{\text{hyper}} \approx 0$ **(good)**: many layers fit within a single node; there is substantial room to reduce communication by choosing a partition.
- **Large** R_{hyper} **(bad)**: capacity violations occur frequently regardless of partitioning, making communication avoidance structurally difficult.

6.3 Consistency Between Cluster Structure and Capacity (Partitioning Difficulty)

Partition the interaction graph $G = (V, E, w)$ into a cluster set

$$\mathcal{C} = \{C_1, C_2, \ldots, C_K\}, \quad C_k \subseteq V.$$

Internal/boundary weights and surface ratio Define

$$W_{\text{in}}(k) = \sum_{\substack{i,j \in C_k \\ (i,j) \in E}} w_{ij}, \qquad W_{\text{bd}}(k) = \sum_{\substack{i \in C_k,\ j \notin C_k \\ (i,j) \in E}} w_{ij},$$

and

$$S_k = \frac{W_{\text{bd}}(k)}{W_{\text{in}}(k) + W_{\text{bd}}(k)}, \qquad S_{\text{avg}} = \sum_{k=1}^{K} \frac{|C_k|}{n} S_k.$$

Intuition S_k is the ratio of “how self-contained the cluster is internally” versus “how strongly it connects to the outside”:

$$S_k \text{ small} \Rightarrow \text{“a good module with a thin boundary,”}$$

$$S_k \text{ large} \Rightarrow \text{“external connections dominate.”}$$

The smaller S_{avg} is, the more likely communication decreases when clusters are placed onto nodes.

Simple example If the internal weight is $W_{\text{in}} = 5$ and the boundary weight is $W_{\text{bd}} = 2$, then

$$S_k = \frac{2}{5+2} \approx 0.286,$$

indicating a cluster dominated by internal self-containment.

Rule of thumb (high/low)

- **Low** S_{avg} **(good)**: strong internal self-containment; cluster-oriented partitioning tends to work well.
- **High** S_{avg} **(bad)**: many external connections; communication tends to remain under any partition.

Cluster size distribution and capacity consistency Using the mean μ_C and standard deviation σ_C of cluster sizes $|C_k|$, define

$$\mathrm{CV}_C = \frac{\sigma_C}{\mu_C},$$

and

$$B_{\text{size}} = 1 - \frac{1}{K} \sum_{k=1}^{K} \frac{\big||C_k| - Q_{\text{node}}\big|}{Q_{\text{node}}}.$$

Intuition CV_C measures **the dispersion of cluster sizes**. The larger the dispersion, the more likely a “ giant cluster ” appears, forcing any method to cut it. B_{size} measures **how well cluster sizes align with the node capacity** Q_{node}, and is better when it is closer to 1.

Simple example If $Q_{\text{node}} = 4$ and the cluster sizes are $(4, 4, 4, 4)$, then $B_{\text{size}} \approx 1$. In contrast, if they are skewed like $(10, 2, 2, 2)$, then B_{size} decreases and partitioning becomes difficult.

Rule of thumb (high/low)

- **Low** CV_C **(good)**: sizes are uniform and easy to place.
- **High** CV_C **(bad)**: giant clusters coexist, tending to increase communication.
- B_{size} **close to 1 (good)**: aligns with capacity; clusters can be placed as-is.
- **Small** B_{size} **(bad)**: deviates from capacity; some cutting becomes necessary.

Hot clusters and overload ratio Define

$$F_k = \sum_{\substack{i,j \in C_k \\ (i,j) \in E}} w_{ij}, \quad F_{\text{tot}} = \sum_{k=1}^{K} F_k, \quad f_k = \frac{F_k}{F_{\text{tot}}},$$

and

$$R_{\text{overload}} = \frac{\displaystyle\sum_{k:\ |C_k| > Q_{\text{node}}} F_k}{\displaystyle\sum_{k=1}^{K} F_k}.$$

Intuition f_k indicates **which clusters concentrate internal two-qubit gate load**. R_{overload} is the fraction of total internal load accounted for by **clusters that have large internal load but cannot be contained within a single node due to capacity violation** ($|C_k| > Q_{\text{node}}$):

$$R_{\text{overload}} \text{ large} \Rightarrow \text{“ communication is structurally unavoidable. ”}$$

Simple example (intuition) If a hot cluster satisfies $|C_k| > Q_{\text{node}}$ and has a large F_k/F_{tot}, then interactions that one would like to keep within the cluster are forced to be split across nodes, making them likely to become remote.

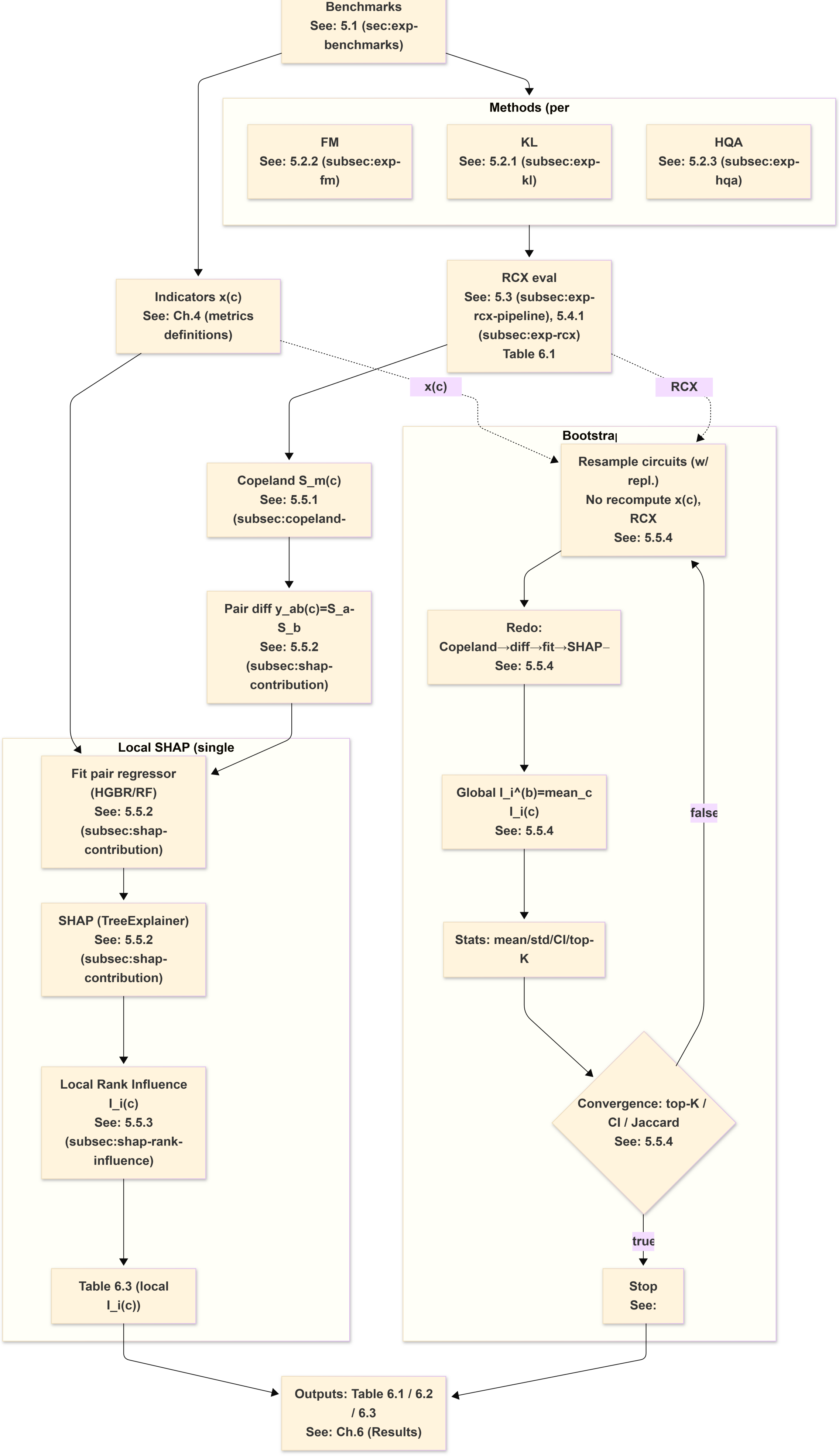
Rule of thumb (high/low)

- $R_{\text{overload}} \approx 0$ **(good)**: hot interactions tend to fit within capacity, leaving substantial room for improvement by choosing partitions.
- **Large** R_{overload} **(bad)**: the hot part must be split, so communication is difficult to reduce under any method (and method differences tend to shrink).

Chapter 5

Experimental Setup

This figure illustrates the experimental setup constructed in this study. It also allows readers to confirm where each component (box) is described in the thesis.


Benchmarks
See: 5.1 (sec:exp-benchmarks)
Methods (per
FM
See: 5.2.2 (subsec:exp-fm)
KL
See: 5.2.1 (subsec:exp-kl)
HQA
See: 5.2.3 (subsec:exp-hqa)
Indicators x(c)
See: Ch.4 (metrics definitions)
RCX eval
See: 5.3 (subsec:exp-rcx-pipeline), 5.4.1 (subsec:exp-rcx)
Table 6.1
x(c)
RCX
Bootstra
Resample circuits (w/ repl.)
No recompute x(c), RCX
See: 5.5.4
Copeland S_m(c)
See: 5.5.1
(subsec:copeland-
Pair diff y_ab(c)=S_a-S_b
See: 5.5.2
(subsec:shap-contribution)
Redo:
Copeland→diff→fit→SHAP–
See: 5.5.4
Local SHAP (single
Fit pair regressor (HGBR/RF)
See: 5.5.2
(subsec:shap-contribution)
Global I_i^(b)=mean_c I_i(c)
See: 5.5.4
false
SHAP (TreeExplainer)
See: 5.5.2
(subsec:shap-contribution)
Stats: mean/std/CI/top-K
Local Rank Influence I_i(c)
See: 5.5.3
(subsec:shap-rank-influence)
Convergence: top-K / CI / Jaccard
See: 5.5.4
true
Table 6.3 (local I_i(c))
Stop
See:
Outputs: Table 6.1 / 6.2 / 6.3
See: Ch.6 (Results)

5.1 Benchmark Circuit Set

In order to observe how the relative strengths and weaknesses of partitioning algorithms vary depending on circuit type, this study uses multiple benchmark circuit families whose computational goals and circuit structures differ substantially. This section explains what computation each circuit generally performs in the experiments. The definitions of the structural indicators and the procedures to compute them are described in the previous chapter, and the results (including RCX) are presented in the next chapter.

5.1.1 QASM-bench

QASM-bench (QASMBench) is a quantum-circuit benchmark suite provided in the OpenQASM format, and is characterized by including a wide variety of computational kernels such as arithmetic circuits and machine-learning-oriented circuits. In this study, in order to include circuits with different properties (e.g., arithmetic, comparison, machine learning), we adopt the following circuits. The suffix `_nXX` in the circuit name denotes the scale of the circuit family (primarily the number of qubits as the circuit width).

- `bigadder_n16`
- `multiplier_n15`, `multiplier_n45`
- `square_root_n18`
- `swap_test_n25`
- `dnn_n16`

(1) Adder (`bigadder_n16`)

An adder circuit is an arithmetic circuit that computes the sum of two numbers on quantum registers. In general, it contains a chain of controlled operations with carry propagation, and it appears widely as a fundamental building block of quantum arithmetic.[14]

(2) Multiplier (`multiplier_n15`, `multiplier_n45`)

A multiplier circuit implements multiplication in quantum arithmetic. In QASMBench, it is typically organized as a shift-and-add construction.[14] Compared with addition, the computation tends to be more complex, and interactions spanning multiple registers tend to increase. In this study, we use two different sizes and also observe how partitioning difficulty changes as the circuit size increases.

(3) Square root (`square_root_n18`)

The square-root circuit implements quantum square-root computation based on amplitude amplification (Grover-style iteration).[14] While it is an arithmetic circuit, it exhibits interaction patterns characteristic of iterative structure.

(4) SWAP test (`swap_test_n25`)

The SWAP test is a fundamental circuit for estimating the distance or overlap (similarity) between two quantum states.[14] Because it is centered on a controlled-SWAP and has an interference-based structure that compares two registers, it has properties different from arithmetic circuits.

(5) DNN (`dnn_n16`)

The DNN circuit is a machine-learning-oriented circuit constructed as a sample quantum neural network (QNN/DNN).[14] Because it can include circuit patterns different from arithmetic circuits (e.g., iterative structure and forms of locality), it is useful as a comparison target for partitioning characteristics.

5.1.2 SupermarQ Families (GHZ / Mermin–Bell / Hamiltonian Simulation)

This study also uses the GHZ, Mermin–Bell, and Hamiltonian-simulation (TFIM) families as representative examples of circuit structure. These are organized as benchmark families reflecting representative tasks such as entanglement generation, evaluation of many-body correlations, and time evolution.[29] The circuits used in this study are as follows.

- GHZ: `ghz_16`, `ghz_32`, `ghz_64`
- Mermin–Bell: `mermin_bell_16`
- TFIM Hamiltonian simulation: `tfim_hamiltonian_sim_16`, `tfim_hamiltonian_sim_32`, `tfim_hamiltonian_sim_64`

(1) GHZ (`ghz_16`, `ghz_32`, `ghz_64`)

GHZ circuits generate a GHZ state spanning multiple qubits. They are typically implemented by starting from a single qubit and expanding entanglement step by step, and are suitable for observing scaling behavior with respect to size (number of qubits).[29]

(2) Mermin–Bell (`mermin_bell_16`)

Mermin–Bell circuits evaluate quantities related to multipartite entanglement and (Mermin-type) Bell inequalities. Because they aim to evaluate global correlations, they are organized as benchmarks that have interaction patterns spanning the entire circuit.[29]

(3) TFIM Hamiltonian simulation (`tfim_hamiltonian_sim_16`, `tfim_hamiltonian_sim_32`, `tfim_hamiltonian_sim_64`)

TFIM (Transverse-Field Ising Model) Hamiltonian-simulation circuits approximately implement the time evolution of a physical model including interaction terms and transverse-field terms via Trotter decomposition. In general, they repeatedly apply nearest-neighbor interactions at each time step, and represent typical simulation circuits that exhibit both "locality" and "repetitiveness."[29] In this study, we use multiple sizes and also observe how trends change as the scale increases.

5.1.3 QFT

The QFT (Quantum Fourier Transform) is positioned as a foundational subroutine in algorithms such as quantum phase estimation and integer factorization. In general, because it contains many controlled phase rotations, interactions tend to appear across a wide range of qubit pairs as the circuit scale increases.[22] Due to this property, in contrast to circuits with strong locality, it tends to impose stringent conditions from the viewpoint of partitioning and placement. In this study, to observe scale-dependent trends, we use QFT circuits of the following sizes.

- `qft_16`
- `qft_32`
- `qft_64`

5.2 Partitioning Algorithms

This study compares multiple methods with different characteristics for quantum circuit partitioning under an assumed distributed execution setting. The partitioning problem considered in this study is formulated as an assignment problem on a structure in which qubits are vertices and two-qubit interactions (primarily CX) are represented as weighted edges (or hyperedges), and the goal is to assign the vertex set to multiple partitions. Methods proposed in prior work mainly aim to reduce interactions that cross partitions, thereby suppressing the required remote operations in distributed execution (measured as RCX in this study).[3, 5] Below, we explain how each method used for comparison (KL, FM, HQA) generally performs partitioning based on its optimization principle.

5.2.1 Kernighan–Lin (KL)

The Kernighan–Lin (KL) algorithm is a classical local-search method for graph bisection.[13] Given an initial partition, it iteratively performs vertex swaps to improve the partition by reducing the cut (the sum of edge weights between partitions). Typically, in each iteration it evaluates the "gain" of swaps, and applies a batch of swaps that maximizes the gain sequence, thereby searching for a local optimum. A key feature is that it improves the solution while maintaining a balance constraint on partition size (the number of vertices in each partition).

When handling more than two partitions as in this study, it is common to extend KL to multi-way partitioning by applying it recursively. That is, the whole set is first bisected, and then each subset is repeatedly bisected until the desired number of partitions is reached.

5.2.2 Fiduccia–Mattheyses (FM)

The Fiduccia–Mattheyses (FM) algorithm is a representative refinement method widely used particularly in the context of (hyper)graph partitioning.[8] While prioritizing moves that strongly improve the objective function (e.g., cut size or hypercut), it iteratively performs move operations that relocate vertices one by one to another partition. Typically, it locks a moved vertex at each move so that the same vertex is not moved multiple times within the same pass, thereby reducing computational cost. It also performs improvements while respecting balance constraints (upper bounds on partition sizes), which aligns well with resource constraints in distributed execution (the qubit capacity of each node).

In this study, we represent the interaction structure of a quantum circuit as a (hyper)graph, and position FM as a local-refinement method aimed at reducing interactions that cross partitions.

