Entanglement-efficient bipartite-distributed quantum computing

Jun-Yi Wu[1,2], Kosuke Matsui[3], Tim Forrer[3], Akihito Soeda[3,4,5], Pablo Andrés-Martínez[6], Daniel Mills[6], Luciana Henaut[6], and Mio Murao[3]

[1]Department of Physics and Center for Advanced Quantum Computing, Tamkang University, 151 Yingzhuan Rd., New Taipei City 25137, Taiwan, ROC
[2]Physics Division, National Center for Theoretical Sciences, Taipei 10617, Taiwan, ROC
[3]The University of Tokyo, Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033, Japan
[4]Principles of Informatics Research Division, National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
[5]Department of Informatics, School of Multidisciplinary Sciences, SOKENDAI (The Graduate University for Advanced Studies), 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
[6]Quantinuum, Terrington House, 13-15 Hills Road, Cambridge CB2 1NL, UK

In noisy intermediate-scale quantum computing, the limited scalability of a single quantum processing unit (QPU) can be extended through distributed quantum computing (DQC), in which one can implement global operations over two QPUs by entanglement-assisted local operations and classical communication. To facilitate this type of DQC in experiments, we need an entanglement-efficient protocol. To this end, we extend the protocol in [Eisert et. al., PRA, 62:052317(2000)] implementing each nonlocal controlled-unitary gate locally with one maximally entangled pair to a packing protocol, which can pack multiple nonlocal controlled-unitary gates locally using one maximally entangled pair. In particular, two types of packing processes are introduced as the building blocks, namely the distributing processes and embedding processes. Each distributing process distributes corresponding gates locally with one entangled pair. The efficiency of entanglement is then enhanced by embedding processes, which merge two non-sequential distributing processes and hence save the entanglement cost. We show that the structure of distributability and embeddability of a quantum circuit can be fully represented by the corresponding packing graphs and conflict graphs. Based on these graphs, we derive heuristic algorithms for finding an entanglement-efficient packing of distributing processes for a given quantum circuit to be implemented by two parties. These algorithms can determine the required number of local auxiliary qubits in the DQC. We apply these algorithms for bipartite DQC of unitary coupled-cluster circuits and find a significant reduction of entanglement cost through embeddings. This method can determine a constructive upper bound on the entanglement cost for the DQC of quantum circuits.

Jun-Yi Wu: junyiwuphysics@gmail.com
Mio Murao: murao@phys.s.u-tokyo.ac.jp

1 Introduction

In the noisy intermediate-scale quantum era [1], the noise in quantum processes and the decoherence of qubits are two bottlenecks of the scalability of quantum computing. In a single quantum processing unit (QPU), the scalability is constrained by the number, connectivity and coherence time of qubits, which determine the effective width and depth of a quantum circuit. The effective size of a quantum processor can be quantified by its quantum volume [2, 3]. It is believed that the connectivity of qubits on a QPU is a key feature to scale up quantum volume [3]. However, each platform has its intrinsic topological limits on the number of qubits and their connectivity on a QPU.

To overcome these limits, one can extend the qubit connectivity across QPUs employing flying qubits to establish entanglement between two remote QPUs. Several protocols for establishing remote qubit connectivity have been proposed [4–11] and realized in the physical systems of trapped ions [12, 13], neutral atoms [14, 15], NV centers [16], quantum dots [17, 18], and superconducting qubits [19]. One can even establish entanglement between remote trapped-ion qubits with a fidelity comparable with local entanglement generation [20]. In particular, one can even established entanglement between remote trapped-ion modules of servers [21].

All of these technologies facilitate the development of quantum internet [22] in different physical systems. Benefiting from the topology of a quantum internet, one can extend the connectivity of local qubits to build up a hybrid quantum computing system with its

QPUs distributed over a quantum network, e.g. quantum computation over the butterfly network [23]. To distribute universal quantum computing over a quantum internet, one has to implement the universal set of quantum gates over local servers. The only global gate in the universal set that needs to be distributed is the CNOT gate. It is locally implementable with the assistance of entangled pairs according to the protocols in [24, 25]. It shows the feasibility of distributed universal quantum computing through local operations and classical communication (LOCC), if a sufficient amount of entanglement is provided. However, the distribution of entanglement over a quantum network is probabilistic and time-consuming with respect to the coherence time of local qubits. To make distributed quantum computing (DQC) feasible, one has to therefore find an efficient way to exploit the costly entanglement resources.

There are two methods for nonlocal gate handling in DQC with entanglement-assisted LOCC. One is based on quantum state teleportation [24]; the other is based on the remote implementation of quantum gates through entanglement-assisted LOCC [26–29], which is also referred to as "telegate"[30, 31]. In the former scheme, qubits associated with nonlocal gates are teleported forward and backward across parties. In the latter scheme, one implements a nonlocal gate with entanglement-assisted LOCC in a way such that all associated qubits of the gate are kept functional locally without sending them to the other parties. It has been experimentally implemented in [32, 33]. Several telegate protocols are employed for compiling multipartite DQC [34–36].

