Entanglement-Efficient Bipartite-Distributed Quantum Computing

Prepared by Gao Dingchao

Reading Seminar

May 23, 2025

References

Pablo Andres-Martinez from TKET team

J-Y. Wu *et al.*, *Entanglement-efficient bipartite-distributed quantum computing*, Quantum **7** (2023).

J. Eisert *et al.*, Phys. Rev. A **62**, 052317 (2000).

Outline

Why Distributed Quantum Computing (DQC)?

- **NISQ limitation:** single QPU constrained by qubit number, coherence, connectivity.
- **Connect QPUs via entanglement** $\Rightarrow$ larger logical device.
- **Bottleneck:** entanglement distribution is costly & probabilistic.
- Goal:

Minimise *EPR pairs* consumption for two-party (bipartite) DQC while retaining universality.

basic protocal

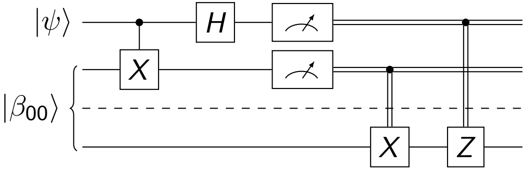
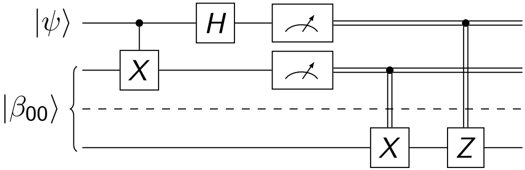


Figure: basic protocol for teleporting a qubit

$$|\psi\rangle\,|\beta_{00}\rangle \to \cdots \to \frac{1}{2}\left[|00\rangle\,|\psi\rangle + |01\rangle\, X\,|\psi\rangle + |10\rangle\, Z\,|\psi\rangle + |11\rangle\, XZ\,|\psi\rangle\right]$$

Protocol in Distributed Quantum Operations



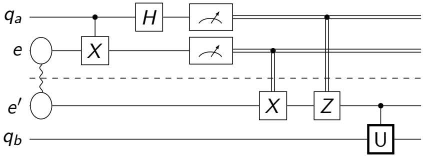


Figure: **Teledata protocol.** The quantum state is first teleported, and then the operation U is applied on the remote system.

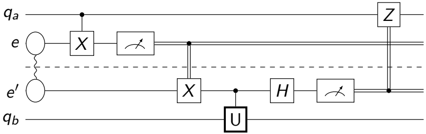


Figure: **Telegate protocol.** The remote party applies a gate U using classical control based on measurement outcomes, without teleporting the quantum state.

Newest Distributed Qubit Experiments (as of May 2025)

Platform	Teledata	Telegate
Superconducting	64 m cryo-bus state teleportation, Qiu *et al.*, 2025	99% SWAP/CZ via detachable cable, Mollenhauer *et al.*, 2025
Trapped Ions	14 km urban-fibre teleportation, Wang *et al.*, 2024	Teleported CZ & Grover's search (2 m), Main *et al.*, 2025
Neutral Atoms	420 km atomic-ensemble entanglement, Luo *et al.*, 2025	Not yet demonstrated

The DQC Problem: Formal Definition

Input: Circuit C on qubits Q, architecture graph $G = (V, E)$.

- Each module $A \in V$ has capacity $\omega(A)$ (data qubits) and $\varepsilon(A)$ (link qubits).
- Non-local gates (across modules) require 1 ebit each.

Output: A distribution $(\varphi, \tilde{C})$ such that:

- Qubit allocation map $\varphi : Q \to V$, $|\varphi^{-1}(A)| \leq \omega(A)$.
- $\tilde{C}$ includes EPR pairs needs for each non-local gate.
- Active Pairs $\leq \varepsilon(A)$ at all times.

Goal: Minimise total ebit usage.

Two Key Subproblems in DQC

- **Qubit Allocation:**
  - Partition qubits across modules respecting $\omega(A)$.
  - Minimise cut edges (i.e., potential non-local interactions).
- **Non-local Gate Distribution:**
  - find a way to implement the non-local gates

TC vs. QCD: Key Differences

- **Gate Types:** TC limits operations to those on-chip; QCD allows all, but nonlocal ones are costly.
- **Objective:** TC minimizes depth; QCD minimizes cross-QPU communication.
- **Optimization:** TC is local (gate-by-gate); QCD is global (qubit grouping).
- **Result:** TC outputs topology-compliant circuits; QCD allows nonlocal gates when needed.