5.2.3 Hungarian Qubit Assignment (HQA)

Hungarian Qubit Assignment (HQA) is an approach that does not primarily view partitioning as graph-cut minimization, but instead frames it as an assignment problem: "how to assign qubits to multiple nodes." Typically, it aggregates the amount of interaction between qubits to define a cost (or gain), and optimizes "which qubits should be placed on which nodes (partitions)" based on that cost. A key feature is that it uses the Hungarian algorithm, which solves the linear assignment problem in polynomial time, as the core of assignment optimization.[7]

HQA implementation used in this study Our HQA does not adopt a **dynamic (re-assignment) scheme** that decomposes the circuit into time slices and sequentially updates assignments. That is, consistent with our comparison setting (static partitioning: fixing a single assignment map f for the entire circuit), we implement HQA as a static assignment method based on interaction information obtained from the whole circuit.

Procedure of static HQA Specifically, (i) we construct a weighted matrix W of two-qubit interactions from the entire circuit, (ii) extract a set of interaction pairs with large weights, (iii) solve a linear assignment that assigns the extracted pairs to "node slots" using the Hungarian method, (iv) reconstruct a qubit-to-node mapping that satisfies the capacity constraint from the assignment result. In addition, if necessary, we apply a small number of local-improvement steps. Thus, our HQA is positioned as a **static** partitioner based on "pair $\rightarrow$ node Hungarian + reconstruction."

5.2.4 Environment for RCX Counting

To compare RCX across partitioning methods under **identical conditions**, this study implements the partitioning algorithms and the RCX counter in-house and builds a unified evaluation pipeline. Specifically, each partitioning method outputs an assignment map $f : q \mapsto v$ that assigns each logical qubit q to a target node v, and RCX is computed for that assignment. Here, RCX is counted as the **number of CNOT (CX) gates**

executed across different nodes under the assignment f. That is, for each CX gate (q_i, q_j) in the circuit, we add 1 if $f(q_i) \neq f(q_j)$, and define the sum as RCX. With this definition, we can directly compare how RCX changes for the same circuit c when only the partitioning method m is varied.

As circuit-side preprocessing, in order not to impair comparability across partitioning methods, we **unified the representation of two-qubit gates**. Specifically, after removing analysis-irrelevant instructions such as barriers, we used the Qiskit transpiler to convert the circuit into a prescribed basis gate set so that the two-qubit gate type is unified to CX only. This prevents differences in two-qubit gate representations such as CZ/ECR from contaminating the comparison of "RCX (crossing count)," and guarantees that RCX differences originate from the partitioning (assignment) itself.

We set the target number of nodes to $k = 4$, and specified the capacity (the maximum number of qubits storable on each node), running all methods under the same constraints. The capacity was set so that it is approximately $n/4$ for an n-qubit circuit (if n is not divisible by 4, some nodes hold one additional qubit), and any assignment exceeding this upper bound is treated as an error, thereby **excluding extremely imbalanced assignments**. With this constraint, we avoid incomparable situations where RCX is accidentally reduced by overpacking into a single node, and we design the evaluation to measure method differences within the range of nearly balanced 4-way partitioning.

We chose $k = 4$ because (i) with $k = 2$ the problem is overly simplified to "cut or not cut," making it difficult for differences in method characteristics (local refinement, assignment variability, boundary formation in multi-way partitioning, etc.) to appear, whereas (ii) with $k \geq 8$ boundaries tend to increase, and for some circuits RCX tends to saturate so that all methods yield extremely large RCX, making interpretation of method differences difficult.

5.3 Evaluation Metric

This study uses only the Remote CNOT count (RCX) as the performance metric for quantum circuit partitioning under an assumed distributed quantum computing setting. This is because the goal of partitioning is to "reduce two-qubit interactions that cross nodes," and the number of such interactions can become a primary bottleneck in distributed execution. In principle, other metrics (e.g., graph-cut volume or load balance) could also be introduced, but the main focus of this study is to clarify the relationship between circuit structure and the relative suitability of partitioning methods. Therefore, we unify the metric to RCX and prioritize comparability.

5.3.1 RCX

RCX (Remote CNOT) is the number of "inter-node operations" that occur in executing a partitioned circuit when the two qubits involved in a CX gate are assigned to different partitions. In this study, based on the qubit assignment (partitioning) obtained as the partitioning result, we determine whether each CX gate in the circuit completes within a single partition or crosses partitions, and we count RCX by adding 1 when it crosses.

RCX is a proxy metric for communication cost. Depending on the actual distributed implementation (teleportation model, scheduling, fault-tolerant procedures, etc.), RCX may not coincide directly with the number of communications or execution time. However, because the purpose of this study is "relative comparison across methods" and "trend explanation based on circuit structure," we adopt RCX as a consistent criterion that does not depend on a specific implementation model.

5.4 Experimental Environment

5.4.1 Relative Evaluation via Copeland Score

In this study, we do not define performance solely by a single best point such as "which method minimized RCX," but instead treat performance as a **relative evaluation** that expresses how much each method dominates other methods on the same circuit. To summarize this relative evaluation into a single value, we introduce a metric based on the Copeland score used in social choice theory.

Consider a circuit c and a set of partitioning methods M to be compared. For each method $m \in M$, we perform pairwise comparisons against every other method $k \in M \setminus \{m\}$, and regard the method with smaller RCX on the same circuit as the winner. Specifically, we define

$$w(m,k;c) = \begin{cases} 1 & (\mathrm{RCX}(m;c) < \mathrm{RCX}(k;c)) \\ \frac{1}{2} & (\mathrm{RCX}(m;c) = \mathrm{RCX}(k;c)) \\ 0 & (\mathrm{RCX}(m;c) > \mathrm{RCX}(k;c)) \end{cases} \tag{5.1}$$

where, in the case of a tie (equal RCX), both sides receive 1/2 (a tie is treated as a 0.5 win).

Next, we define the Copeland score of method m on circuit c as

$$S_m(c) = \sum_{k \in M \setminus \{m\}} w(m,k;c) \tag{5.2}$$

$S_m(c)$ represents how many wins method m achieves against other methods on circuit c, and a larger value indicates stronger relative dominance.

Due to experimental constraints, for some circuits RCX values may not be available for some methods. In that case, comparisons involving missing methods are excluded from the pairwise comparison set. Furthermore, to preserve comparability of scores even when the number of comparable opponents differs across circuits, we also use a normalized score

$$\widetilde{S}_m(c) = \frac{S_m(c)}{|M_c| - 1} \tag{5.3}$$

where $M_c \subseteq M$ is the set of methods that are comparable on circuit c. This provides a common scale even when the number of comparable opponents differs.

Using the Copeland score has two advantages. First, it provides a stable summary of the entire dominance relation (ranking) without relying on point estimation of a single best method. Second, in the subsequent SHAP analysis, by predicting and decomposing this relative-evaluation score from structural indicators, we can explain, for each circuit, "which structural indicators contributed to forming the method ranking."

5.4.2 Contribution of Structural Indicators to Ranking

To extract, in an **interpretable form**, the relationship between the relative evaluation score $\tilde{S}_m(c)$ (normalized Copeland score) and the circuit structural indicators $\mathbf{x}(c)$, this study uses SHAP.

Positioning of SHAP (connecting predictive models and factor analysis) SHAP (SHapley Additive exPlanations) is a post-hoc explanation method that represents the output $f(\mathbf{x})$ of a trained predictor f as the sum of a **baseline value (typical/average output) and the contributions of individual features**.[17, 18] Here, machine learning is not used to estimate causal effects, but rather to **approximate a complex nonlinear function f and decompose/visualize which input features produced differences in the output**. Accordingly, our SHAP analysis describes how structural indicators relate to ranking formation (relative score differences) as a **contribution decomposition within the predictive model**, and it is not causal inference per se.[18]

How SHAP values are computed (average marginal contribution as Shapley values) Based on the Shapley value in cooperative game theory,[27] SHAP defines the contribution ϕ_i of feature i as the average, over all feature orderings, of the "change in the output (marginal contribution)" when that feature is added. Let the feature set be F, and consider a subset $S \subseteq F \setminus \{i\}$ excluding i. A typical definition is

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left(f_{S \cup \{i\}}(\mathbf{x}_{S \cup \{i\}}) - f_S(\mathbf{x}_S) \right),$$

[27, 17] where $f_S(\mathbf{x}_S)$ is often implemented as an expected output obtained by fixing only the features in S and marginalizing (integrating out) the remaining features with respect to some distribution.[17] With this definition, SHAP organizes the model output into an additive form

$$f(\mathbf{x}) = \phi_0 + \sum_{i=1}^{d} \phi_i$$

(additive feature attribution), and properties such as local accuracy are discussed.[17]

Why we use TreeExplainer (TreeSHAP) in this study Our difference model $f_{a,b}$ is trained as a tree-based ensemble regressor (e.g., random forests or gradient boosting), and SHAP values are computed using `shap.TreeExplainer`. TreeExplainer is based on the TreeSHAP algorithm and can compute SHAP values efficiently and (under assumptions) exactly for tree models.[16, 1] By exploiting tree structure, TreeSHAP reduces the naively exponential-time Shapley-value computation to polynomial time.[16] In implementation, it also allows choices regarding how to marginalize unobserved features, including whether to provide background data explicitly.[1]

Interpretation caveats (feature dependence and limits of causal interpretation) SHAP values depend on the distribution used when marginalizing unobserved features. In particular, when features are dependent (correlated), interpretation can change depending on whether a conditional distribution is used or the features are treated interventionally.[1, 12] Therefore, in this study we use SHAP as a "contribution decomposition" to explain ranking formation (score differences), and we do not claim that an indicator *causes* the ranking.[12]

What we aim to explain is not the score of a single method m itself, but the **score differences that produce the relative ordering (ranking) among methods**. Accordingly, in the current implementation, rather than training a model f_m for each method m, we train a **difference model** that directly predicts score differences for each method pair.

For methods $a, b \in \mathcal{M}$, we define the score difference on circuit c as

$$y_{a,b}(c) \;=\; \tilde{S}_a(c) - \tilde{S}_b(c) \tag{5.4}$$

and train a predictor

$$\hat{y}_{a,b}(c) \;=\; f_{a,b}(\mathbf{x}(c)) \tag{5.5}$$

SHAP decomposes this difference prediction $\hat{y}_{a,b}(c)$ into an additive form of "baseline value + feature contributions," namely

$$\hat{y}_{a,b}(c) \;=\; \phi_{0,(a,b)} + \sum_{i=1}^{d} \phi_{i,(a,b)}(c) \tag{5.6}$$

where d is the number of structural indicators, $\phi_{0,(a,b)}$ is the baseline value (expected output) of the difference model $f_{a,b}$, and $\phi_{i,(a,b)}(c)$ represents how much structural indicator i on circuit c contributes to $\hat{y}_{a,b}(c)$ (i.e., "in the direction where a is more favorable than b," or the reverse).

Using the contributions $\phi_{i,(a,b)}(c)$ obtained by Eq. (5.6), we explain, for each circuit, "to what extent circuit structure produced relative differences among methods." In the next subsection, we aggregate these contributions and define structural indicators that are **strongly related** to ranking formation itself.

5.4.3 Degree of Involvement in Ranking Formation (Definition Based on Difference Models)

What we want to know is not "which indicators boost a particular method," but **which indicators separate the rankings (relative superiority) among methods**. Under the current design, method rankings are determined by the score differences $y_{a,b}(c)$, so it is natural to define the involvement measure by aggregating "how large the SHAP contributions of the difference models are."

For a circuit c, we define the set of method pairs to be evaluated as

$$\mathcal{P}(c) \;=\; \{(a,b) \mid a, b \in \mathcal{M},\, a < b,\, \tilde{S}_a(c), \tilde{S}_b(c) \text{ are both defined}\} \tag{5.7}$$

That is, we aggregate only method pairs without missing values on circuit c.

Next, we define the involvement of structural indicator i in ranking formation on circuit c (local Rank Influence) as

$$I_i(c) \;=\; \frac{1}{|\mathcal{P}(c)|} \sum_{(a,b)\in\mathcal{P}(c)} \left|\phi_{i,(a,b)}(c)\right| \tag{5.8}$$

A larger $I_i(c)$ indicates that structural indicator i on circuit c acts strongly in the direction of **separating** score differences (relative superiority) among methods. Here we take the absolute value because the essence of ranking formation is not "which method is favorable (sign)," but **how widely the differences spread (separation strength).**

Furthermore, we define global Rank Influence averaged over the circuit set $\mathcal{C}$ as

$$I_i \;=\; \frac{1}{|\mathcal{C}|} \sum_{c\in\mathcal{C}} I_i(c) \tag{5.9}$$

This allows us to extract, with priorities, indicators that tend to produce ranking differences (method differences) even when many structural indicators exist.

Note that the baseline value $\phi_{0,(a,b)}$ of the difference model is a constant for each pair (a,b) and is not an explanatory target for ranking separation attributable to circuit structure $\mathbf{x}(c)$. Therefore, in our involvement definition (5.8), we aggregate **only the feature contributions $\phi_{i,(a,b)}(c)$.**

5.4.4 Stability Check

The Rank Influence in this study is an estimate based on a finite benchmark circuit set. Therefore, the importance ranking can vary depending on the choice of circuits (sampling fluctuations), and the resulting ranking should not be over-generalized. Accordingly, we evaluate the reproducibility (stability) of Rank Influence estimation via bootstrap resampling of the circuit set.

Let the circuit set be C with $|C| = N$. In bootstrap resampling, we generate resampled sets $C^{(b)}$ of the same size N by sampling with replacement from C, and repeat this for iterations $b = 1, 2, \ldots$. At each iteration b, we compute Copeland scores from the RCX results on $C^{(b)}$. Then, using method-pair regression models whose target variable is the score difference and SHAP decomposition, we obtain circuit-wise Rank Influence values $I_i(c)$ (defined in the previous subsection). Averaging the circuit-wise values over the resampled set yields the global importance at iteration b:

$$I_i^{(b)} = \frac{1}{|C^{(b)}|} \sum_{c\in C^{(b)}} I_i(c) \tag{5.10}$$

Through this procedure, we obtain a bootstrap distribution $\{I_i^{(b)}\}$ for each feature i.

In this study, we define the importance ranking by the mean of this distribution:

$$\mathrm{mean}_i = \frac{1}{B} \sum_{b=1}^{B} I_i^{(b)} \tag{5.11}$$

and also report the standard deviation of the distribution:

$$\mathrm{std}_i = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\left(I_i^{(b)} - \mathrm{mean}_i\right)^2} \tag{5.12}$$

Furthermore, using the percentile method, we provide a 95% confidence interval (CI) as

$$\mathrm{CI}_{95\%}(I_i) = \left[Q_{0.025}\left(\{I_i^{(b)}\}_{b=1}^{B}\right),\ Q_{0.975}\left(\{I_i^{(b)}\}_{b=1}^{B}\right)\right] \tag{5.13}$$

where $Q_p(\cdot)$ denotes the empirical p-quantile of the distribution. This CI represents "the variability of importance estimates under fluctuations of the circuit set," and is conceptually different from a predictive error interval.

In addition, as a concise indicator of ranking reproducibility, we introduce the frequency of appearing within the top K ranks (top-K frequency). Consider the event that feature i is within the top K of the importance ranking at iteration b. We define

$$f_i^{(K)} = \frac{1}{B}\sum_{b=1}^{B}\mathbf{1}\left(\mathrm{rank}_i^{(b)} \le K\right) \tag{5.14}$$

A feature with a large $f_i^{(K)}$ tends to appear in the top ranks even under resampling, and its importance ranking is interpreted as relatively stable.

Choosing the number of bootstrap iterations B (convergence check) Rather than fixing the bootstrap count B to a single value, this study **sets an upper limit** $B_{\max}$ and increases B stepwise according to an incremental schedule (e.g., $250, 500, 1000, \dots$). At each stage, we evaluate changes in summary statistics (mean_i, $\mathrm{CI}_{95\%}$, $f_i^{(K)}$) and the agreement of the top-K feature sets, and stop when a prescribed convergence condition is satisfied (or run up to the upper limit $B_{\max}$ if it is not satisfied). In this study, we adopted a policy of using a 2% threshold for the agreement of the top-10 set and for changes in summary statistics. In the runs reported in this thesis, we performed convergence checking with an upper limit of $B_{\max} = 100{,}000$, and adopted the value of B at the stage where the convergence condition was satisfied.

5.4.5 Why We Introduce Confidence Intervals (CI)

In this study, we do not present the importance of each structural indicator only as a single point estimate (mean_i), but also report CIs based on the bootstrap distribution to make uncertainty explicit. This is because importance estimation can fluctuate depending on a finite circuit set, and we need to rule out the possibility that an observation that "seems important" is due to sampling bias. The purpose of introducing CIs is threefold:

1. To confirm whether the importance consistently takes a positive magnitude even under fluctuations of the circuit set.
2. To visualize variability (stability) of the estimates that is not visible from a point estimate alone.
3. To strengthen the objectivity of importance interpretation and clarify reservations about the generalizability of the results.

Chapter 6

Partitioning Performance Results

This chapter organizes the per-circuit RCX results for each partitioning method and then summarizes, via SHAP analysis, which circuit-structure indicators tend to induce performance differences across methods. Because the definitions and procedures of the Copeland score, Rank Influence, and the bootstrap-based stability evaluation (mean/CI/top-K frequency) used in the SHAP analysis were described in the previous chapter, this chapter reports only the results. This is because, for some circuits, applying the algorithm exceeded practical computational limits and became infeasible, and therefore no results were obtained.