In particular, the telegate protocol in [26] referred to as the EJPP protocol in this paper can be employed to implement arbitrary nonlocal control unitaries. The DQC schemes based on the state teleportation and the EJPP protocol are compared and combined in a general quantum network [37]. The EJPP protocol consumes only one ebit, that is one maximally entangled pair, which is optimum in entanglement efficiency. Based on this protocol, one can find the optimum partition for a circuit consisting of control-Z gates and single-qubit gates through hypergraph partitioning [34]. The EJPP protocol employed in the partitioning can be extended for sequential control-Z gates and hence reduces entanglement cost [35]. In [35], the EJPP protocol is terminated by each single-qubit gate, and restarted for control-Z gates with a new entangled pair. This leads to an entanglement cost much higher than the theoretical lower bound given by the operator Schmidt rank of a general circuit [38]. To further facilitate DQC in experiments, we develop a DQC protocol with a higher entanglement efficiency, in which we introduce a notion of entanglement-assisted packing processes for bipartite DQC through an extension of the EJPP protocol. Note that this method is extended to multipartite DQC with some additional treatments and is employed as the core nonlocal-gate-handling subroutine for modular quantum computing under network architectures [39]. A similar application of this method can be also employed for other modular compilation architectures [31, 36, 40–44].

As two particular types of entanglement-assisted packing processes, we introduce distributing processes and embedding processes as the two fundamental building blocks for DQC. In each distributing process, one ebit of entanglement is employed to achieve an operation equivalent to a global unitary, implemented by packing local gates using entanglement-assisted LOCC. The number of distributing processes equals the entanglement cost for DQC. To reduce the entanglement cost, we introduce embedding processes of intermediate gates to merge two non-sequential distributing processes into a single distributing process. Each embedding process can therefore save one ebit without changing the distributability of the circuit. The distributability and embeddability of gates in a circuit reveal the underlying entanglement cost for DQC employing packing processes. Such a packing method provides a tighter constructive upper bound on the entanglement cost for DQC. It can significantly enhance the entanglement efficiency of DQC.

The rest of the paper is structured as follows. In Section 2, we review the EJPP protocol. In Section 3, we introduce the entanglement-assisted packing processes and two special types of processes, namely, the distributing and embedding. The conflicts between embeddings are addressed. The packing graph and conflict graph are introduced to represent the distributing and embedding structure. In Section 4, we derive the algorithms for the identification of packing graphs and conflict graphs for quantum circuits consisting of one-qubit and two-qubit gates. In Section 5, we demonstrate the enhancement of entanglement efficiency in unitary coupled-cluster (UCC) circuits [45, 46] employing our method. In Section 6, we conclude the paper.

2 EJPP protocol

As shown in Fig. 1, a controlled-unitary gate C_U on the left-hand side can be implemented through entanglement-assisted LOCC [47] with only one maximally entangled state on the right-hand side according to the EJPP protocol presented in [26]. The controlled unitary on the left side has a controlling qubit q on the local quantum processor A and a target unitary U acting on the multi-qubit subsystem Q^B on the local quantum processor B. In the EJPP protocol, there is a pre-shared maximally-entangled pair $\{e, e'\}$, on which one implements local controlled unitaries and global measurement-controlled gates through classical communication. It consumes only one ebit, i.e. a maximally-entangled pair, which is the minimum nec-

essary amount of entanglement for a distributed implementation of a controlled unitary determined by its operator Schmidt rank [38]. It means that the EJPP protocol is most entanglement-efficient for a controlled unitary.

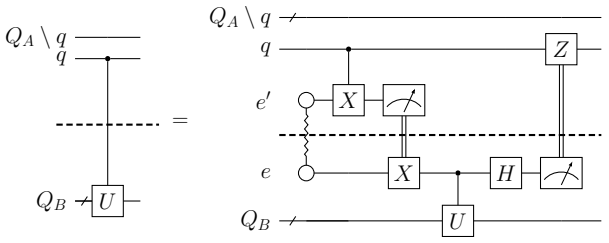


Figure 1: The quantum circuit on the left side is a controlled-unitary gate. The quantum circuit on the right side is a distributed implementation of the controlled-unitary gate according to the EJPP protocol, where $Q_A \cup \{e'\}$ and $Q_B \cup \{e\}$ are qubits in the local quantum processor A and B, respectively. The qubits $\{e', e\}$ are the auxiliary memory qubits that store a pre-shared maximally entangled pair, namely one ebit entanglement.