Formal Restatement as Partitioning

Interaction Graph:

- Vertices: qubits Q.
- Edges: between q_1, q_2 if a two-qubit gate in C involves them.

Objective: Partition Q into k disjoint sets

- Each set has size at most $\omega(A)$,
- Minimise the number of cut edges (gates across different sets).
- (Opinion) load-balance: size of each set $\leq (1+\delta)\frac{|V|}{k}$

This is a **capacity-constrained graph partitioning** problem.

Main Partitioning Strategies

- **Local Methods:**
  - Iteratively improve an initial partition.
  - Sensitive to starting configuration.
  - Examples: Kernighan–Lin, Fiduccia–Mattheyses.
- **Global Methods:**
  - Use global graph properties to guide partitioning.
  - Avoid arbitrary initialisation.
  - Examples: Spectral Partitioning, Multilevel Partitioning.

Considerations in Practice

- Hardware-specific capacity limits must be respected.
- Gate commutativity can allow gate packet fusion before partitioning.
- Partitioning quality affects EPR cost and circuit depth.

Definition: Hypergraph Partitioning

Definition: A hypergraph $G = (V, H)$ consists of a set of weighted vertices V and a set of hyperedges H, where a hyperedge $h \subseteq V$ may connect more than 2 vertices.

Connectivity metric ($\lambda - 1$):

$$\lambda_G = \sum_{h \in H} ((\#\text{partitions containing endpoints of } h) - 1)$$

The hypergraph partitioning problem is to find a balanced k-way partition minimizing this metric.

Hypergraph Example

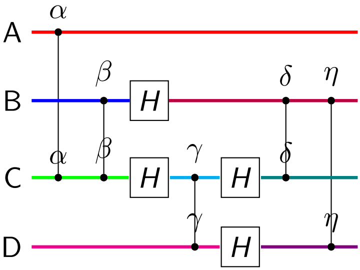


Figure: Example quantum circuit, applying Hadamard gates H and CZ gates.

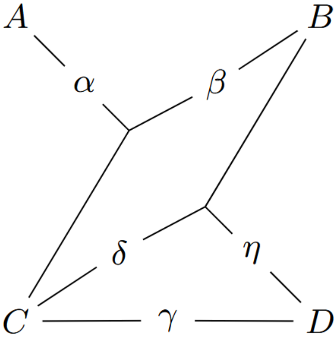


Figure: hypergraph of the example circuit

DQC and Hypergraph Mapping

Correspondence between DQC and hypergraph partitioning:

Hypergraph	**DQC**
Vertex	Qubits $\cup$ CRZ gates
Hyperedge	Wire segment $\{q_i\} \cup \{g_1, \dots\}$
Vertex weight	1 for qubits, 0 for CRZ
$(\lambda - 1)$ metric	EPR cost
Partitioning	Qubit allocation and gate execution

Why Hypergraph, Not Graph?

- Graph cut counts *each gate* separately.
- Hypergraph cut counts *each packet* once.
- Captures the cost structure of shared EJPPs (ebit sharing).

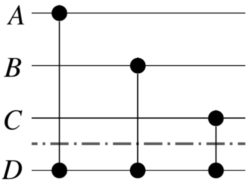


Figure: An example circuit

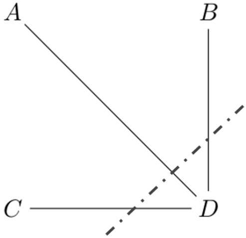


Figure: **Graph**: This partition shown cuts three edges

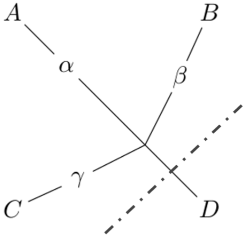


Figure: **Hypergraph**: This partition shown cuts One edges

Key Result

Theorem

For fully connected networks,

- Each valid distribution $\Rightarrow$ a hypergraph partition
- If the circuit's hypergraph can be partitioned cutting n edges, the circuit can be distributed using n ebits.
- *Optimal partition* $\Leftrightarrow$ *Minimum ebit allocation*

Workflow Summary

1. Rebase circuit to {H, RZ, CRZ}
2. Construct hypergraph (qubit/gate vertices, one edge per packet)
3. Use hypergraph partitioner (e.g. KaHyPar) with capacity constraints
4. Read module assignment from partition

Ebit count = Hypergraph cut cost

EJPP protocol

Controlled-unitary CU can be implemented non-locally with **one ebit** using:

- *Starting process*: cat-entangler.
- *Kernel*: local $C_{e,U}$.
- *Ending process*: cat-disentangler.