6.1 RCX results by circuit

Table 6.1 shows the RCX (Remote CNOT) counts for each method on each circuit. This table indicates that the relative superiority of methods can change depending on the circuit, and that the RCX scale differs substantially across circuit families.

Table 6.1: Per-circuit RCX counts

Circuit	fm	kl	hqa
bigadder_n16	0	0	0
dnn_n16	96	98	96
ghz_16	3	3	3
ghz_32	3	3	3
ghz_64	3	4	3
mermin_bell_16	114	116	116
multiplier_n15	74	89	135
multiplier_n45	582	631	1549
qft_16	192	192	192
qft_32	768	768	768
qft_64	3072	3072	3072
square_root_n18	326	383	511
swap_test_n25	36	36	36
tfim_hamiltonian_sim_16	6	6	6
tfim_hamiltonian_sim_32	6	6	6
tfim_hamiltonian_sim_64	6	8	6

6.2 SHAP Analysis Results

In this section, we use Rank Influence, defined in the previous chapter, to summarize which circuit-structure indicators tend to induce performance differences across methods (Copeland score differences). The feature ranking is based on bootstrap aggregation, and the number of bootstrap iterations adopted after the convergence check is $B = 100000$ (procedure: Chapter 5.4.4).

6.2.1 Global feature ranking (bootstrap aggregation)

Table 6.2 reports summary statistics (mean, std, 95% CI, and top-10 frequency) of the global Rank Influence over all features (26 in total). The top five features by mean are Graph density (D), Edge duplication ratio (Rdup), Entanglement variance (EntVar), Edge lifetime avg, and Spectral recursive lambda2 avg ($\bar{\lambda}_2$(rec)). These are extracted as features that, on average, are likely to produce differences across methods (rank differences).

(1) Meaning of each column (variable) in Table 6.2

Table 6.2 shows the bootstrap-aggregated results of the global Rank Influence I_i for each feature i. The meaning of each column is as follows.

- `mean_I`: the mean of $I_i^{(b)}$ obtained over B bootstrap iterations. The feature ranking in this study is defined by this value.
- `std_I`: the standard deviation of $\{I_i^{(b)}\}_{b=1}^{B}$.
- `ci_low`, `ci_high`: the 95% confidence interval based on the percentile method (2.5%, 97.5%) of $\{I_i^{(b)}\}_{b=1}^{B}$.
- `top10_freq`: the proportion of iterations in which the feature appears within the top 10 in the importance ranking constructed in each iteration.

`top10_freq` is defined as follows.

$$\text{top10\_freq}(i) = \frac{1}{B}\sum_{b=1}^{B} \mathbf{1}\left[\text{rank}_i^{(b)} \le 10\right], \tag{6.1}$$

i.e., " the number of times it falls within the top 10 divided by B. " Because the rank $\text{rank}_i^{(b)}$ is computed by `rank(ascending=False, method='min')`, if ties occur, the tied group shares the minimum rank.

Table 6.2: All features (25 total) for Rank Influence (bootstrap aggregation). This table quantifies, over the entire circuit set, how strongly each circuit-structure indicator tends to affect the pairwise " win/loss " relationships (relative superiority) among partitioning methods, and lists the indicators in descending order of the mean. Specifically, we train a model that predicts the relative superiority of methods from circuit-structure indicators, and define I_i as the magnitude (absolute amount, not the sign) of how much each feature changes the prediction. We resample the circuit set via bootstrap and summarize the distribution of $I_i^{(b)}$ obtained in each iteration, reporting the mean (descending; top five in bold), standard deviation, the 95% percentile CI (`ci_low`, `ci_high`), and the top-10 appearance frequency (Top10 rate) ($B = 100000$). A larger mean indicates that the feature is, on average, more likely to produce method differences, and a larger Top10 frequency indicates that this tendency is stable under resampling.

Rank	Feature	mean	std	ci_low	ci_high	top10_freq
1	**Parallelism avg (average parallelism, Pavg)**	**0.115459**	0.126592	0	0.344274	0.74457
2	**Entanglement variance (entanglement variance, EntVar)**	**0.092783**	0.090934	0	0.317980	0.90076
3	**M (total_weight) (total number of 2-qubit gates)**	**0.063474**	0.044521	0.000650	0.168094	0.94776
4	**Bridge centrality concentration (bridge centrality, Bk)**	**0.054914**	0.073977	0	0.246889	0.68981
5	**Graph density (density of 2-qubit gate pairs, D)**	**0.053556**	0.069434	0	0.243889	0.72139
6	m (num_edges) (number of distinct pairs used)	0.052601	0.073369	0	0.267420	0.74660
7	Edge duplication ratio (duplication, Rdup)	0.051027	0.075225	0	0.259870	0.76726
8	n (num_vertices) (number of qubits)	0.026906	0.039135	0	0.150408	0.71480
9	Cluster size CV (cluster size coefficient of variation, CVC)	0.022900	0.059866	0	0.218297	0.18118
10	Hyperedge size avg (average hyperedge size, $\|H_t\|$)	0.020526	0.029761	0	0.083176	0.55090
11	Spectral recursive lambda2 avg (recursive spectral mean, $\bar{\lambda}_2$(rec))	0.018453	0.032470	0	0.104039	0.49894
12	Modularity (modularity, Qmod)	0.014748	0.020692	0	0.073404	0.55439
13	Edge lifetime avg (edge lifetime mean, Lavg)	0.009410	0.033303	0	0.150458	0.16873
14	Gate density (gate density, GateDensity)	0.009209	0.015861	0	0.053745	0.33727
15	Cluster size balance (cluster size balance, Bsize)	0.007457	0.011180	0	0.039371	0.35733
16	Hot-Qubit concentration (hot-qubit concentration, Hmax)	0.005956	0.010230	0	0.037612	0.35942
17	Cluster boundary ratio avg (average boundary ratio, Savg)	0.004149	0.014188	0	0.057521	0.10588
18	Entanglement time variance (entanglement time variance, Var_ent)	0.003881	0.009793	0	0.033445	0.14485
19	Parallelism effective (effective parallelism, Peff)	0.003036	0.009926	0	0.033925	0.10642
20	Parallelism max (maximum parallelism, Pmax)	0.002810	0.008601	0	0.026796	0.09534
21	Cluster overload ratio (hot-cluster overload ratio, Roverload)	0.001655	0.006084	0	0.024299	0.06338
22	Hyperedge overcapacity ratio (hyperedge overcapacity ratio, Rhyper)	0.001561	0.007355	0	0.019906	0.05226
23	Long edge ratio (long-lifetime hot-edge ratio, Rlong)	0.001168	0.004418	0	0.012124	0.05015

Rank	Feature	mean	std	ci_low	ci_high	top10_freq
24	Spectral P indicator (P-partition spectral indicator, $S_{\text{spec}}(P)$)	0.001137	0.003889	0	0.010602	0.04593
25	Qnode (node capacity, $Q_{\text{node}} = n/P$)	0	0	0	0	0

6.2.2 Per-circuit contributions (local Rank Influence)

While the global ranking summarizes the entire circuit set, the dominant features can differ from circuit to circuit. Table 6.3 lists, for each circuit, the features in descending order of local Rank Influence. This table confirms that the “factors that produce method differences” can switch depending on the circuit.

Table 6.3: All features of local Rank Influence for each circuit (symbol, measured value, and $I_i(c)$). For each circuit c, local Rank Influence $I_i(c)$ quantifies how strongly the structural indicator i contributes to separating (amplifying) the score differences across methods, and the features are shown in descending order. Value denotes the measured value of the structural indicator for that circuit, $x_i(c)$, and $I_i(c) = \frac{1}{|P(c)|}\sum_{(a,b)\in P(c)} |\phi_{i,(a,b)}(c)|$ represents the separation strength (unsigned).

Circuit	Rank	Feature	Value	$I_i(c)$	Circuit	Rank	Feature	Value	$I_i(c)$
bigadder_n16	1	Pavg	8.0000	0.3748	dnn_n16	1	Pavg	0.6894	0.2513
	2	CVC	0.0000	0.1165		2	M	384.0000	0.1011
	3	M	8.0000	0.0679		3	CVC	0.0000	0.0983
	4	m	8.0000	0.0223		4	EntVar	0.0000	0.0560
	5	Qmod	0.8750	0.0220		5	m	16.0000	0.0353
	6	EntVar	0.0000	0.0169		6	Bk	0.0625	0.0349
	7	Bk	0.0000	0.0167		7	Bsize	1.0000	0.0161
	8	$\|H_t\|$	16.0000	0.0116		8	D	0.1333	0.0137
	9	n	16.0000	0.0081		9	Rdup	24.0000	0.0123
	10	Savg	0.0000	0.0065		10	Savg	0.4000	0.0106
	11	$\bar{\lambda}_2$(rec)	2.0000	0.0064		11	$\|H_t\|$	13.7594	0.0099
	12	GateDensity	1.0000	0.0062		12	GateDensity	0.8600	0.0073
	13	Hmax	0.0625	0.0052		13	n	16.0000	0.0071
	14	D	0.0667	0.0049		14	Lavg	407.0000	0.0061
	15	Bsize	0.5000	0.0037		15	$\bar{\lambda}_2$(rec)	14.0589	0.0044
	16	Rdup	1.0000	0.0034		16	Qmod	0.5000	0.0035
	17	Peff	8.0000	0.0031		17	Peff	8.0000	0.0035
	18	Lavg	0.0000	0.0019		18	Hmax	0.0625	0.0031
	19	Rhyper	0.7500	0.0006		19	Rhyper	0.6800	0.0006
	20	$S_{\text{spec}}(P)$	0.0000	0.0005		20	Var_{ent}	5.0400	0.0003
	21	Var_{ent}	0.0000	0.0003		21	$S_{\text{spec}}(P)$	7.1221	0.0003
	22	Pmax	8.0000	0.0000		22	Pmax	8.0000	0.0000
	23	Rlong	0.0000	0.0000		23	Rlong	1.0000	0.0000
	24	Qnode	4	0.0000		24	Qnode	4	0.0000
	25	Roverload	0.0000	0.0000		25	Roverload	0.0000	0.0000
ghz_16	1	Pavg	0.8333	0.1841	ghz_32	1	Pavg	0.9118	0.1864
	2	M	15.0000	0.1029		2	M	31.0000	0.1181
	3	CVC	0.0000	0.0937		3	CVC	0.3536	0.0860
	4	EntVar	0.0632	0.0294		4	Bk	0.1448	0.0574
	5	Bk	0.1000	0.0253		5	EntVar	0.0330	0.0297
	6	m	15.0000	0.0202		6	$\bar{\lambda}_2$(rec)	0.1522	0.0201
	7	$\|H_t\|$	1.8333	0.0156		7	$\|H_t\|$	1.9118	0.0165
	8	n	16.0000	0.0145		8	m	31.0000	0.0096
	9	Bsize	1.0000	0.0112		9	D	0.0625	0.0076
	10	Savg	0.3250	0.0060		10	Savg	0.2625	0.0064
	11	Hmax	0.0667	0.0049		11	n	32.0000	0.0064
	12	$\bar{\lambda}_2$(rec)	0.5858	0.0041		12	Hmax	0.0323	0.0053
	13	D	0.1250	0.0031		13	Qmod	0.6550	0.0038
	14	Lavg	0.0000	0.0030		14	Bsize	0.6667	0.0033
	15	Qmod	0.5489	0.0028		15	Lavg	0.0000	0.0029
	16	GateDensity	0.1146	0.0020		16	Rhyper	0.0000	0.0020
	17	Rdup	1.0000	0.0017		17	GateDensity	0.0597	0.0018
	18	$S_{\text{spec}}(P)$	0.1759	0.0012		18	Rdup	1.0000	0.0017
	19	Peff	1.0000	0.0011		19	$S_{\text{spec}}(P)$	0.0447	0.0012
	20	Rhyper	0.0000	0.0006		20	Peff	1.0000	0.0009
	21	Var_{ent}	0.1389	0.0005		21	Var_{ent}	0.0804	0.0003
	22	Pmax	1.0000	0.0000		22	Pmax	1.0000	0.0000
	23	Rlong	0.0000	0.0000		23	Rlong	0.0000	0.0000
	24	Qnode	4	0.0000		24	Qnode	8	0.0000
	25	Roverload	0.0000	0.0000		25	Roverload	0.0000	0.0000
ghz_64	1	Pavg	0.9545	0.2272	mermin_bell_16	1	Pavg	1.0059	0.4002
	2	CVC	0.0000	0.1008		2	CVC	0.3536	0.1106
	3	EntVar	0.0168	0.0602		3	M	171.0000	0.0348
	4	M	63.0000	0.0431		4	Savg	0.7722	0.0191
	5	Bk	0.1425	0.0392		5	Bk	0.0913	0.0175
	6	n	64.0000	0.0357		6	EntVar	0.2964	0.0165

Circuit	Rank	Feature	Value	$I_i(c)$	Circuit	Rank	Feature	Value	$I_i(c)$
	7	m	63.0000	0.0319		7	$\|H_t\|$	6.3882	0.0150
	8	Qmod	0.7638	0.0291		8	D	1.0000	0.0140
	9	$\bar{\lambda}_2$(rec)	0.0384	0.0206		9	Qmod	0.0837	0.0094
	10	$\|H_t\|$	1.9545	0.0162		10	m	120.0000	0.0083
	11	Hmax	0.0159	0.0152		11	$\bar{\lambda}_2$(rec)	6.1230	0.0066
	12	D	0.0312	0.0137		12	Lavg	24.3000	0.0065
	13	Savg	0.1979	0.0110		13	n	16.0000	0.0059
	14	Bsize	0.5000	0.0039		14	Hmax	0.0906	0.0046
	15	Lavg	0.0000	0.0015		15	Rdup	1.4250	0.0045
	16	Rdup	1.0000	0.0015		16	Bsize	0.7500	0.0035
	17	GateDensity	0.0305	0.0014		17	GateDensity	0.3993	0.0025
	18	$S_{\text{spec}}(P)$	0.0112	0.0012		18	Var_{ent}	0.5470	0.0013
	19	Peff	1.0000	0.0008		19	Peff	1.2761	0.0011
	20	Var_{ent}	0.0434	0.0006		20	$S_{\text{spec}}(P)$	17.6863	0.0006
	21	Rhyper	0.0000	0.0006		21	Rhyper	0.2994	0.0006
	22	Pmax	1.0000	0.0000		22	Pmax	4.0000	0.0000
	23	Rlong	0.0000	0.0000		23	Rlong	0.5029	0.0000
	24	Qnode	16	0.0000		24	Qnode	4	0.0000
	25	Roverload	0.0000	0.0000		25	Roverload	0.4762	0.0000
`multiplier_n15`	1	CVC	0.4320	0.4022	`multiplier_n45`	1	CVC	0.7611	0.4051
	2	Pavg	0.7760	0.1574		2	Pavg	0.8758	0.1520
	3	M	246.0000	0.0786		3	M	2574.0000	0.0640
	4	Bk	0.2000	0.0558		4	EntVar	0.2759	0.0431
	5	EntVar	0.5869	0.0478		5	Bk	0.3937	0.0427
	6	D	0.2857	0.0184		6	D	0.1788	0.0205
	7	$\|H_t\|$	3.0410	0.0122		7	n	45.0000	0.0125
	8	m	30.0000	0.0105		8	Savg	0.4313	0.0088
	9	Rdup	8.2000	0.0098		9	Bsize	0.4074	0.0059
	10	Savg	0.3524	0.0090		10	Rdup	14.5424	0.0044
	11	Bsize	0.5333	0.0067		11	$\|H_t\|$	3.4253	0.0036
	12	n	15.0000	0.0064		12	$\bar{\lambda}_2$(rec)	5.8449	0.0036
	13	Lavg	153.1333	0.0033		13	Lavg	877.3277	0.0034
	14	$\bar{\lambda}_2$(rec)	11.2954	0.0028		14	m	177.0000	0.0032
	15	Hmax	0.1524	0.0027		15	GateDensity	0.0761	0.0026
	16	GateDensity	0.2027	0.0023		16	Hmax	0.0437	0.0024
	17	Rhyper	0.0739	0.0020		17	Rhyper	0.0030	0.0020
	18	Var_{ent}	0.4956	0.0013		18	Peff	1.3448	0.0012
	19	Peff	1.2059	0.0009		19	$S_{\text{spec}}(P)$	7.4427	0.0007
	20	Qmod	0.4357	0.0007		20	Qmod	0.5811	0.0003
	21	$S_{\text{spec}}(P)$	5.8118	0.0006		21	Var_{ent}	0.7947	0.0002
	22	Pmax	5.0000	0.0000		22	Pmax	17.0000	0.0000
	23	Rlong	1.0000	0.0000		23	Rlong	1.0000	0.0000
	24	Qnode	4	0.0000		24	Qnode	12	0.0000
	25	Roverload	0.7031	0.0000		25	Roverload	0.1336	0.0000
`qft_16`	1	Pavg	2.0952	0.4348	`qft_32`	1	Pavg	4.0945	0.4384
	2	CVC	0.0000	0.1170		2	CVC	0.0000	0.1163
	3	M	264.0000	0.0418		3	M	1040.0000	0.0431
	4	Savg	0.9180	0.0234		4	Savg	0.9600	0.0234
	5	EntVar	0.0000	0.0228		5	EntVar	0.0000	0.0215
	6	Bk	0.0625	0.0187		6	Bk	0.0938	0.0147
	7	D	1.0000	0.0144		7	D	1.0000	0.0147
	8	Qmod	0.0265	0.0120		8	Qmod	0.0144	0.0120
	9	$\|H_t\|$	7.4286	0.0115		9	$\|H_t\|$	14.4252	0.0106
	10	m	120.0000	0.0098		10	m	496.0000	0.0099
	11	GateDensity	0.4643	0.0065		11	GateDensity	0.4508	0.0065
	12	$\bar{\lambda}_2$(rec)	8.0000	0.0054		12	$\bar{\lambda}_2$(rec)	16.0000	0.0061
	13	n	16.0000	0.0053		13	Bsize	0.2500	0.0056
	14	Rdup	2.2000	0.0049		14	n	32.0000	0.0033
	15	Hmax	0.0625	0.0041		15	Peff	6.6667	0.0031
	16	Bsize	0.5000	0.0033		16	Lavg	5.1290	0.0022
	17	Lavg	5.2667	0.0023		17	Hmax	0.0312	0.0019
	18	Peff	3.4737	0.0012		18	Rdup	2.0968	0.0013
	19	Rhyper	0.3655	0.0006		19	Rhyper	0.3520	0.0006
	20	Var_{ent}	5.8957	0.0004		20	Var_{ent}	24.9360	0.0005
	21	$S_{\text{spec}}(P)$	32.0000	0.0003		21	$S_{\text{spec}}(P)$	64.0000	0.0003
	22	Pmax	8.0000	0.0000		22	Pmax	16.0000	0.0000
	23	Rlong	1.0000	0.0000		23	Rlong	1.0000	0.0000
	24	Qnode	4	0.0000		24	Qnode	8	0.0000
	25	Roverload	0.0000	0.0000		25	Roverload	0.0000	0.0000
`qft_64`	1	Pavg	8.0941	0.4424	`square_root_n18`	1	CVC	0.4082	0.4054
	2	CVC	0.0000	0.1179		2	Pavg	0.5779	0.1699
	3	M	4128.0000	0.0358		3	M	898.0000	0.0747
	4	Savg	0.9802	0.0234		4	Bk	0.1468	0.0519
	5	EntVar	0.0000	0.0212		5	EntVar	0.6173	0.0490
	6	Bk	0.0938	0.0146		6	D	0.2353	0.0206
	7	n	64.0000	0.0146		7	$\|H_t\|$	2.4929	0.0129
	8	D	1.0000	0.0142		8	m	36.0000	0.0106
	9	Qmod	0.0075	0.0121		9	Savg	0.3732	0.0088
	10	Hmax	0.0156	0.0119		10	n	18.0000	0.0069
	11	$\|H_t\|$	28.4235	0.0107		11	Bsize	0.4444	0.0057
	12	m	2016.0000	0.0095		12	$\bar{\lambda}_2$(rec)	23.9210	0.0035
	13	$\bar{\lambda}_2$(rec)	32.0000	0.0064		13	Rdup	24.9444	0.0034
	14	Bsize	0.1250	0.0059		14	Lavg	1372.8611	0.0031