The EJPP protocol can be divided into three parts, namely the start, kernel, and end. The starting and ending are constructed with fixed gates, while the kernel can be tailored for the implementation of particular unitaries. Here, we define the two fixed parts as the *starting process*, and the *ending process*, which correspond to the *cat-entangler* and the *cat-disentangler* in [48], respectively.

Definition 1 (Starting process and ending process) *The quantum circuit on the right-hand side of Fig. 2 (a) and (b) are referred to as a* starting process *and an* ending process *rooted on q, respectively, which are denoted by the symbols on the left-hand side. The auxiliary qubits e and e' are initialized as a Bell state $(|00\rangle + |11\rangle)/\sqrt{2}$ in the starting process.*

The starting process duplicates the computational-basis components of an input state $|\psi\rangle$ on q to an entangled state

$$\begin{aligned} S_{q,e}(|\psi_q\rangle) &= C_{q,X_e} |\psi_q, 0_e\rangle \\ &= \langle 0_q|\psi_q\rangle |0_q, 0_e\rangle + \langle 1_q|\psi_q\rangle |1_q, 1_e\rangle , \end{aligned} \quad (1)$$

where C_{q,X_e} is a control-X gate with the control qubit q and target qubit e, and S is a linear isometry operator $S \in \mathcal{L}(\mathcal{H}^{\mathrm{q}} \to \mathcal{H}^{\mathrm{q}} \otimes \mathcal{H}^{\mathrm{e}})$,

$$S_{q,e} : |i\rangle_q \mapsto |i\rangle_q |i\rangle_e , \text{ for all } i \in \{0, 1\}. \quad (2)$$

On the other hand, by applying an ending process to the state $|\Psi_{q,e}\rangle = S_{q,e}(|\psi_q\rangle)$, one obtains the partial trace of $C_{q,X_e} |\Psi_{q,e}\rangle$

$$E_{q,e}(|\Psi_{q,e}\rangle) = \mathrm{tr}_e \left(C_{q,X_e} |\Psi_{q,e}\rangle \langle\Psi_{q,e}| C_{q,X_e}\right). \quad (3)$$

The ending process of $|\psi\rangle$ is therefore the inverse of the starting isometry S, which restores the original state

$$E_{q,e} \circ S_{q,e}(|\psi_q\rangle) = |\psi_q\rangle . \quad (4)$$

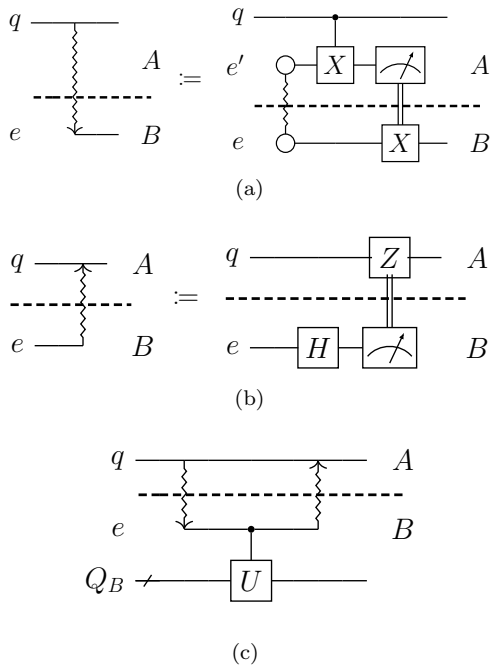


Figure 2: The starting and ending processes: the symbol on the left side represents the operation given by the quantum circuit on the right side. A working qubit q and an auxiliary qubit e' belong to a local QPU A, while and an auxiliary qubit e belongs to another local QPU B. (a) The starting process. (b) The ending process. (c) The protocol in [26] for distributed implementation of controlled unitary gates.

Inserting a local unitary $C_{e,U_{Q_B}}$ acting on $\{e\} \cup Q_B$ between the starting and ending processes, one can implement a global controlled unitary $C_{q,U_{Q_B}}$ with local gates according to the following equality

$$\begin{aligned} C_{q,U_{Q_B}} |\psi_q\rangle \langle\psi_q| C_{q,U_{Q_B}} &= E_{q,e} \circ C_{e,U_{Q_B}} \circ S_{q,e}(|\psi_q\rangle) \\ &= \mathrm{tr}_e \left(|\widetilde{\Psi}_{q,e}\rangle \langle\widetilde{\Psi}_{q,e}|\right), \end{aligned} \quad (5)$$

where

$$|\widetilde{\Psi}_{q,e}\rangle = C_{q,X_e} C_{e,U_{Q_B}} C_{q,X_e} |\psi_q, 0_e\rangle . \quad (6)$$

Employing the squiggly arrows for starting and ending processes, the protocol in [26] can be represented as Fig. 2 (c).