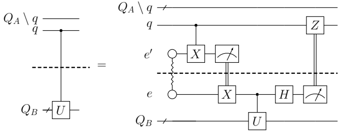


Figure: EJPP scheme.

Distributing vs. Embedding Processes

Definition: Distributing Process

A unitary U is *q-rooted distributable* if it is diagonal/antidiagonal in q and decomposes as $U = \sum_{ij} \Delta_{ij} \, |i\rangle\langle j|_q \otimes V_j \otimes W_j$.

- Implementable with **one ebit** (extends EJPP).

Definition: Embedding Process

A unitary U is *q-rooted embeddable* if
$C_{q,X_e} \, U \, C_{q,X_e} = (L_A \otimes L_B) \, U \, (K_A \otimes K_B)$.

- Enables merging of non-sequential distributing gates.

Theorem 3 (Merging Packing Processes)

Statement: If $P_{q,e}[K_1]$ and $P_{q,e}[K_2]$ are packing processes implementing U_1 and U_2 respectively, then

$$U_2 U_1 = P_{q,e}[K_2 K_1] \quad \text{(still 1 ebit)}.$$

Theorem 3 (Merging Packing Processes)

Statement: If $P_{q,e}[K_1]$ and $P_{q,e}[K_2]$ are packing processes implementing U_1 and U_2 respectively, then

$$U_2 U_1 = P_{q,e}[K_2 K_1] \quad \text{(still 1 ebit)}.$$

Implication

Sequential or *merged-via-embedding* distributable gates can share **one** entangled pair.

Packing Graph

- **Vertices:** distributable nodes (q, t).
- **Edges:** exist if intervening gates are embeddable $\Rightarrow$ *packable*.
- Connected vertices form a **distributable packet** implemented with one packing process.

Conflict Graph and Edge Types

- **Vertices:** indecomposable packing kernels ('D' for distribute, 'B' for embed).
- **Edges:**

DD-type	intrinsic conflict between two distributing options.
DB-type	extrinsic resource conflict distribute vs embed.
BB-type	embed vs embed (resource or structural).

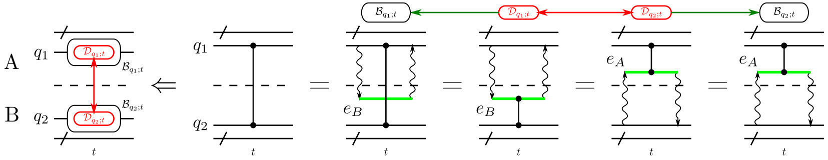


Figure: DD-type conflict example.

Application to UCC Circuits

- Tested on Unitary Coupled-Cluster ansatz.
- Packing heuristic finds **significant entanglement reduction**.
- Achieved constructive upper bound close to theoretical lower bound.

QFT circuit

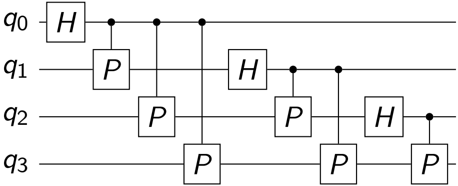
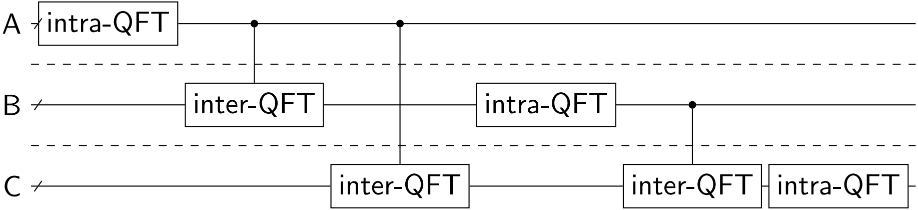


Figure: A example circuit of QFT

Challenges


A
intra-QFT
B
inter-QFT
intra-QFT
C
inter-QFT
inter-QFT
intra-QFT