Circuit	Rank	Feature	Value	$I_i(c)$	Circuit	Rank	Feature	Value	$I_i(c)$
	15	Lavg	5.0635	0.0036		15	GateDensity	0.1385	0.0028
	16	Peff	13.0633	0.0030		16	Hmax	0.1158	0.0025
	17	GateDensity	0.4441	0.0026		17	Rhyper	0.0079	0.0020
	18	Rdup	2.0476	0.0020		18	Qmod	0.3567	0.0014
	19	$S_{\text{spec}}(P)$	128.0000	0.0008		19	Peff	1.0754	0.0012
	20	Rhyper	0.3461	0.0006		20	Var_{ent}	0.3302	0.0006
	21	Var_{ent}	103.0186	0.0004		21	$S_{\text{spec}}(P)$	16.3374	0.0006
	22	Pmax	32.0000	0.0000		22	Pmax	3.0000	0.0000
	23	Rlong	1.0000	0.0000		23	Rlong	0.9989	0.0000
	24	Qnode	16	0.0000		24	Qnode	5	0.0000
	25	Roverload	0.0000	0.0000		25	Roverload	0.9021	0.0000
`swap_test_n25`	1	Pavg	0.8727	0.1804	`tfim_hamiltonian_sim_16`	1	Pavg	0.5769	0.2042
	2	CVC	0.3795	0.0849		2	M	30.0000	0.1029
	3	Bk	1.0000	0.0559		3	CVC	0.0000	0.0942
	4	EntVar	0.2974	0.0427		4	EntVar	0.1300	0.0325
	5	M	96.0000	0.0244		5	Bk	0.1000	0.0208
	6	Savg	0.5429	0.0192		6	m	15.0000	0.0201
	7	$\lvert H_t\rvert$	4.2000	0.0186		7	n	16.0000	0.0146
	8	n	25.0000	0.0157		8	Bsize	1.0000	0.0128
	9	D	0.1200	0.0147		9	$\lvert H_t\rvert$	3.5962	0.0070
	10	m	36.0000	0.0096		10	$\bar{\lambda}_2$(rec)	1.1716	0.0062
	11	Bsize	0.3636	0.0072		11	Savg	0.3250	0.0060
	12	Lavg	21.3333	0.0053		12	Hmax	0.0667	0.0045
	13	Hmax	0.2500	0.0048		13	Lavg	2.0000	0.0033
	14	$\bar{\lambda}_2$(rec)	0.5000	0.0034		14	GateDensity	0.2248	0.0032
	15	Rdup	2.6667	0.0028		15	Qmod	0.5489	0.0030
	16	Qmod	0.4036	0.0026		16	D	0.1250	0.0023
	17	GateDensity	0.1680	0.0021		17	Peff	1.0000	0.0012
	18	Rhyper	0.0648	0.0020		18	Rdup	2.0000	0.0012
	19	Var_{ent}	2.7111	0.0012		19	$S_{\text{spec}}(P)$	0.3518	0.0008
	20	Peff	1.5238	0.0010		20	Var_{ent}	0.2441	0.0006
	21	$S_{\text{spec}}(P)$	2.0000	0.0007		21	Rhyper	0.1010	0.0006
	22	Pmax	12.0000	0.0000		22	Pmax	1.0000	0.0000
	23	Rlong	1.0000	0.0000		23	Rlong	1.0000	0.0000
	24	Qnode	7	0.0000		24	Qnode	4	0.0000
	25	Roverload	0.0000	0.0000		25	Roverload	0.0000	0.0000
`tfim_hamiltonian_sim_32`	1	Pavg	0.6200	0.2009	`tfim_hamiltonian_sim_64`	1	Pavg	0.6429	0.2470
	2	M	62.0000	0.0851		2	CVC	0.0000	0.1008
	3	CVC	0.3536	0.0840		3	M	126.0000	0.0824
	4	Bk	0.1448	0.0502		4	EntVar	0.0339	0.0436
	5	EntVar	0.0669	0.0263		5	Bk	0.1425	0.0314
	6	$\lvert H_t\rvert$	3.7900	0.0131		6	m	63.0000	0.0309
	7	m	31.0000	0.0093		7	Qmod	0.7638	0.0278
	8	D	0.0625	0.0082		8	n	64.0000	0.0227
	9	Savg	0.2625	0.0069		9	$\bar{\lambda}_2$(rec)	0.0769	0.0175
	10	n	32.0000	0.0058		10	Hmax	0.0159	0.0137
	11	Bsize	0.6667	0.0045		11	D	0.0312	0.0137
	12	Hmax	0.0323	0.0044		12	Savg	0.1979	0.0110
	13	Qmod	0.6550	0.0034		13	$\lvert H_t\rvert$	3.8929	0.0057
	14	$\bar{\lambda}_2$(rec)	0.3045	0.0034		14	Bsize	0.5000	0.0028
	15	Lavg	2.0000	0.0028		15	Lavg	2.0000	0.0016
	16	Rhyper	0.0525	0.0020		16	Rdup	2.0000	0.0015
	17	GateDensity	0.1184	0.0020		17	GateDensity	0.0608	0.0014
	18	Rdup	2.0000	0.0012		18	$S_{\text{spec}}(P)$	0.0225	0.0012
	19	$S_{\text{spec}}(P)$	0.0895	0.0012		19	Peff	1.0000	0.0007
	20	Peff	1.0000	0.0008		20	Var_{ent}	0.2296	0.0007
	21	Var_{ent}	0.2356	0.0006		21	Rhyper	0.0268	0.0006
	22	Pmax	1.0000	0.0000		22	Pmax	1.0000	0.0000
	23	Rlong	1.0000	0.0000		23	Rlong	1.0000	0.0000
	24	Qnode	8	0.0000		24	Qnode	16	0.0000
	25	Roverload	0.0000	0.0000		25	Roverload	0.0000	0.0000

6.3 Discussion

First, the introductory paragraph for the discussion goes here.

Classification of indicator groups The circuit-structure indicators used in this study can be broadly classified into the following groups.

- **Density-related (D, GateDensity, M, m, $|H_t|$):** These include sparsity/density (D), the total number of two-qubit gates (M), the number of distinct interacting pairs (m), and the hyperedge size $|H_t|$, i.e., the size of the set of two-qubit gates that appear simultaneously at time step t. Together, they describe how dense, how large-scale, and how diverse the two-qubit interactions are.

- **Bias-related (EntVar, Rdup, Hmax, Var_ent):** These include the variance of interaction counts per qubit (EntVar), the strength of repeated use of the same pair (Rdup), the concentration on hot qubits (Hmax), and the temporal variance of interactions (Var_ent), capturing concentration and repetition in interactions.
- **Parallelism-related (Pavg, Pmax, Peff):** These include average parallelism (Pavg), maximum parallelism (Pmax), and effective parallelism (Peff), describing how many "two-qubit gates that do not share qubits" can be executed simultaneously in the same layer (where a layer is defined as the set of gates assigned the same layer index t by our greedy/ASAP layering).
- **Spectral / partitionability-related ($S_{\text{spec}}(P)$, $\bar{\lambda}_2(\text{rec})$, Qmod, Bk):** This group reflects the existence of natural cut boundaries and the strength of community structure, including algebraic connectivity under recursive subgraph extraction ($\bar{\lambda}_2(\text{rec})$) and modularity (Qmod).
- **Balance / capacity-related (Bsize, CVC, Savg, Roverload, Rhyper, Q_{node}):** This group includes cluster-size imbalance (Bsize, CVC), boundary-edge ratio (Savg), hot-cluster overload (Roverload), over-capacity hyperedge ratio (Rhyper), and node-capacity-related quantities (Q_{node}).

6.3.1 Circuit structures to inspect for partitioning as suggested by the global ranking, and contributions of this study

In quantum circuit partitioning, when a circuit is assigned to multiple nodes, **two-qubit gates acting on qubit pairs assigned to different nodes (remote CNOTs)** often dominate the communication resources. However, in prior work, what to focus on—which circuits tend to increase remote CNOTs and when method differences tend to emerge—has often been discussed in terms of circuit-family names or experience-based heuristics, and a *common viewpoint for mechanically diagnosing this directly from the circuit itself* has been lacking.

To address this, this study defines **circuit-structure indicators** computable from the circuit, in order to explain "why communication costs differ across partitioning methods" from circuit structure. The aim is to provide a basis for comparing and predicting "which partitioning method is likely to be advantageous" from intrinsic circuit structure, without relying on benchmark-specific or method-specific heuristics. Moreover, we systematize the indicators into groups (density, bias, parallelism, spectral/partitionability, and balance/capacity), and explicitly present, as a checklist, "where to look in a circuit before partitioning." This is an important contribution of this work.

How to read the global ranking (Table 6.2): not the magnitude of RCX, but conditions under which method differences emerge Rank Influence is not intended to predict the absolute RCX value itself. Rather, it is a framework for extracting structural factors that are strongly associated with **the phenomenon that the relative ordering (win/loss relation) among methods changes from circuit to circuit**. In Table 6.2, each indicator's contribution is repeatedly evaluated via bootstrapping and summarized by the mean, variability (std), 95% CI, and the top-K frequency. Therefore, the message here is not "which circuits have many remote CNOTs," but **under what circuit structures RCX is more likely to change depending on the choice of partitioning method**.

Implications of the top indicators: what structures tend to matter for partitioning The top indicators in Table 6.2 (e.g., P_{avg}, EntVar, M, B_k, D, m, Rdup, n) can be summarized into the following three points when mapped onto the indicator groups above (see Table 4.1 for definitions).

1. **More simultaneously executable two-qubit gates (parallelism) makes method differences more visible.**
   The average parallelism $P_{\text{avg}} = \frac{1}{T}\sum_{t=1}^{T} g_t$ (the mean number of two-qubit gates g_t in layer t) measures "how many two-qubit gates run at the same time" (strength of simultaneous communication demand). In circuits with many independent two-qubit gates within the same layer, differences in node assignment are more likely to appear as *remote CNOT occurrences around the same time step*, which tends to split the relative performance among methods.

2. **Stronger concentration on specific qubits/pairs (bias) makes method differences more visible.**
   EntVar measures "imbalance of two-qubit-gate load across qubits," and Rdup, defined as $R_{\text{dup}} = M/|E|$, represents "the strength of repetition of the same qubit pair." Their high rank implies that *hot qubits* and *repeatedly appearing pairs* tend to dominate RCX, and that method differences are likely to emerge depending on how well a method handles them (e.g., local update effectiveness, assignment policy). A key point is that these often heuristically discussed notions of "concentration/repetition" are cast into **automatically computable quantities** (EntVar, Rdup, and Hmax), making the discussion reproducible.

3. **Large total interaction amount and pair diversity (density) can, depending on the situation, shrink method differences.**
   M is the total number of two-qubit gates (sum of weights), m is the number of distinct interacting pairs, and $D = m/\binom{n}{2}$ is the used-pair density. Large values indicate that interactions are "many/diverse," which can lead to situations where *a certain amount of remote CNOT is unavoidable regardless of assignment.* In our interpretation, circuits with strong density and pair diversity tend to exhibit smaller method differences, whereas circuits with strong local repetition or hotspots tend to exhibit clearer method differences.

"A few qubits become key" can also be diagnosed in circuit vocabulary B_k (concentration of the top betweenness-centrality values) measures whether *the role of connecting multiple interaction clusters* is concentrated on a small number of qubits, implying that assignment of those qubits can dominate RCX. This also turns what is often described informally as "some key points exist" into a quantitative measure (B_k), enabling method-difference discussions.

Why some indicators have mean 0 Table 6.2 includes indicators with mean (and std, CI) displayed as 0 (e.g., Circuit width, Q_{node}). This is not because these quantities are meaningless; rather, under the current experimental setting, we **fix the number of partitions at $P = 4$**, which makes them **information-wise redundant**, so that the learner (tree-based models) uses them rarely, resulting in SHAP contributions being 0 (or extremely small and rounded to 0).

In summary, the global ranking provides a circuit-level vocabulary for "what to inspect before partitioning": **parallelism (simultaneity), bias (concentration/repetition), density (total amount/diversity), key qubits (B_k), and capacity conditions**. In the next subsection, we organize how these structural viewpoints correspond to each method ' s update operations and assignment policy.

6.3.2 Guidelines for method-specific discussion (linking update operations to two-qubit-gate placement)

From this subsection onward, we examine how differences in partitioning outcomes arise from circuit structure by returning to each method ' s update operation (what information it uses and at what granularity it changes assignments). We focus on three methods: FM, KL, and HQA. Because their optimization procedures differ, the ease of RCX (Remote CNOT) reduction can differ even on the same circuit.

For each method, we first clarify the nature of the update operation during optimization, and then explain under what forms of two-qubit-gate placement in the circuit that update operation is more likely to act in the direction of lowering RCX. Concretely, we describe two-qubit-gate placement characteristics such as whether interaction pairs repeat, whether interactions concentrate on specific qubits, and whether many independent two-qubit gates appear within the same layer, and relate them to the circuit-structure indicators defined in this study.

6.3.3 FM: local improvement via single-qubit moves

In this subsection, based on the nature of FM (Fiduccia–Mattheyses) updates, we discuss what kinds of two-qubit-gate placements tend to allow RCX reductions. In our implementation, to obtain a k-way partition, we run FM multiple times by repeatedly applying bipartitioning to subsets.

(1) Nature of the optimization (FM)

FM is a local search that, while satisfying capacity constraints, **moves a single qubit to another partition** and improves the objective using the reduction in the number (or weight) of two-qubit gates crossing partitions (corresponding to Remote CNOT) as the gain. Thus, FM is strong in situations where **changing the membership of one qubit alone makes many CNOT pairs incident to that qubit become intra-node**. On the other hand, because it does not directly include multi-qubit simultaneous swaps, when crossings are distributed over many qubit pairs, the reduction achievable by one move tends to be small. Moreover, within one pass, FM proceeds while fixing already-moved qubits, so repeating passes does not guarantee reaching the global optimum; under capacity constraints, effective move candidates can be exhausted early.