3 Packing distributable unitaries

3.1 Entanglement-assisted packing processes

Beyond controlled-unitary gates, one can extend the EJPP protocol [26] to locally implement a swapping gate with the assistance of two-ebit of entanglement, which is shown in Fig. 3. There are two steps to construct the distributed implementation. In the first step, we extend the protocol through the replacement of the local control unitary $C_{e,U_{Q_B}}$ in Fig. 2 (c) by

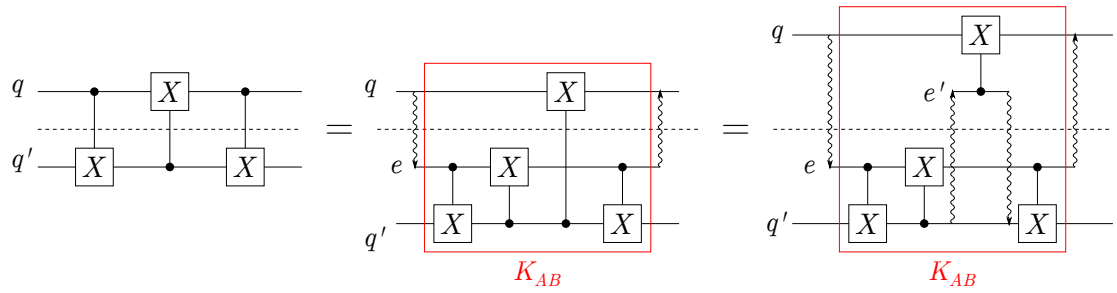


Figure 3: Distributed implementation of swapping with two entanglement-assisted LOCCs

a global unitary K_{AB}. In the second step, we implement the EJPP protocol for the remaining global controlled-X gate starting and ending on the qubit q'.

In this example, the process in the first step is implemented with a kernel K_{AB} sandwiched by the starting and ending processes. The kernel K_{AB} is a global unitary acting on the working qubit $\{q, q'\}$ and the auxiliary qubit e. In general, K_{AB} can be either local or global. For a global kernel K_{AB}, one needs further steps to locally implement the global gates in the kernel with the starting and ending processes. Once all the global gates are distributed, then the distribution of the quantum circuit is finished. Such an extension of the EJPP protocol through the replacement of local controlled-unitary gates by a global kernel K_{AB} is therefore a general building block for the distributed quantum computing based on the starting and ending processes. In this paper, we refer to these building blocks as *entanglement-assisted packing processes*.

Definition 2 (Entanglement-assisted packing process) *An* entanglement-assisted packing process $\mathcal{P}_{q,e}[K]$ *with a kernel K rooted on a qubit q assisted by two auxiliary qubits (e, e') is a quantum circuit that packs a unitary K by the entanglement-assisted starting and ending processes rooted on q (see Fig. 4(a)),*

$$\mathcal{P}_{q,e}[K] := E_{q,e} \circ K \circ S_{q,e}, \tag{7}$$

In short, we call $\mathcal{P}_{q,e}[K]$ a q-rooted e-assisted packing process.

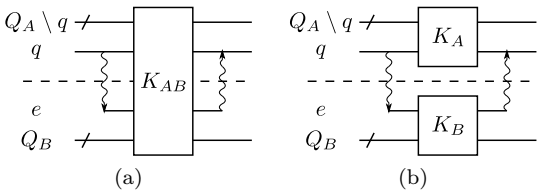


Figure 4: Entanglement-assisted packing process. (a) A q-rooted e-assisted packing process with a kernel K_{AB}. (b) A q-rooted e-assisted $A|B$-distributing process with a local kernel $K_A \otimes K_B$.

In the starting process, one needs a maximally entangled pair, namely one ebit, and two auxiliary memory qubits e' and e. We refer to the first auxiliary qubit e' as the *root auxiliary qubit* and the second auxiliary qubit e as the *packing auxiliary qubit*. The memory space of a root auxiliary qubit can be released immediately after the starting process, while a packing auxiliary qubit is occupied for the kernel K_{AB} until the ending process. The required memory space on each local QPU is therefore one root auxiliary qubit for initializing a starting process and several packing auxiliary qubits for parallel packing processes. In DQC based on entanglement-assisted packing processes, the external resources that can be optimized are therefore the entanglement cost and the number of packing auxiliary qubits. By definition, an entanglement-assisted packing process consumes one-ebit entanglement and occupies one packing auxiliary qubit e. A packing auxiliary qubit e implies one-ebit preshared entanglement between (e, e'). The number of packing processes in a compilation of a unitary U is therefore exactly the number of maximally entangled pairs and the number of packing auxiliary memory qubits that are required for the DQC implementation of U. This number is always a constructive upper bound on the entanglement cost for the DQC implementation of U based on entanglement-assisted LOCC. A tighter bound can be determined if one finds a more entanglement-efficient compilation of a circuit.