(2) Structural conditions where FM tends to work well

FM tends to work well when **changing the membership of a small number of qubits can reduce many crossing CNOTs (RCX) at once**. In terms of our 25 indicators, this corresponds roughly to the following.

- **A small number of qubits perform CNOTs frequently with many partners:** As indicated by EntVar or Hmax, interaction counts are biased across qubits. Then, moving that hot qubit to co-locate with its frequent partners can resolve many crossings simultaneously.
- **The same qubit pair ' s CNOT appears repeatedly in the circuit:** As indicated by Rdup, repeated use of the same pair is strong. If a frequent pair is split across nodes, the same crossing repeats proportional to the number of occurrences, increasing RCX, so moving qubits involved in that pair tends to be directly effective.
- **Interactions contributing to crossings concentrate around specific qubits:** As indicated by Bk, there exist qubits that strongly mediate inter-partition connections. Moving the membership of such a qubit alone can substantially change the total number (or weight) of crossing CNOTs.
- **Pair types are not uniformly spread over the entire circuit:** As indicated by D or m, the pair diversity has not reached the "almost all pairs" regime (not the situation where most pairs appear only once). Then, there remain many pairs that can be made intra-node by a small number of moves.

Evidence (validation) In best-vs-worst comparisons where FM is advantageous, bias-related indicators (EntVar/Hmax/Bk) and repeated-pair indicators (Rdup) tend to appear frequently on the "winning-factor" side. This aligns with FM ' s update mechanism, which reduces crossings through single-qubit moves.

(3) Conditions where FM tends to work poorly

FM tends to work poorly when **crossing CNOTs are distributed over many pairs and moving one qubit does not reduce crossings much**. Typical indicators are:

- **Many types of CNOT pairs appear broadly:** D and m are large and crossings are spread across many pairs.
- **Many independent two-qubit gates appear within the same layer:** Pavg, Peff, and $|H_t|$ are large. When many independent CNOTs appear in the same layer, small assignment perturbations can cause many crossings within the same layer, and single-qubit moves may fail to sufficiently reduce such simultaneous crossings.
- **Moves are restricted by capacity constraints:** Moving the qubits that would most reduce crossings may violate capacity constraints, preventing high-impact moves.

(4) Indicator profile of circuits expected to favor FM

Circuits expected to favor FM are characterized by:

- **Biased interaction counts across qubits:** EntVar, Hmax, and Bk are prominent.
- **Strong repeated-pair structure:** Rdup is prominent.
- **Pair diversity has not fully diffused:** D and m do not reach the "almost all pairs" level.
- **Very high within-layer parallelism makes improvement harder:** larger Pavg, Peff, and $|H_t|$ increases situations where single moves cannot eliminate simultaneous crossings.

6.3.4 KL: assignment readjustment via swapping qubit pairs

In this subsection, based on the KL (Kernighan–Lin) update operation (swap) and the nature of extending to k-way partitions by recursively applying bipartitioning, we discuss what kinds of two-qubit-gate placements tend to yield improvements.

(1) Nature of the optimization operation (KL)

KL ' s basic operation is to **swap a pair of qubits belonging to different partitions** and iteratively reduce the total number (or weight) of two-qubit gates crossing partitions (corresponding to Remote CNOT). Unlike single-qubit moves, a swap moves qubits on both sides simultaneously, allowing continuous changes to the global assignment combination while maintaining capacity constraints. As a result, KL is strong in situations where **even if crossings are distributed across many pairs, repeated swaps can gradually exchange the set of qubits involved in crossings and reduce crossing CNOTs step by step**.

(2) Structural conditions where KL tends to work well

KL tends to work well when **repeated swaps can gradually reduce crossing CNOTs**. Typical indicators are:

- **Many independent two-qubit gates appear within the same layer:** Pavg, Peff, and $|H_t|$ are large. In such circuits, which qubit set is co-located can significantly change the number of crossings, so iterative combination changes by swaps can be effective.
- **Many CNOT pairs are likely to become crossings after partitioning:** As indicated by Savg, many interactions extend inside and outside partitions, so assignment differences readily appear as RCX differences.
- **Many pair types exist:** D and m are large. With many pair types, single-qubit moves may hit a limit on how many pairs can be co-located, while swaps can capture improvement opportunities by rearranging multi-qubit relations.
- **Interactions appear as groups of qubits:** When Qmod or $\bar{\lambda}_2$(rec) indicates that interactions concentrate into several groups, recursive bipartitioning and swaps can

more easily converge toward assignments that co-locate those groups.

Evidence (validation) In best-vs-worst comparisons where KL is advantageous, parallelism/layer-simultaneity indicators (Pavg/Peff/$|H_t|$) and Savg tend to appear frequently on the "winning-factor" side, while strong hot-qubit concentration (EntVar/Hmax) tends not to appear on the winning side. This aligns with swap-based updates that adjust assignment combinations.

(3) Conditions where KL tends to work poorly

KL tends to work poorly when **the membership of a small number of qubits almost determines RCX, and a direct single move is the shortest path**.

- **Membership of a particular qubit dominates the number of crossings:** When EntVar, Hmax, and Bk are prominent and moving one qubit toward a certain node drastically changes RCX, swaps may be less efficient than direct moves because the counterpart qubit is also moved.
- **The size of strongly coupled qubit groups does not match node capacity:** When CVC or Bsize indicates strong size imbalance, equal-capacity partitioning may be forced to split the group, leaving crossings that swaps cannot eliminate.
- **Crossings are dominated by a small number of frequent pairs:** When Rdup is prominent and most crossings are determined by a few frequent pairs, directly co-locating qubits involved in those pairs via moves may be more efficient.

(4) Indicator profile of circuits expected to favor KL

Circuits expected to favor KL are characterized by:

- **Many independent CNOTs within the same layer:** Pavg, Peff, and $|H_t|$ are prominent.
- **Many interactions likely become crossings after partitioning:** Savg is prominent.
- **Many pair types:** D and m are prominent.
- **Interactions appear as groups:** Qmod and $\bar{\lambda}_2$(rec) are above a certain level.
- **Group-size imbalance is not too strong:** the imbalance indicated by CVC/Bsize is not excessive (easier to co-locate under equal capacity).

6.3.5 HQA: one-shot partitioning based on aggregated whole-circuit interactions

In this subsection, based on the nature of HQA (Hungarian-based Qubit Assignment) updates, we fix the conditions under which improvements are likely to occur. Unlike FM/KL, HQA does not perform iterative local improvement; it decides the partition in one shot using the whole circuit (or summarized interactions).

(1) Nature of the optimization (HQA)

HQA constructs a partitioning problem by aggregating interaction information in the circuit, and uses the Hungarian method to **determine the partitioning result in a single batch**. Thus, rather than gradually reducing crossings via sequential search, HQA is strong in **constructing, in one shot, an assignment that places qubit pairs that tend to increase crossings (frequently occurring interactions or interactions that tend to co-occur within the same layer) into the same partition**. On the other hand, because it lacks iterative updates after the partition is fixed, success depends on how well the partitioning stage can co-locate the important interactions.

(2) Structural conditions where HQA tends to work well

HQA tends to work well when **there exist interaction patterns that repeat throughout the circuit and an assignment that co-locates them can be determined straightforwardly**.

- **High parallelism within layers and repeated reappearance of similar interactions across the circuit:** When parallelism is high (Pavg, Pmax, Peff, $|H_t|$) and similar interactions reappear across the circuit (as indicated by Rdup and Lavg), a one-shot assignment that captures the main co-location relationships can significantly reduce RCX.
- **Interactions appear concentrated into several qubit groups:** When Qmod or $\bar{\lambda}_2$(rec) indicates group structure, a static assignment that packs each group into one node can be effective.
- **Assignment differences directly translate to crossing CNOT counts:** When Savg is large, which qubit group is co-located readily appears as RCX differences.

Evidence (validation) In best-vs-worst comparisons where HQA is advantageous (or comparable to KL), layer-level simultaneity scale (Pavg/Pmax/Peff/$|H_t|$), Savg, and group-structure indicators (Qmod, $\bar{\lambda}_2$(rec)) tend to appear on the "winning-factor" side. This aligns with HQA ' s one-shot decision based on whole-circuit interactions.

(3) Conditions where HQA tends to work poorly

HQA tends to work poorly when **crossings can be greatly reduced by changing the membership of only a few qubits, but after one-shot assignment, HQA cannot specifically target and move those qubits**.

- **Membership of a particular qubit dominates crossings:** When EntVar, Hmax, and Bk are prominent and RCX changes greatly by flipping one qubit ' s assignment, iterative-update methods (e.g., FM) that can move that qubit may have an advantage.
- **Desired co-located group sizes do not fit node capacity:** When Bsize or CVC indicates that groups that should be co-located do not fit equal capacity, static assignment cannot avoid splitting them, leaving crossings.

- **Most crossings are determined by a small number of frequent pairs:** When Rdup is prominent and feasibility constraints prevent co-locating those dominant frequent pairs, HQA can become strongly disadvantaged.

(4) Indicator profile of circuits expected to favor HQA

Circuits expected to favor HQA are characterized by:

- **Many independent CNOTs within layers:** Pavg, Pmax, Peff, and $|H_t|$ are prominent.
- **Assignment differences directly translate to crossing counts:** Savg is prominent.
- **Interactions appear as groups:** Qmod and $\bar{\lambda}_2$(rec) are above a certain level.
- **Not dominated by the membership of a single hot qubit:** EntVar, Hmax, and Bk are not overwhelmingly dominant.
- **Group-size imbalance is not too strong:** the imbalance indicated by Bsize/CVC is not excessive (easier to fit into capacity).

6.3.6 Per-circuit discussion (structure interpretation based on SHAP)

In this subsection, using local Rank Influence obtained from SHAP analysis, we check whether the circuit-structure indicators computed for each quantum circuit appropriately represent the circuit ' s two-qubit-gate placement characteristics. The goal here is not to argue which partitioning method is superior, but rather to examine from the circuit-structure viewpoint whether the indicators provide explanations consistent with the actual circuit structure, i.e., "which indicators contributed to explaining method differences for that circuit."

For each circuit, we first refer to the ranking of local Rank Influence and the values of $I_i(c)$, and, from both highly contributing indicators and indicators with almost no contribution, we concretely verbalize how two-qubit gates appear (whether pairs repeat, whether involved qubits are biased, layer-level parallelism, etc.). Next, we check whether the interpretation is consistent with two-qubit-gate placement that can be read directly from the circuit construction (e.g., GHZ-state generation, QFT, TFIM simulation, arithmetic circuits). Through this procedure, before discussing "method-selection rules based on structure indicators" in later sections, we verify per circuit that the indicators indeed represent circuit structure.

6.3.7 `bigadder_n16`

Characteristics inferred from circuit-structure indicators In local Rank Influence, Pavg is ranked 1st (0.3748), about 3.22 times the 2nd-ranked CVC (0.1165). The 3rd-ranked indicator is M (0.0679). Meanwhile, Pmax, Rlong, Q_{node}, and Roverload are all 0. Thus, for this circuit, the factors separating method differences are foregrounded as layer-level parallelism (Pavg), cluster-size imbalance (CVC), and total two-qubit-gate volume (M).

This circuit has 16 qubits and an extremely small number of two-qubit gates, with $M = 8$. The number of distinct pairs is $m = 8$, and the duplication ratio is $Rdup = M/m = 1.000$. The density is also small ($D = 0.067$), indicating that the two-qubit gates are limited to 8 independent pairs. In this situation, each qubit participates in at most one two-qubit gate, which is consistent with $Hmax = 1$ and $EntVar = 0$, and with the fact that the circuit is composed only of

$$\mathrm{CX}(q_i \to q_{i+8}) \quad (i = 0, \ldots, 7)$$

i.e., 8 two-qubit gates that *do not share qubits*. In other words, the placement does not produce patterns where a qubit appears in multiple two-qubit gates with many partners; rather, it is completed as "8 pairs each appearing exactly once."

Method implications derived from structure indicators For independent-pair structure, the essence is simply "keep each pair within the same partition," and once this is achieved, the two-qubit gates contributing to the cut disappear. This condition is less sensitive to the type of update operation, and many partitions can yield the same objective value. Hence, rather than search granularity, the key is whether a method can consistently realize assignments that preserve pairs under capacity constraints.

6.3.8 `dnn_n16`

Characteristics inferred from circuit-structure indicators Local Rank Influence ranks Pavg 1st (0.2513), M 2nd (0.1011), and CVC 3rd (0.0983), with the ratio between 1st and 2nd about 2.49. The bottom indicators Pmax, Rlong, Q_{node}, and Roverload are 0. Thus, layer-level parallelism (Pavg) and total two-qubit-gate volume (M), with cluster imbalance (CVC) as the next contributor, form the main axes separating method differences.

This circuit has $n = 16$, $M = 384$, and $m = 16$, so $Rdup = 24.000$. The density is relatively small ($D = 0.133$), but $m = 16$ means that two-qubit gates appear only on 16 distinct pairs. Here, "limited to 16 pairs" means that the interacting pairs are restricted to

$$(q_i, q_{(i+1) \bmod 16}) \quad (i = 0, \ldots, 15),$$

i.e., adjacent pairs on a ring (with (q_{15}, q_0) closing the ring). Moreover, $M/m = 384/16 = 24$ implies that *each pair appears 24 times* in the circuit.

Indeed, $Hmax = 48$ and $EntVar = 0$ (all qubits participate equally), consistent with this rule. Each qubit q_i appears in two pairs, (q_{i-1}, q_i) and (q_i, q_{i+1}). Since each pair appears 24 times, q_i appears $24 \times 2 = 48$ times in two-qubit gates for all i, yielding zero variance (EntVar). This matches the typical DNN-family benchmark pattern where the interacting pairs are fixed and repeatedly applied over many layers.[14]

Method implications derived from structure indicators With repeated fixed pairs ($Rdup = 24.000$), RCX is determined almost entirely by *which of the 16 pairs remain within the same node and which are split across nodes.* If a pair is split, its impact accumulates 24 times. Because the structure is symmetric, multiple assignments can share the same property (same set of intra-node pairs), leading to multiple equivalent solutions. In this setting, either single-qubit moves (FM) or swaps (KL), as long as they reduce the number of split pair types, can reach similar objective values. Thus, the key is not the update type itself but whether the implementation stably converges to assignments that keep repeated pairs intra-node under capacity constraints.

6.3.9 `ghz_16`

Characteristics inferred from circuit-structure indicators Local Rank Influence ranks Pavg 1st (0.1841), M 2nd (0.1029), and CVC 3rd (0.0937). The ratio between 1st and 2nd is about 1.79, and the absolute values are smaller than in some other circuits, indicating a weaker signal separating methods. The bottom indicators Pmax, Rlong, Q_{node}, and Roverload are 0.

This circuit has $n = 16$, $M = 15$, $m = 15$, and $Rdup = 1.000$. The density is small ($D = 0.125$), and the two-qubit-gate pairs are limited to

$$(q_0, q_1), (q_1, q_2), \ldots, (q_{14}, q_{15}),$$

each appearing exactly once. Thus, RCX increases depending solely on how many of these 15 gates end up acting on qubits assigned to different nodes.

The degree distribution differs by qubit index, with $Hmax = 2$ and $EntVar = 0.109$: there is no hub, but participation counts are not uniform. This is consistent with the structure: q_0 and q_{15} appear once, while $q_1, \ldots, q_{14}$ appear twice. The mean is $15 \times 2/16 = 1.875$, and

$$\frac{2(1-1.875)^2 + 14(2-1.875)^2}{16} = 0.109375,$$

matching EntVar=0.109. This matches the GHZ-state generation CNOT chain (a linear chain of CNOTs).[14]

Method implications derived from structure indicators In this circuit, two-qubit gates appear only as (q_i, q_{i+1}) and each pair appears once (Rdup=1.000). Therefore, the objective is primarily determined by which adjacent pairs remain within the same node versus become inter-node (remote CNOT). Assignments that group contiguous index ranges,

$$\{q_s, q_{s+1}, \ldots, q_t\},$$

tend to keep many adjacent pairs intra-node and reduce RCX. Conversely, mixing far-apart indices within a node tends to increase the number of adjacent pairs split across nodes. Because each edge has equal weight and appears once, many assignments can yield the same number of remote CNOTs, making results less sensitive to the update type. Thus, stability in reaching such contiguous-group assignments under capacity constraints is more important than the search type.

6.3.10 `ghz_32`

Characteristics inferred from circuit-structure indicators Local Rank Influence ranks Pavg 1st (0.1864), M 2nd (0.1181), and CVC 3rd (0.0860), with the ratio between 1st and 2nd about 1.58. The bottom indicators Pmax, Rlong, Q_{node}, and Roverload are 0.