In general, a packing process is a completely positive and trace-preserving map (CPTP) represented by a set of two Kraus operators. For certain types of kernels, the two Kraus operators are equivalent up to a global phase φ. In this case, a packing process is equivalent to a unitary acting on the working qubits. We call $\mathcal{P}_{q,e}[K]$ a *canonical unitary-equivalent packing process*, if the global phase φ is equal to $\varphi = 0$. The condition for a packing process being canonically unitary-equivalent is discussed in Appendix A, where the packing processes with general kernels that leads to Kraus operators are also discussed. In the rest of this paper, we consider this ideal case and treat all packing processes as canonically unitary-equivalent.

The canonical unitary-equivalent packing processes are the building blocks of DQC. They have an important property that two sequential packing processes can be merged into one packing process according to the following theorem.

Theorem 3 (Merging of packing processes)

Let $\mathcal{P}_{q,e}[K_{1,2}]$ be two q-rooted e-assisted packing processes that implement two unitaries $U_{1,2}$, respectively,

$$U_i = \mathcal{P}_{q,e}[K_i] \quad \text{for } i = 1, 2. \tag{8}$$

The product of unitaries U_2U_1 can be then implemented by a single q-rooted e-assisted packing process with the kernel K_2K_1

$$U_2U_1 = \mathcal{P}_{q,e}[K_2]\mathcal{P}_{q,e}[K_1] = \mathcal{P}_{q,e}[K_2K_1]. \tag{9}$$

Proof: See Appendix D.1. ■

This theorem allows us to implement the unitary U_2U_1 through the packing of the two kernels K_2K_1 into one packing process and hence save one ebit. To decompose a circuit in DQC, two types of packing processes are needed, namely the distributing processes and embedding processes, which will be introduced in Section 3.2 and 3.3, respectively.

3.2 Distributing process

Our ultimate goal is to find an efficient implementation of a quantum circuit with packing processes that contain only local kernels. To this end, we need to decompose a circuit into distributing processses, which are defined as follows.

Definition 4 (Distributing process) *A unitary U is q-rooted distributable (or distributable on q) over two local systems $A|B$, if there exists a q-rooted entanglement-assisted process with a local kernel with respect to $A|B$ (see Fig. 4 (b)),*

$$U = \mathcal{P}_{q,e}[K_A \otimes K_B]. \tag{10}$$

*The kernel describes a $(A|B)$-*distributing rule $\mathcal{D}_q$ *of U,*

$$\mathcal{D}_q(U) = K_A \otimes K_B. \tag{11}$$

The process $\mathcal{P}_{q,e}[K_A \otimes K_B]$ is called a q-rooted $(A|B)$-distributing process *of U.*

Explicitly, the EJPP protocol in Fig. 1 is a distributing process, while the DQC implementation of the swapping gate in Fig. 3 involves two nested distributing processes.

Distributing processes are the backbones of distributed quantum computing. The solution for entanglement-assisted quantum computation is to reveal the distributability structure of a quantum circuit, namely to find a decomposition of quantum circuits that consists of distributable unitaries.

In general, a unitary U on the left-hand side of Fig. 5 can be written in the following form

$$U = \sum_{i,j} |i\rangle\langle j|_q \otimes U_{ij}^{(\bar{q})}, \tag{12}$$

where $\bar{q}$ denotes the set of qubits complement to $\{q\}$. The q-rooted $A|B$-distributability can be determined with the following necessary and sufficient condition.

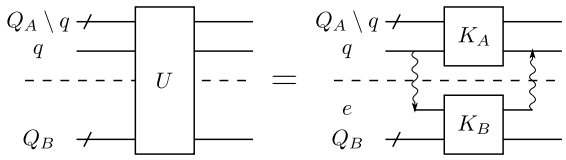


Figure 5: A q-rooted distributable unitary.

Theorem 5 (Distributability condition) *A unitary U is q-rooted distributable over $A|B$, if and only if U is diagonal or anti-diagonal on q and given in the following form,*

$$U = \sum_{i,j} \Delta_{ij} |i\rangle\langle j|_q \otimes U_j^{(\bar{q})} \quad \text{with} \quad \Delta_{ij} = \delta_j^i \text{ or } \delta_j^{i\oplus 1}, \tag{13}$$

where $U_j^{(\bar{q})}$ is local with respect to $A|B$

$$U_j^{(\bar{q})} = V_j^{(Q_A \setminus q)} \otimes W_j^{(Q_B)}. \tag{14}$$

The unitary V_j and W_j act on the qubits $Q_A \setminus q$ and Q_B, respectively. Without loss of generality, we assume that $q \in Q_A$ is on the QPU A. The corresponding distributing rule described by the kernel $K_A \otimes K_B$ can be constructed as

$$K_A = \sum_{i,j\in\{0,1\}} \Delta_{ij} |i\rangle\langle j|_q \otimes V_j^{(Q_A\setminus q)}, \tag{15}$$

$$K_B = \sum_{i,j\in\{0,1\}} \Delta_{ij} |i\rangle\langle j|_e \otimes W_j^{(Q_B)}. \tag{16}$$