This circuit has $n = 32$, $M = 31$, $m = 31$, $Rdup = 1.000$, and $D = 0.062$. The interacting pairs are limited to

$$(q_0, q_1), (q_1, q_2), \ldots, (q_{30}, q_{31}),$$

each appearing once. Thus, as in `ghz_16`, RCX differences are determined by how many of these 31 gates are split across nodes. $Hmax = 2$ and $EntVar = 0.059$ reflect that q_0 and q_{31} appear once, while $q_1, \ldots, q_{30}$ appear twice. The GHZ-chain structure is consistently reflected in the indicators.

Method implications derived from structure indicators As in `ghz_16`, the number of remote CNOTs is determined largely by which adjacent pairs remain intra-node. Because many assignments can yield the same RCX count, the update type matters less than stable convergence under constraints. Hence, implementations that naturally induce contiguous-index grouping

$$\{q_s, q_{s+1}, \ldots, q_t\}$$

are suitable.

6.3.11 `ghz_64`

Characteristics inferred from circuit-structure indicators Local Rank Influence ranks Pavg 1st (0.2272), CVC 2nd (0.1008), and EntVar 3rd (0.0602), with the ratio between 1st and 2nd about 2.25. The bottom indicators include Pmax, Rlong, Q_{node}, and Roverload at 0.

This circuit has $n = 64$, $M = 63$, $m = 63$, $Rdup = 1.000$, and $D = 0.031$. Pairs are limited to

$$(q_0, q_1), (q_1, q_2), \ldots, (q_{62}, q_{63}),$$

each appearing once. $Hmax = 2$ and $EntVar = 0.030$ reflect that the endpoint qubits q_0 and q_{63} appear once and the others appear twice, with the relative impact decreasing as n increases. Structurally, the circuit is consistently dominated by the same adjacent-pair rule.

Method implications derived from structure indicators RCX differences are again determined by how many adjacent pairs become inter-node. Because all edges are equal weight and appear once, the objective is largely determined by the number of split edges, not which edges are split, yielding multiple equivalent solutions. Thus, searches that tend to group contiguous indices

$$\{q_s, q_{s+1}, \ldots, q_t\}$$

and keep (q_i, q_{i+1}) within nodes are preferred. As n increases, capacity constraints have relatively stronger effects; small changes in assignment can change RCX by one edge. Therefore, in implementation, it is desirable to update candidates multiple times to reliably incorporate changes that reduce RCX.

6.3.12 `mermin_bell_16`

Characteristics inferred from circuit-structure indicators Local Rank Influence ranks Pavg 1st (0.4002), CVC 2nd (0.1106), and M 3rd (0.0348). The ratio between 1st and 2nd is about 3.62, making parallelism the main axis separating methods. The bottom indicators include Pmax, Rlong, Q_{node}, and Roverload at 0.

This circuit has $n = 16$ and $m = 120$ with $D = 1.000$, i.e., the used pairs cover all combinations (a complete graph). Here, $D = 1.000$ means that *for any distinct pair* (q_i, q_j) *with* $i \neq j$*, at least one two-qubit gate appears in the circuit*, so interactions are not limited to a small subset of pairs.

On the other hand, $M = 171$ and $Rdup = 1.425$, so pair occurrence counts are not uniform. Indeed, $Hmax = 31$ and $EntVar = 7.109$ are large, indicating concentration on specific qubits. Even on a complete graph, if pair frequencies are nonuniform, the number of times particular pairs repeat can be biased, so RCX can vary depending on whether high-frequency pairs are co-located. This aligns with the Mermin–Bell family, which tends to involve many qubits and exhibit broad pair coverage.[14]

Method implications derived from structure indicators In a complete graph ($D = 1.000$), a large number of inter-node two-qubit gates is unavoidable for any assignment. However, because weights are nonuniform ($Rdup = 1.425$, nonzero weight variation), RCX can still change depending on whether high-frequency pairs (large w_{ij}) can be kept intra-node. EntVar indicates that there exist qubits with many two-qubit gate occurrences. If such a qubit is placed on a node and its frequent partners can also be brought to the same node, RCX tends to decrease. Thus, searches that can gradually eliminate situations where high-frequency pairs are split across nodes via single-qubit moves are suitable.

6.3.13 `multiplier_n15`

Characteristics inferred from circuit-structure indicators Local Rank Influence ranks CVC 1st (0.4022), Pavg 2nd (0.1574), and M 3rd (0.0786). The ratio between 1st and 2nd is about 2.56, indicating that cluster imbalance (CVC) is a dominant axis separating method differences. The bottom indicators include Pmax, Rlong, Q_{node}, and Roverload at 0.

This circuit has $n = 15$, $M = 246$, $m = 30$, $D = 0.286$, and $Rdup = 8.200$. It exhibits strong repetition of a small subset of pairs, and also extreme concentration: $Hmax = 75$ and $EntVar = 443.760$. That is, a particular qubit appears in two-qubit gates far more often and repeatedly interacts with multiple partners. This matches typical arithmetic-circuit patterns with aggregation or frequently referenced control lines.[14]

Method implications derived from structure indicators When concentration (EntVar, Hmax) and repetition (Rdup) are both strong, RCX is often dominated by whether the high-frequency pairs (large w_{ij}) are split across nodes. CVC being top-ranked suggests that a dense interacting qubit group and a relatively sparse group coexist, with uneven group sizes. Then, the qubit group one would like to pack into a single node can become larger than node capacity, creating unavoidable splits.

In this setting, the search must update assignments while always satisfying capacity constraints. If an implementation permits capacity-violating updates or resolves overload unstably, updates that keep high-frequency pairs intra-node can be invalidated, worsening RCX. Hence, searches that can target split high-frequency pairs and improve them step by step without breaking capacity are suitable.

6.3.14 `multiplier_n45`

Characteristics inferred from circuit-structure indicators Local Rank Influence ranks CVC 1st (0.4051), Pavg 2nd (0.1520), and M 3rd (0.0640), with the ratio between 1st and 2nd about 2.66. The bottom indicators include Pmax, Rlong, Q_{node}, and Roverload at 0.

This circuit has $n = 45$, $M = 2574$, $m = 177$, $D = 0.179$, and $Rdup = 14.542$. Repetition is even stronger, with extreme concentration ($Hmax = 225$, $EntVar = 5477.040$). While there are 177 distinct pairs, each pair appears on average about 14.5 times, and a particular qubit's participation dominates. As the scale increases, the "group to co-locate" grows and is more likely to collide with capacity constraints.

Method implications derived from structure indicators At this scale, even if we try to pack high-frequency pairs intra-node, capacity constraints prevent packing all desired qubits together. Thus, which qubits to move out (which qubits to move/swap) directly affects RCX. Therefore, rather than deciding in one large step, it is preferable to use searches that can gradually reduce split high-frequency pairs while strictly satisfying capacity constraints at every step. CVC being 1st indicates that group-size nonuniformity is a primary source of method differences, so it is important that the implementation enforces capacity-feasible updates only.

6.3.15 `qft_16`

Characteristics inferred from circuit-structure indicators Local Rank Influence ranks Pavg 1st (0.4348), CVC 2nd (0.1170), and M 3rd (0.0418), with ratio about 3.72; parallelism is the main axis. The bottom indicators include Pmax, Rlong, Q_{node}, and Roverload at 0.

This circuit has $n = 16$ and $m = 120$ with $D = 1.000$, i.e., a complete graph. Moreover, $EntVar = 0$, meaning all qubits have identical participation counts (a regular interaction structure). $Rdup = 1.200$ indicates some repetition but no hub concentration. This complete-graph regularity aligns with the expansion of all-to-all controlled-phase interactions in QFT.[14]

Method implications derived from structure indicators For a complete and regular graph ($D = 1.000$, $EntVar = 0$), partitioning cannot fundamentally reduce crossings, and many equivalent solutions exist. As a result, the update type is less likely to affect outcomes, and different methods may converge to similar partitions.

6.3.16 `qft_32`

Characteristics inferred from circuit-structure indicators Local Rank Influence ranks Pavg 1st (0.4384), CVC 2nd (0.1163), and M 3rd (0.0431), with ratio about 3.77. The bottom indicators are the same set as `qft_16` (0-valued group including Pmax, Rlong, Q_{node}, Roverload).

QASM aggregation gives $n = 32$, $m = 496$, $D = 1.000$ (complete graph), and $EntVar = 0$ (regular). $Rdup = 1.097$ decreases, and scale-up increases the "many pairs appear once" component.

Method implications derived from structure indicators Due to complete-graph regularity, many equivalent solutions exist and method differences are structurally unlikely. Hence, it is reasonable to choose a method based on operational criteria such as stability or runtime.

6.3.17 `qft_64`

Characteristics inferred from circuit-structure indicators Local Rank Influence ranks Pavg 1st (0.4424), CVC 2nd (0.1179), and M 3rd (0.0358), with ratio about 3.75. The bottom indicators are again the same 0-valued group as above.

QASM aggregation gives $n = 64$, $m = 2016$, $D = 1.000$, $EntVar = 0$, $Rdup = 1.048$, making complete-graph regularity even more pronounced.

Method implications derived from structure indicators Because of the abundance of equivalent solutions induced by complete-graph regularity, method differences are unlikely to be reflected in results.

6.3.18 `square_root_n18`

Characteristics inferred from circuit-structure indicators Local Rank Influence ranks CVC 1st (0.4054), Pavg 2nd (0.1699), and M 3rd (0.0747), with ratio about 2.39, indicating that cluster imbalance is a dominant axis (as in the `multiplier` family). The bottom indicators include Pmax, Rlong, Q_{node}, and Roverload at 0.

This circuit has $n = 18$, $M = 898$, $m = 36$, $D = 0.235$, and $Rdup = 24.944$. Repetition is extremely high, and concentration is extreme: $Hmax = 208$, $EntVar = 3718.173$. That is, only a limited set of pairs appears many times, and a particular qubit ' s participation dominates. This bias is natural in square-root circuits that involve repeated control and aggregation in reversible arithmetic.[14]

Method implications derived from structure indicators With both strong repetition (Rdup) and strong concentration (EntVar, Hmax), RCX is largely caused by whether the dominant qubit and its frequent partners are split across nodes. Thus, searches that can gradually reduce split high-frequency interactions via qubit moves can be advantageous. Because CVC is top-ranked, the "group to co-locate" tends to be nonuniform in size, so satisfying capacity constraints while improving assignments can determine the outcome.

6.3.19 `swap_test_n25`

Characteristics inferred from circuit-structure indicators Local Rank Influence ranks Pavg 1st (0.1804), CVC 2nd (0.0849), and Bk 3rd (0.0559), with the ratio between 1st and 2nd about 2.12. The bottom indicators include Pmax, Rlong, Q_{node}, and Roverload at 0.

This circuit has $n = 25$, $M = 96$, $m = 36$, $D = 0.120$, and $Rdup = 2.667$. It exhibits concentration ($Hmax = 48$, $EntVar = 67.738$). Because the swap test uses an ancillary (control) qubit to apply similar operations to many qubits, one qubit can appear in many two-qubit gates and can have many partners. If that qubit is placed on a different node, many related two-qubit gates can become remote CNOTs at once.[14]

Method implications derived from structure indicators When a qubit appears many times and interacts with multiple partners, RCX is largely determined by where that qubit is placed and whether its frequent partners can be co-located. Under capacity constraints, reducing RCX requires updates (moves/swaps) that can shrink the situation where many gates incident to that qubit become inter-node. Thus, searches that can gradually improve this via moves/swaps are suitable. However, because one qubit interacts with many partners, it is generally impossible to pack all partners into a single node under capacity constraints. As a result, some inter-node gates remain in any partition, and when further improvements are small, different methods may reach similar RCX.

6.3.20 `tfim_hamiltonian_sim_16`

Characteristics inferred from circuit-structure indicators Local Rank Influence ranks Pavg 1st (0.2042), M 2nd (0.1029), and CVC 3rd (0.0942), with the ratio between 1st and 2nd about 1.98. The bottom indicators include Pmax, Rlong, Q_{node}, and Roverload at 0.

This circuit has $n = 16$, $M = 30$, $m = 15$, $D = 0.125$, and $Rdup = 2.000$. Each pair appears twice on average, so $Rdup = 2$, and $Hmax = 4$, $EntVar = 0.438$, indicating stronger participation bias than `ghz_16` (Rdup=1). Here, $Rdup = 2$ means that the interacting pairs are limited to

$$(q_0, q_1), (q_1, q_2), \ldots, (q_{14}, q_{15})$$

(15 types), and each pair appears about twice. Thus, q_0 and q_{15} appear twice, while

$q_1, \ldots, q_{14}$ appear four times, yielding

$$\frac{2(2-3.75)^2 + 14(4-3.75)^2}{16} = 0.4375,$$

consistent with EntVar=0.438. This matches the typical structure of discretized time evolution for 1D Ising-type Hamiltonians such as TFIM, where similar two-qubit gate patterns repeat across multiple steps.[14]

Method implications derived from structure indicators When the same adjacent pair (q_i, q_{i+1}) repeats, if that pair is split across nodes, the same remote CNOT repeats and accumulates. Thus, reducing RCX requires assignments that keep frequently appearing pairs intra-node. Here, rather than the update type, it is important that the implementation can stably search without excessively breaking the adjacent-pair rule.

6.3.21 `tfim_hamiltonian_sim_32`

Characteristics inferred from circuit-structure indicators Local Rank Influence ranks Pavg 1st (0.2009), M 2nd (0.0851), and CVC 3rd (0.0840), with the same 0-valued bottom group as above.

QASM aggregation gives $n = 32$, $M = 62$, $m = 31$, $D = 0.062$, $Rdup = 2.000$. Pairs are limited to

$$(q_0, q_1), (q_1, q_2), \ldots, (q_{30}, q_{31}),$$

each appearing about twice. $Hmax = 4$ and $EntVar = 0.234$ indicate that relative bias decreases with scale, but the backbone "the same adjacent pairs repeat" remains unchanged.

Method implications derived from structure indicators RCX is determined by how well repeated pairs (q_i, q_{i+1}) are kept intra-node. When edge weights are nearly uniform, multiple assignments can yield the same number of split pairs, making it harder for method differences to reflect in the final result. Operationally, it is reasonable to prioritize search cost and stability while selecting a method that does not easily break the pair rule.

6.3.22 `tfim_hamiltonian_sim_64`

Characteristics inferred from circuit-structure indicators Local Rank Influence ranks Pavg 1st (0.2470), CVC 2nd (0.1008), and M 3rd (0.0824), with the same 0-valued bottom group as above.

This circuit has $n = 64$, $M = 126$, $m = 63$, $D = 0.031$, $Rdup = 2.000$, $Hmax = 4$, $EntVar = 0.121$. Pairs are limited to

$$(q_0, q_1), (q_1, q_2), \ldots, (q_{62}, q_{63}),$$

each appearing about twice. With scale-up, the endpoint-vs-middle participation difference becomes relatively smaller, further reducing EntVar, but the pair rule remains unchanged.

Method implications derived from structure indicators Reducing RCX mainly requires keeping repeated adjacent pairs (q_i, q_{i+1}) intra-node. At larger scale, small assignment changes under capacity constraints can change the number of split pairs, so searches that try multiple moves/swaps and reliably incorporate RCX-reducing changes are desirable. However, because the circuit rule itself is simple, the range where update-type differences are reflected tends to be limited.

Method implications derived from structure indicators Reducing RCX mainly requires keeping repeated adjacent pairs (q_i, q_{i+1}) intra-node. At larger scale, small assignment changes under capacity constraints can change the number of split pairs, so searches that try multiple moves/swaps and reliably incorporate RCX-reducing changes are desirable. However, because the circuit rule itself is simple, the range where update-type differences are reflected tends to be limited.

In the above, through per-circuit two-qubit-gate placement and structure interpretation based on local Rank Influence, we organized per circuit "under what structural conditions FM/KL/HQA updates tend to reduce RCX." Next, we check how well these interpretations align with actual partitioning outcomes. Concretely, we predict for each circuit the "method likely to be minimal" from structure indicators (local Rank Influence ranking), and compare it with the method that achieved the minimum in empirical RCX (Table 6.1). The correspondence is summarized in Table 6.4, and we discuss both matching and failing circuits to clarify conditions and limits of prediction.

6.3.23 Validity of structure-based method selection (correspondence between prediction and empirical RCX)

In this subsection, we summarize, in a per-circuit at-a-glance form, whether the insights from "linking update operations to two-qubit-gate placement" (method-specific hypotheses) and "per-circuit discussion (SHAP-based structure interpretation)" are consistent with the empirical minimum-RCX method in the actual RCX results (Table 6.1).

Table 6.4: Agreement between predictions based on circuit-structure indicators (upper side of local Rank Influence) and empirical RCX results. "Top" lists, for each circuit c, the indicators sorted in descending order of the *local Rank Influence* $I_i(c)$ (Table 6.3). For the worst side, we first summarize the group of indicators with $I_i(c) = 0$, and then list the bottom three indicators (in ascending order) after excluding zeros.

Circuit	Pred(best)	Emp(best)	Top–5 indicators ($I_i(c)$)	Worst-side indicators
bigadder_n16	fm/kl/hqa	fm/kl/hqa	Pavg (0.3748) / CVC (0.1165) / M (0.0679) / m (0.0223) / Q_{mod} (0.0220)	**0:** Pmax / Rlong / Q_{node} / Roverload (0.0000) **Low (nonzero):** Var_{ent} (0.0003) / $S_{\text{spec}}(P)$ (0.0005) / Rhyper (0.0006)
dnn_n16	fm/hqa	fm/hqa	Pavg (0.2513) / M (0.1011) / CVC (0.0983) / EntVar (0.0560) / m (0.0353)	**0:** Pmax / Rlong / Q_{node} / Roverload (0.0000) **Low (nonzero):** $S_{\text{spec}}(P)$ (0.0003) / Var_{ent} (0.0003) / Rhyper (0.0006)
ghz_16	fm/kl/hqa	fm/kl/hqa	Pavg (0.1841) / M (0.1029) / CVC (0.0937) / EntVar (0.0294) / Bk (0.0253)	**0:** Pmax / Rlong / Q_{node} / Roverload (0.0000) **Low (nonzero):** Var_{ent} (0.0005) / Rhyper (0.0006) / Peff (0.0011)
ghz_32	fm/kl/hqa	fm/kl/hqa	Pavg (0.1864) / M (0.1181) / CVC (0.0860) / Bk (0.0574) / EntVar (0.0297)	**0:** Pmax / Rlong / Q_{node} / Roverload (0.0000) **Low (nonzero):** Var_{ent} (0.0003) / Peff (0.0009) / $S_{\text{spec}}(P)$ (0.0012)
ghz_64	fm/hqa	fm/kl/hqa	Pavg (0.2272) / CVC (0.1008) / EntVar (0.0602) / M (0.0431) / Bk (0.0392)	**0:** Pmax / Rlong / Q_{node} / Roverload (0.0000) **Low (nonzero):** $S_{\text{spec}}(P)$ (0.0012) / Rhyper (0.0006) / Peff (0.0008)

Circuit	Pred(best)	Emp(best)	Top–5 indicators ($I_i(c)$)	Worst-side indicators
`mermin_bell_16`	fm	fm	Pavg (0.4002) / CVC (0.1106) / M (0.0348) / S_{avg} (0.0191) / Bk (0.0175)	**0:** Pmax / Rlong / Q_{node} / Roverload (0.0000) **Low (nonzero):** $S_{\mathrm{spec}}(P)$ (0.0006) / Rhyper (0.0006) / Peff (0.0011)
`multiplier_n15`	fm	fm	CVC (0.4022) / Pavg (0.1574) / M (0.0786) / Bk (0.0558) / EntVar (0.0478)	**0:** Pmax / Rlong / Q_{node} / Roverload (0.0000) **Low (nonzero):** Q_{mod} (0.0007) / $S_{\mathrm{spec}}(P)$ (0.0006) / Peff (0.0009)
`multiplier_n45`	fm	fm	CVC (0.4051) / Pavg (0.1520) / M (0.0640) / EntVar (0.0431) / Bk (0.0427)	**0:** Pmax / Rlong / Q_{node} / Roverload (0.0000) **Low (nonzero):** Q_{mod} (0.0003) / $\mathrm{Var}_{\mathrm{ent}}$ (0.0002) / $S_{\mathrm{spec}}(P)$ (0.0007)
`qft_16`	fm/kl/hqa	fm/kl/hqa	Pavg (0.4348) / CVC (0.1170) / M (0.0418) / S_{avg} (0.0234) / EntVar (0.0228)	**0:** Pmax / Rlong / Q_{node} / Roverload (0.0000) **Low (nonzero):** $S_{\mathrm{spec}}(P)$ (0.0003) / $\mathrm{Var}_{\mathrm{ent}}$ (0.0004) / Rhyper (0.0006)
`qft_32`	fm/kl/hqa	fm/kl/hqa	Pavg (0.4384) / CVC (0.1163) / M (0.0431) / S_{avg} (0.0234) / EntVar (0.0215)	**0:** Pmax / Rlong / Q_{node} / Roverload (0.0000) **Low (nonzero):** $S_{\mathrm{spec}}(P)$ (0.0003) / $\mathrm{Var}_{\mathrm{ent}}$ (0.0005) / Rhyper (0.0006)
`qft_64`	fm/kl/hqa	fm/kl/hqa	Pavg (0.4424) / CVC (0.1179) / M (0.0358) / S_{avg} (0.0234) / EntVar (0.0212)	**0:** Pmax / Rlong / Q_{node} / Roverload (0.0000) **Low (nonzero):** $\mathrm{Var}_{\mathrm{ent}}$ (0.0004) / $S_{\mathrm{spec}}(P)$ (0.0008) / Rhyper (0.0006)
`square_root_n18`	fm	fm	CVC (0.4054) / Pavg (0.1699) / M (0.0747) / Bk (0.0519) / EntVar (0.0490)	**0:** Pmax / Rlong / Q_{node} / Roverload (0.0000) **Low (nonzero):** $\mathrm{Var}_{\mathrm{ent}}$ (0.0006) / $S_{\mathrm{spec}}(P)$ (0.0006) / Rhyper (0.0020)
`swap_test_n25`	fm/kl/hqa	fm/kl/hqa	Pavg (0.1804) / CVC (0.0849) / Bk (0.0559) / EntVar (0.0427) / M (0.0244)	**0:** Pmax / Rlong / Q_{node} / Roverload (0.0000) **Low (nonzero):** Peff (0.0010) / $S_{\mathrm{spec}}(P)$ (0.0007) / $\mathrm{Var}_{\mathrm{ent}}$ (0.0012)
`tfim_hamiltonian_sim_16`	fm/kl/hqa	fm/kl/hqa	Pavg (0.2042) / M (0.1029) / CVC (0.0942) / EntVar (0.0325) / Bk (0.0208)	**0:** Pmax / Rlong / Q_{node} / Roverload (0.0000) **Low (nonzero):** $\mathrm{Var}_{\mathrm{ent}}$ (0.0006) / $S_{\mathrm{spec}}(P)$ (0.0008) / Rhyper (0.0006)
`tfim_hamiltonian_sim_32`	fm/kl/hqa	fm/kl/hqa	Pavg (0.2009) / M (0.0851) / CVC (0.0840) / Bk (0.0502) / EntVar (0.0263)	**0:** Pmax / Rlong / Q_{node} / Roverload (0.0000) **Low (nonzero):** $\mathrm{Var}_{\mathrm{ent}}$ (0.0006) / Peff (0.0008) / $S_{\mathrm{spec}}(P)$ (0.0012)
`tfim_hamiltonian_sim_64`	fm/hqa	fm/kl/hqa	Pavg (0.2470) / CVC (0.1008) / M (0.0824) / EntVar (0.0436) / Bk (0.0314)	**0:** Pmax / Rlong / Q_{node} / Roverload (0.0000) **Low (nonzero):** Peff (0.0007) / $\mathrm{Var}_{\mathrm{ent}}$ (0.0007) / Rhyper (0.0006)

Interpretation with emphasis on circuits with clear rankings (Multiplier/Square root) `multiplier_n15`/`multiplier_n45` and `square_root_n18` show clear empirical RCX separations, where FM achieves the minimum RCX and HQA becomes the worst with a large gap (Table 6.1) This " FM-best / HQA-worst " tendency is consistent with the structure-side predictions: for all three circuits, the local Rank Influence Top–5 is dominated by CVC (rank 1) and Pavg (rank 2), followed by M, with Bk and EntVar also appearing in the Top–5 (Table 6.4). Figure 6.1 further makes this correspondence visually explicit: these three circuits exhibit very similar radar-chart shapes—characterized by a strong CVC "spike" with a secondary Pavg component and non-negligible M—indicating a shared structural regime of " strong cluster imbalance (hub dependence) + high intra-layer parallelism, " under which RCX is largely determined by whether a small set of hot qubits and their frequent partners can be co-located. In such hub-dependent repetitive circuits, FM ' s single-qubit move updates can progressively repair locally harmful splits while respecting capacity constraints, making it more likely to reach an RCX-reducing assignment than one-shot aggregation approaches.

Contrast to circuits with identical outcomes across methods Interestingly, the same " shape similarity " also appears on the opposite end: circuits where all methods yield identical RCX (e.g., `bigadder_n16`, `ghz_16/32`, `qft_16/32/64`, `swap_test_n25`, `tfim_hamiltonian_sim_16/32`) share highly similar Top–5 indicator sets and radar profiles within each circuit family. This supports the interpretation that local Rank Influence (and its Top–5 radar shape) is capturing the structural condition under which method differences emerge: CVC-dominant “ hub + repetition ” profiles tend to amplify method differences (as in Multiplier/Square root), whereas more regular/constraint-determined families tend to suppress them, leading to indistinguishable RCX outcomes across partitioners.

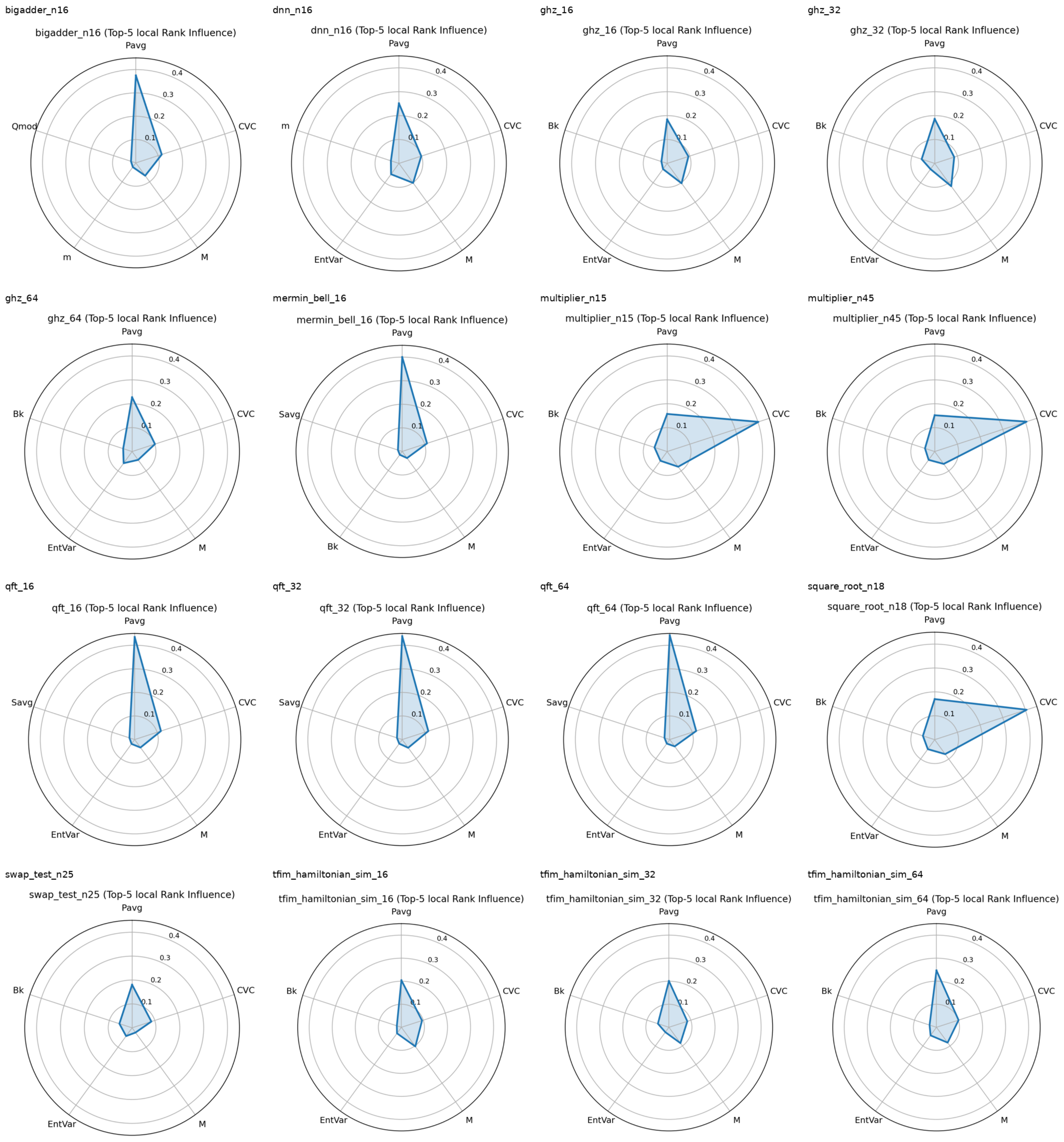


Figure 6.1: Radar charts of the Top–5 circuit-structure indicators for each circuit, ranked by local Rank Influence $I_i(c)$. Each axis corresponds to one of the Top–5 indicators, and the radius indicates its $I_i(c)$ value.

Why a worst method emerges In the three circuits above, HQA tends to become the worst because when the globally aggregated assignment fails to satisfy the condition of "placing the hub and its interaction partner set within the same node," cross-node interactions are repeated across many layers, amplifying RCX. In other words, in circuits with hub-dependent and repetitive structures (high CVC, high Pavg, large M), local misplacements accumulate over layers, and the error of HQA's one-shot decision is more likely to appear as the worst-case outcome.

6.3.24 Impact of Minor Changes in Two-Qubit Gate Placement

For partitioning-method selection to be useful in practice, it is necessary to avoid situations where "a small change in the circuit causes the optimal method to change drastically." From the perspective of quantum circuit design, it is also useful to estimate in advance how much the communication cost can increase in distributed execution when CNOTs are added or replaced. In circuit design and optimization, gate order may be adjusted without changing functionality, and the number of CNOTs may increase or decrease due to ancilla allocation or gate decomposition. What matters is understanding which circuit properties are likely to change under small edits, and distinguishing edits that are unlikely to induce method differences from those that are likely to do so.

Types of minor edits and circuit properties that change easily Minor edits to CNOTs can be organized into at least the following four types. Because each type affects different circuit properties, the conditions under which RCX gaps between methods arise also differ.

1. **Adding a new pair (new edge):** This edit adds a single CNOT to a qubit pair (q_i, q_j) that previously had no CNOT. At the circuit level, it introduces a new two-qubit coupling, increasing the diversity of interacting pairs. As a result, the number of distinct pairs m tends to increase, and the density D also tends to increase. Because the number of interaction pairs that can be split across nodes increases, the number of CNOTs that can potentially become remote also increases. In particular, adding a pair with a large index distance can broaden interactions and amplify method differences.
2. **Repeating an existing pair (duplicate on same pair):** This edit adds CNOTs to an already existing qubit pair (q_i, q_j), increasing the occurrence count of the same pair. At the circuit level, it corresponds to repeated operations on a specific pair. While the number of distinct pairs tends not to change, the total number of two-qubit gates M increases. The repetition-strength indicator Rdup also tends to increase. If this pair is split across nodes after partitioning and assignment, the CNOTs on that pair become remote as many times as they occur, and RCX accumulates accordingly. Therefore, increasing repetition on the same pair tends to enlarge the gap between good and bad assignments.
3. **Hub amplification (hub amplification):** This edit adds or replaces CNOTs such that a specific qubit q_h is included as an endpoint. At the circuit level, it strengthens

the situation where one qubit has CNOT interactions with many partners. As a result, interaction imbalance increases, and EntVar (variance of interaction counts across qubits) and Hmax (maximum concentration) tend to increase. From the post-partitioning perspective, which node q_h is assigned to can strongly dominate RCX. In other words, hub-amplifying edits readily create a situation where assignment differences on a few qubits appear as the presence/absence of many remote CNOTs.

4. **Changing layer structure (layering / parallelism shift):** This edit changes the gate order or layering without changing the set of CNOTs. For example, inserting a CNOT into an existing layer or moving it to a different layer. At the circuit level, the number of two-qubit gates executed simultaneously within the same layer changes, as does the presence of highly parallel layers. As a result, quantities reflecting layer-level congestion, such as GateDensity and parallelism indicators (Pavg/Pmax/Peff), tend to change. When remote CNOTs exist after partitioning, whether they concentrate in specific layers can affect implementation-level waiting and usable parallelism. Therefore, changing the layer structure can alter the temporal concentration of remote operations (including Var_ent), potentially expanding or shrinking method differences.

Positioning as a design guideline This classification suggests that structure indicators provide clues for estimating how much the difficulty of distributed execution can change under small circuit edits. Even when circuit designers must add or replace CNOTs while preserving functionality, being aware of which edit type applies can make it easier to choose edits that are less likely to amplify performance gaps between methods. Examples include the following.

- Avoid edits that increase the variety and spread of interaction pairs (which tend to increase m and D).
- When increasing repetition on the same pair, reshape the circuit so that frequent pairs are more likely to be placed within the same node, mitigating the disadvantage implied by Rdup.
- Avoid edits that strengthen concentration on specific qubits, suppressing increases in EntVar/Hmax.
- Avoid layers that create excessive simultaneous-execution concentration, suppressing biases in GateDensity and parallelism.

This framework has the potential to provide feedback to the design side not only for method selection, but also for identifying "which CNOT placements tend to be unfavorable for distributed execution" as changes in structure indicators.