Proof: See Appendix D.2 ■

The q-rooted distributability of a unitary U depends on its representation in the computational basis of the root qubit. Note that the distributability condition can also be formulated in another basis of the root qubit, if one transforms the control qubit in the starting process and the correction gate Z in the ending process into the corresponding basis. However, such variant starting processes and ending processes can not be merged with the standard ones defined in Definition 1. We therefore stick to the definition of distributability with respect to the standard starting and ending processes, and determine the distributability in the computational basis of the root qubit with Theorem 5.

As a result of Theorem 5, some trivial distributable gates can be determined as follows.

Corollary 6 *The following unitaries are all q-rooted distributable.*

1. *Single-qubit unitaries on q that are diagonal or anti-diagonal.*
2. *Two-qubit controlled-unitary gates that have q as their control qubit.*

The distributability of these gates does not depend on the bipartition $A|B$.

As a result of this corollary, a two-qubit control-phase gate $C_V(\theta)$ acting on $q_{1,2}$ is $q_{1,2}$-rooted distributable,

$$\begin{aligned} C_V(\theta) &= |0\rangle\langle 0|_{q_1} \otimes \mathbb{1}_{q_2} + |1\rangle\langle 1|_{q_1} \otimes V_{q_2}(\theta) \\ &= \mathbb{1}_{q_1} \otimes |0\rangle\langle 0|_{q_2} + V_{q_1}(\theta) \otimes |1\rangle\langle 1|_{q_2}, \end{aligned} \tag{17}$$

where $V(\theta)$ is a phase gate

$$V(\theta) := |0\rangle\langle 0| + \exp(\mathbb{i}\,\theta)\,|1\rangle\langle 1|. \tag{18}$$

According to Theorem 3, one can implement two sequential q-rooted distributable unitaries U_2U_1 through the packing of their distributing kernels $\mathcal{D}_q(U_2)\mathcal{D}_q(U_1)$ within a single packing process consuming only one ebit. Such packing can be even possible for two non-sequential distributable unitaries through the another type of packing processs, namely embeddings.

3.3 Embedding process

In this section, we introduce embedding processes, which enable the packing of non-sequential distributing processes through the embedding of the intermediate unitary between them.

Definition 7 (Embedding process) *A unitary U is* q-rooted embeddable *with respect to a bipartition $A|B$, if there exists a decomposition of U, $U = \prod_i V_i$ (Fig.6 (a)), and corresponding kernels $\mathcal{B}_q(V_i)$,*

$$\mathcal{B}_q(V_i) = (L_{i,A} \otimes L_{i,B})\, V_i\, (K_{i,A} \otimes K_{i,B}), \tag{19}$$

where $L_{i,A}$ and $K_{i,A}$ act on local qubits Q_A, and $L_{i,B}$ and $K_{i,B}$ act on local qubits $Q_B \cup \{e\}$ (Fig.6 (b)), such that

$$V_i = \mathcal{P}_{q,e}[\mathcal{B}_q(V_i)]. \tag{20}$$

The unitary V_i is an embedding unit *of U. The kernel $\mathcal{B}_q(V_i)$ describes a* q-rooted embedding rule *of V_i. The process $\mathcal{P}_{q,e}[\mathcal{B}_q(V_i)]$ is a* q-rooted e-assisted embedding process *of V_i.*

According to Theorem 3, a q-rooted embeddable unitary can be packed into a packing process with additional local kernels $K_{i,A} \otimes K_{i,B}$ and $L_{i,A} \otimes L_{i,B}$ sandwiching the original embedding units V_i (see Fig. 6 (c)),

$$U = \mathcal{P}_{q,e}[\prod_i \mathcal{B}_q(V_i)]. \tag{21}$$

Note that V_i can be a nonlocal unitary that still needs to be distributed.

An example of embedding processes can be found in the first step of DQC implementation of swapping in Fig. 3, where a CNOT gate is packed into a q-rooted embedding process. As shown in Fig.7, an additional local CNOT gate is introduced in the kernel of the embedding process, while the nonlocal CNOT gate remains unaltered in the circuit. Such an embedding process in the swapping circuit can merge two non-sequential distributing processes into one packing process according to Theorem 3, as illustrated in Fig. 8 (a).