Future evaluation design To quantify this potential, it is necessary to create a set of circuits where minor edits such as adding new pairs, repeating the same pair, hub amplification, and layer-structure changes are artificially applied, and then systematically measure how much the structure indicators change and how often the recommended method (ranking) switches. This would allow us to verify whether the framework provides robust guidance not only for the input circuits themselves but also for edits introduced during design and optimization.

6.3.25 Limitations of the RCX Metric Under teledata/telegate and cat-entangler Assumptions

This study compared partitioning methods using RCX (the number of Remote CNOTs) as a proxy for communication load in distributed execution. RCX counts the number of CNOTs executed between qubits assigned to different nodes after partitioning. However, the true cost in a distributed implementation (execution time, EPR generation resources, communication waiting) is not necessarily determined uniquely by RCX alone. As shown by distributed arithmetic-circuit implementations on a distributed-memory quantum multicomputer in [31], the realization method for nonlocal operations (teledata/telegate), EPR-generation bottlenecks, communication-interface (transceiver) contention, and limitations in network parallelism can cause cases where the number of remote operations does not monotonically correspond to execution time. Therefore, while our ranking is consistent under implementation conditions where RCX is the dominant factor, its external validity may weaken in regimes where the cost model differs.

Factors that break the correspondence between RCX and true cost At least the following three factors can cause RCX differences not to translate directly into true cost differences.

First, EPR generation can become the bottleneck. While more remote gates consume more EPR pairs, execution time can be dominated not by the number of remote gates but by EPR-generation throughput, success probability, and the degree of parallel generation. Second, contention for communication interfaces (transceivers) or network bandwidth can dominate. If remote operations concentrate simultaneously in particular layers, waiting time can increase even when RCX is similar. Third, the choice between teledata and telegate can change the dominant factors. Teledata tends to be dominated by resource occupancy and congestion due to state transfer, whereas telegate tends to be dominated by round-trip classical communication and sequential-execution constraints [31]. All of these become salient when the implicit implementation assumptions behind using RCX alone are violated.

How changing the cost model can change which structure indicators matter Our discussion and results show that circuit-structure indicators correspond to differences in relative performance across methods. However, this correspondence is interpreted under the assumption that RCX is an appropriate proxy. If the cost model changes, the indicator groups that become important can also change as follows.

When EPR-generation bottlenecks or transceiver contention is strong, the dominant factor may not be total communication volume but the amount of communication required simultaneously at specific times (layers) and its temporal concentration. Accordingly, indicators reflecting intra-layer congestion and simultaneity (GateDensity, Pavg/Pmax/Peff, and quantities capturing layer-wise bias) may become relatively more important. On the other hand, when the teledata/telegate choice dominates, true cost can increase if remote operations concentrate on specific qubits or a small number of nodes, so indicators reflecting degree imbalance or hubness (EntVar, Hmax) may have stronger effects.

Impact of introducing cat-entangler/distangler on the ranking If circuit transformations such as cat-entangler/distangler become available on the distributed-implementation side, the meaning of RCX as a direct count of remote CNOTs may also change. Surveys of distributed quantum computing discuss cat-state-based realizations of remote operations and their resource implications [2, 20]. Such transformations can decouple the implementation cost of inter-node operations from the simple count of "remote CNOTs," so minimizing RCX may no longer be optimal.

From the perspective of structure indicators, introducing cat-entangler/distangler may increase the relative importance of at least the following two structural types. First, structures with strong concentration of interactions on particular qubits. In such cases, circuits that appeared unfavorable under the RCX framework may see their disadvantage mitigated if the implementation can bundle remote operations, and the interpretation of hubness (EntVar, Hmax) may change. Second, structures with high repetition on the same pair. Under RCX, if a frequent pair is split across nodes, RCX increases by the number of repetitions, but if the implementation can bundle repeated remote operations, the way Rdup manifests may change. Furthermore, for circuits where remote operations tend to concentrate simultaneously in specific layers, scheduling of cat-state generation and measurement can become the dominant factor, potentially increasing the importance of GateDensity and parallelism-related indicators.

Chapter 7

Conclusion

7.1 Objectives and Achievements of This Study

This study focuses on quantum circuit partitioning for reducing communication costs, which tend to be dominant in Distributed Quantum Computing (DQC), with particular emphasis on Remote CNOT (RCX) as the communication metric. Rather than pursuing the question of “**which partitioning method is always the best,**” this work aims to clarify “**under what circuit structural conditions relative performance differences among methods are likely to arise.**” Under this objective, the study integrates (i) comparative experiments of multiple partitioning methods, (ii) organization of performance through relative evaluation using Copeland scores, and (iii) explainable analysis that extracts relationships between circuit-structure indicators and relative performance using SHAP-based Rank Influence. Through this integration, the study presents a framework that provides **structure-based rationales for selecting partitioning methods**.

The main achievements of this study are summarized as follows.

- Multiple partitioning methods were compared under identical conditions across several benchmark circuit sets, and it was experimentally clarified that **the relative superiority of methods can vary significantly from circuit to circuit**.
- To address situations that are difficult to capture by evaluations based solely on the method with minimum RCX, relative evaluation using Copeland scores was introduced, enabling **the dominance relations among methods to be summarized into a single metric**.
- The relationship between circuit-structure indicators and relative performance was decomposed into an explainable form using machine learning models and SHAP, allowing **quantitative identification of structural factors that tend to generate performance differences among methods**.
- Furthermore, by introducing a **global Rank Influence–based overall ranking**, the study organized **which structural indicators should be prioritized when selecting partitioning methods**. Conventionally, the choice of partitioning algorithms has often relied on experience-based heuristics regarding which circuit statistics or structural properties to examine. In contrast, this study contributes by presenting, in ranked form, indicators that effectively separate method differences

(Copeland score differences), thereby **systematizing the key observation points for method selection**.

7.2 Main Findings

7.2.1 Relative Performance Strongly Depends on Circuit Structure

Experimental results show that no single partitioning method consistently outperforms all others across all circuits, and that the method (or set of methods) achieving minimum RCX varies depending on the circuit. This observation suggests that partitioning problems should not be framed as the search for a single “best” method, but rather as a problem of **structure-aware method selection**. The Copeland score employed in this study proved effective in organizing such circuit-dependent relative performance, as it aggregates the pairwise dominance relationships among methods.

7.2.2 What the Global Ranking Reveals About Circuit Structures Relevant to Partitioning

The global Rank Influence proposed in this study aggregates, across the entire circuit set, structural indicators that effectively separate differences in method rankings (Copeland score differences), thereby providing **prioritized axes of observation when selecting partitioning methods**. In the bootstrap-aggregated global ranking (Table 6.2), the top five features based on the mean Rank Influence were graph density (D), edge duplication ratio ($Rdup$), entanglement variance ($EntVar$), average edge lifetime, and spectral recursive lambda2 average (e.g., $\bar{\lambda}_2$(rec)). These indicators can be interpreted as structural factors that, on average, tend to generate differences in relative method performance, and an important contribution of this study is that it explicitly identifies “what should be examined first” in terms of concrete indicator names.

In addition, by reporting confidence intervals (CI) and top-K appearance frequencies, the analysis simultaneously conveys the variability of rankings due to finite circuit sets and their stability under resampling. As a result, the global ranking is not merely a list, but rather a **reproducibility-aware organization that can serve as practical evidence for method selection**.

7.2.3 Structural Indicators Capture Conditions Under Which Method Differences Emerge, Not Absolute RCX Values

Through Rank Influence (aggregation of SHAP-based contributions), the study identifies circuit-structure indicators that are associated not so much with the absolute magnitude of RCX itself, but with **the ease with which relative performance differences among methods emerge**. Intuitively,

- circuits with high **interaction density** or strong **pair diversity** tend to retain unavoidable inter-node crossings regardless of the partitioning method, leading to smaller performance differences, whereas

- circuits exhibiting strong **local repetition** or **interaction bias (hotspots)** tend to reflect partitioning and assignment choices more directly in RCX, making method differences more pronounced.

Thus, rather than directly predicting RCX, the structural indicators proposed in this study play a significant role in **visualizing the structural conditions under which method differences are likely to arise**.

7.2.4 Explainability Enables Separation of Global Trends and Circuit-Specific Factors

By using SHAP, the output of a predictor can be decomposed additively into feature-wise contributions, allowing conditional effects to be interpreted in a transparent manner. Leveraging this property, the study defines Rank Influence and separates discussions into **general trends across the circuit set (global)** and **dominant factors specific to individual circuits (local)**. This separation enables explanations such as how, even within the same circuit family, differences in local two-qubit interaction layouts or bias patterns can cause the factors driving method differences to switch.

7.3 Positioning of This Study

Prior research has often focused on proposing new partitioning algorithms or comparing performance on specific benchmark circuits. In contrast, this study aims to **organize the relationship between circuit structure and relative performance across multiple methods**, and to provide structure-based rationales for method selection. In particular, presenting a ranked list of "circuit structures to examine in partitioning" via global ranking offers a shared observational axis for method selection that might otherwise depend heavily on implementation details or heuristics. That is, the contribution of this study lies not only in improving partitioners themselves, but in providing an **analytical framework for understanding and articulating the applicability conditions of partitioning methods**.

7.4 Limitations and Future Work

This study has the following limitations and open issues.

- The analysis assumes static partitioning and primarily uses RCX as the communication cost metric. In practical systems that account for hardware and network effects, other factors such as latency, synchronization overhead, and bandwidth constraints may become dominant.
- Although consistency across methods was ensured through unified two-qubit gate representations and preprocessing, circuit transformations (optimization or rewriting rules) may alter the number or placement of two-qubit gates, potentially affecting the interpretation of RCX as a stable communication metric.

- Analyses based on machine learning models and SHAP depend on the data distribution of the circuit set, and generalization to unseen circuit distributions requires further validation. The global ranking is also an estimate derived from a finite circuit set, and re-evaluation is necessary when expanding or changing the circuit distribution.

7.5 Summary

As Distributed Quantum Computing moves toward practical deployment, decision-making must go beyond simple comparisons of “which partitioning method is better” and instead address **which method should be chosen under which circuit structural conditions**. By combining circuit-structure indicators with explainable analysis, this study quantitatively organizes the conditions under which relative performance differences among methods arise, and provides **structure-based rationales for method selection**. In particular, the global ranking presents “circuit structures to examine in partitioning” with explicit priorities, making it possible to organize observation indicators that had previously been unclear. This framework is expected to serve as a design guideline for partitioning strategies in future DQC systems involving larger, more diverse circuits and more realistic constraints.

Acknowledgment

I would like to express my sincere gratitude to all the faculty members who have supported this thesis research and my research activities in general. First and foremost, I would like to thank my supervisor, Professor Rodney Van Meter, for his continuous support and guidance. Since the very beginning of the four-year integrated master ’ s program, he has kindly taken the time to consult with me, and he provided the decisive opportunity for me to start this research. Moreover, the friendly atmosphere and environment that he has fostered—where research can be carried out enjoyably among the lab members—has been a major reason I was able to continue my work.

I am also deeply grateful to Professor Sakuma, who consistently devoted a great deal of time to me and guided me from the ground up throughout this project. I would also like to thank Professor Nagayama for his highly valuable advice and insightful feedback.

Although it is still uncertain whether the four-year integrated master ’ s program will proceed smoothly and whether I will truly be able to graduate this year, I would like to continue pursuing research in this field going forward. I can only hope that I will not need to write another undergraduate thesis next year (laugh). Finally, I would like to once again express my heartfelt gratitude to everyone who has guided and supported me; this concludes my acknowledgments.

Bibliography

[1] SHAP documentation: shap.TreeExplainer. `https://shap.readthedocs.io/en/latest/generated/shap.TreeExplainer.html`. Accessed: 2026-01-09.

[2] Marcello Caleffi, Michele Amoretti, Davide Ferrari, Daniele Cuomo, Jessica Illiano, Antonio Manzalini, and Angela Sara Cacciapuoti. Distributed quantum computing: a survey. *arXiv preprint arXiv:2212.10609*, 2022.

[3] Waldemir Cambiucci, Regina Melo Silveira, and Wilson Vicente Ruggiero. Hypergraphic partitioning of quantum circuits for distributed quantum computing. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 268–269, 2023.

[4] Fan R. K. Chung. *Spectral Graph Theory*, Vol. 92 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, 1997.

[5] Davide Cuomo, Marcello Caleffi, and Angela Sara Cacciapuoti. Towards a realistic distributed quantum computing architecture. *IEEE Journal on Selected Areas in Communications*, Vol. 38, No. 3, pp. 575–588, 2020.

[6] KaHyPar Developers. Kahypar: Karlsruhe hypergraph partitioning. `https://kahypar.org/`, 2017. Accessed: 2025-12-23.

[7] Oriol Escofet, Ovidi Abadal, Eduard Alarcón, and Mateo Valero. Hungarian qubit assignment for optimized mapping of quantum circuits on modular quantum computing architectures. *arXiv preprint arXiv:2309.12182*, 2023.

[8] Charles M. Fiduccia and Robert M. Mattheyses. A linear-time heuristic for improving network partitions. *IEEE Transactions on Computers*, Vol. C-31, No. 8, pp. 712–720, 1982.

[9] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, Vol. 23, No. 2, pp. 298–305, 1973.

[10] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, Vol. 86, No. 3, p. 032324, 2012.

[11] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, Vol. 40, No. 1, pp. 35–41, 1977.

[12] Dominik Janzing, Lenon Minorics, and Patrick Blöbaum. Feature relevance quantification in explainable AI: A causal problem. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (AISTATS)*, Vol. 108 of *Proceedings of Machine Learning Research*, pp. 2907–2916, 2020.

[13] Brian W. Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, Vol. 49, No. 2, pp. 291–307, 1970.

[14] Ang Li, Samuel Stein, Sriram Krishnamoorthy, and James Ang. Qasmbench: A low-level quantum benchmark suite for nisq evaluation and simulation. *ACM Transactions on Quantum Computing*, 2023.

[15] Daniel Litinski. A game of surface codes: Large-scale quantum computing with lattice surgery. *Quantum*, Vol. 3, p. 128, 2019.

[16] Scott M. Lundberg, Gabriel G. Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles. 2018.

[17] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, Vol. 30, 2017.

[18] Christoph Molnar. *Interpretable Machine Learning*. 3 edition, 2025.

[19] Prakash Murali, Ali Javadi-Abhari, Frederic T. Chong, and Margaret Martonosi. Software architecture for quantum computing design tools. *PRX Quantum*, Vol. 2, No. 1, p. 010357, 2021.

[20] Ali Naghavian, Michele Amoretti, Marcello Caleffi, and Angela Sara Cacciapuoti. Towards fault-tolerant distributed quantum computation (ft-dqc): Taxonomy, recent progress, and challenges. *Computer Science Review*, Vol. 51, p. 100712, 2024.

[21] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, Vol. 69, No. 2, p. 026113, 2004.

[22] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[23] Michael A. Perlin, et al. Quantum circuit cutting with maximum-likelihood tomography. *npj Quantum Information*, Vol. 7, p. 64, 2021.

[24] Daisuke Sakuma, et al. Q-fly: An optical interconnect for modular quantum computers, 2024.

[25] Ryosuke Satoh. Rula: A programming language for ruleset-based quantum repeaters. Master's thesis, Keio University, Graduate School of Media and Governance, 2022. Master's Thesis.

[26] Ryosuke Satoh, Rodney Van Meter, and Michal Hajdušek. Federated graph state preparation on noisy, distributed quantum computers. IPSJ Quantum Software SIG (QS) Workshop / Technical Report, 2020. Listed in the program of IPSJ QS Research Meeting.

[27] Lloyd S. Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker, editors, *Contributions to the Theory of Games, Volume II*, pp. 307–317. Princeton University Press, 1953.

[28] Yulong Tang, Trevor Tomesh, Martin Suchara, Jeffrey Larson, and Margaret Martonosi. Cutqc: Using small quantum computers for large quantum circuit evaluations. *arXiv preprint arXiv:2012.02333*, 2020.

[29] Teague Tomesh, Pranav Gokhale, Victory Omole, Gokul Subramanian Ravi, Kaitlin N. Smith, Joshua Viszlai, Xin-Chuan Wu, Nikos Hardavellas, Margaret R. Martonosi, and Frederic T. Chong. Supermarq: A scalable quantum benchmark suite. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 587–603. IEEE, 2022.

[30] Rodney Doyle Van Meter. *Architecture of a Quantum Multicomputer Optimized for Shor's Factoring Algorithm.* PhD thesis, Keio University, Department of Computer Science, 7 2006.

[31] Rodney Van Meter, W. J. Munro, Kae Nemoto, and Kohei M. Itoh. Arithmetic on a distributed-memory quantum multicomputer. *arXiv preprint arXiv:quant-ph/0607160*, 2006. arXiv:quant-ph/0607160.

[32] Rodney Van Meter, William J. Munro, Kae Nemoto, and Kohei M. Itoh. Arithmetic on a distributed-memory quantum multicomputer. *ACM Journal on Emerging Technologies in Computing Systems*, Vol. 3, No. 4, pp. 1–23, 2008.

[33] Amrutur B Yimsiriwattana and Samuel J Lomonaco. Generalized teleportation and quantum computational networks. In *Quantum Information and Computation II*, Vol. 5436, pp. 360–372. SPIE, 2004.