Formally, let U be a q-rooted embeddable unitary with the embedding rule $\mathcal{B}_q(U) = (M_A \otimes M_B)U(L_A \otimes L_B)$ and placed between two q-rooted distributable unitaries $D_{1,2}$, which is shown in Fig. 8 (b). According to Theorem 3, one can pack the two non-sequential distributable unitaries $D_{1,2}$ into a packing process through the embedding of U,

$$\begin{aligned} D_2\, U\, D_1 &= \mathcal{P}_{q,e}[\mathcal{D}_q(D_2)\mathcal{B}_q(U)\mathcal{D}_q(D_1)] \\ &= \mathcal{P}_{q,e}[\widetilde{K}_2 U \widetilde{K}_1]. \end{aligned} \tag{22}$$

where $\mathcal{D}_q(D_i) = K_{i,A} \otimes K_{i,B}$ are the distributing rules for $D_{1,2}$ and $\widetilde{K}_1 = L_A K_{1,A} \otimes L_B K_{1,B}$. In the end, the embeddable unitary U is embedded into a merged packing process with the kernel $\widetilde{K}_2 U \widetilde{K}_1$, where $\widetilde{K}_2 = K_{2,A}M_A \otimes K_{2,A}M_B$ are both local unitaries and the distributability of U remains untouched.

The utility of embedding is not limited to packing processes rooted on a single qubit. It can be extended to simultaneous embedding processes rooted on multiple qubits. As shown in Fig. 9 (a), the unitary implemented by three CNOT gates can be jointly embedded in four packing processes which start simultaneously with four starting processes rooted on $\{q_1, q_2, q_3\}$ assisted by $\{e_1, e_1', e_2, e_3\}$ and end simultaneously with four ending processes. In general, one can simultaneously embed a unitary into a joint packing process $\{\mathcal{P}_{q_i,e_i}[K_i]\}_i$ rooted on multiple qubits $\{q_1, ..., q_k\}$, which is referred to as a $\{q_1, ..., q_k\}$-rooted joint embedding and illustrated in Fig. 9 (b).

Definition 8 (Joint embedding) *A unitary U is jointly embeddable on a set of root qubits $\mathbb{Q} = \{q_1, ..., q_k\}$, if there exists a decomposition $U = \prod_i V_i$ and corresponding joint embedding rules*

$$\mathcal{B}_{\mathbb{Q}}(V_i) = (L_{i,A} \otimes L_{i,B})V_i(K_{i,A} \otimes K_{i,B}), \tag{23}$$

such that each embedding unit can be implemented by a joint entanglement-assisted process $\mathcal{P}_{\mathbb{Q},\mathbb{A}}$ rooted on a set of qubits $\mathbb{Q}$ and assisted with a set of auxiliary qubits $\mathbb{A} = \{e_1, ..., e_k\}$,

$$V_i = \mathcal{P}_{\mathbb{Q},\mathbb{A}}[\mathcal{B}_{\mathbb{Q}}(V_i)] \text{ with } \mathcal{P}_{\mathbb{Q},\mathbb{A}} := \mathcal{P}_{q_k,e_k} \circ \cdots \circ \mathcal{P}_{q_1,e_1}. \tag{24}$$

With joint embedding, one can simultaneously merge non-sequential distributing processes on multiple qubits. Note that a joint embedding can have packing processes on the same root qubit multiple times. As shown in Fig. 9, the joint embedding of U on $\{q_1, q_2, ..., q_k\}$ has two packing processes rooted on the same qubit q_1 and assisted with two auxiliary qubits e_1 and e_1', respectively.

In general, it is non-trivial to derive a necessary condition for the embeddability of U, since one has

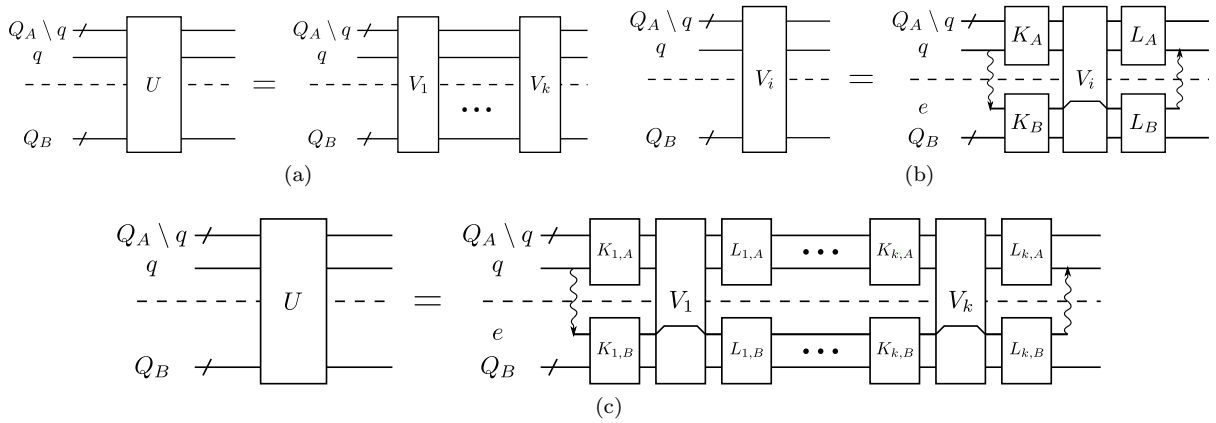


Figure 6: Embeddable unitary: (a) The decomposition of U as a product of embedding units. (b) The embeddability of each embedding unit. (c) The embedding of U.

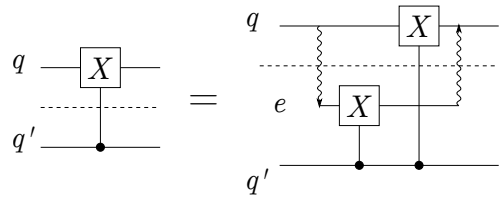


Figure 7: The embedding process of a CNOT gate

to explore all the possible decompositions of U. However, one can still derive a primitive embedding rule $\mathcal{B}_{\mathbb{Q}}(U)$ as a sufficient condition for the $\mathbb{Q}$-rooted embeddability of U as follows.

Corollary 9 (Primitive embedding rule) *A unitary U is $\mathbb{Q}$-rooted embedabble with an embedding rule $\mathcal{B}_{\mathbb{Q}}(U) = (L_A \otimes L_B)\, U\, (K_A \otimes K_B)$, if*

$$C_{\mathbb{Q},X_{\mathbb{A}}}\, U\, C_{\mathbb{Q},X_{\mathbb{A}}} = (L_A \otimes L_B)\, U\, (K_A \otimes K_B), \tag{25}$$

where $C_{\mathbb{Q},X_{\mathbb{A}}}$ is a sequence of control-X gates on $\{(q_i, e_i) : q_i \in \mathbb{Q}, e_i \in \mathbb{A}\}$,

$$C_{\mathbb{Q},X_{\mathbb{A}}} = \prod_{q_i \in \mathbb{Q}, e_i \in \mathbb{A}} C_{q_i,X_{e_i}}. \tag{26}$$

This embedding rule is called the primitive $\mathbb{Q}$-rooted embedding rule *of U.*
Proof: This corollary is a result of Corollary 39 in Appendix A. ■

Due to the diagonal and anti-diagonal condition for distributability in Theorem 5, one can show that a q-rooted distributable is always q-rooted embeddable.

Theorem 10 (Distributability and embeddability) *If a unitary U is q-rooted distributable, then U is also q-rooted embeddable with the following embedding rules,*

1. *for U being diagonal on q,*

$$U = \mathcal{P}_{q,e}[U]; \tag{27}$$

2. *for U being anti-diagonal on q,*

$$U = \mathcal{P}_{q,e}[UX_e] = \mathcal{P}_{q,e}[X_e U], \tag{28}$$

where X_e is an X gate on the auxiliary qubit e.

Proof: According to Theorem 5, a q-rooted distributable unitary U is diagonal or anti-diagonal on q. One can show that, for U being diagonal on q,

$$C_{q,X_e} U C_{q,X_e} = U, \tag{29}$$

while for U being anti-diagonal on q,

$$C_{q,X_e} U C_{q,X_e} = U X_e = X_e U. \tag{30}$$

As a result of Corollary 9, U is embeddable on q with the embedding rules $\mathcal{B}_q(U) = C_{q,X_e} U C_{q,X_e}$. ■

In general, the q-rooted embeddability is not a sufficient condition for q-rooted distributability, since there are some unitaries which are q-rooted embeddable but not q-rooted distributable. However, if the unitary we consider is a local unitary, one can show that embeddability is equivalent to distributability.

Corollary 11 (Embeddability of local unitaries) *A local unitary is q-rooted embeddable, if and only if it is q-rooted distributable.*
Proof: Without loss of generality, we assume that q is on the system A. If a local unitary $U_A \otimes \mathbb{1}_B$ is q-rooted embeddable, there exists an embedding rule,

$$\begin{aligned} U_A \otimes \mathbb{1}_B &= \mathcal{P}_{q,e}[(M_A \otimes M_B)\,(U_A \otimes \mathbb{1}_B)\,(L_A \otimes L_B)], \\ &= \mathcal{P}_{q,e}[(M_A U_A L_A) \otimes (M_B L_B)]. \end{aligned} \tag{31}$$

The unitary U_A is therefore q-rooted distributable with the local kernel $(M_A U_A L_A) \otimes (M_B L_B)$, which shows that q-rooted embeddability is a sufficient condition for q-rooted distributability of a local unitary. According to Theorem 10, q-rooted embeddability is also a necessary condition for q-rooted distributability. ■

According to Theorem 5, this corollary implies that a local unitary is embeddable on q, if and only if it is diagonal or anti-diagonal on q.

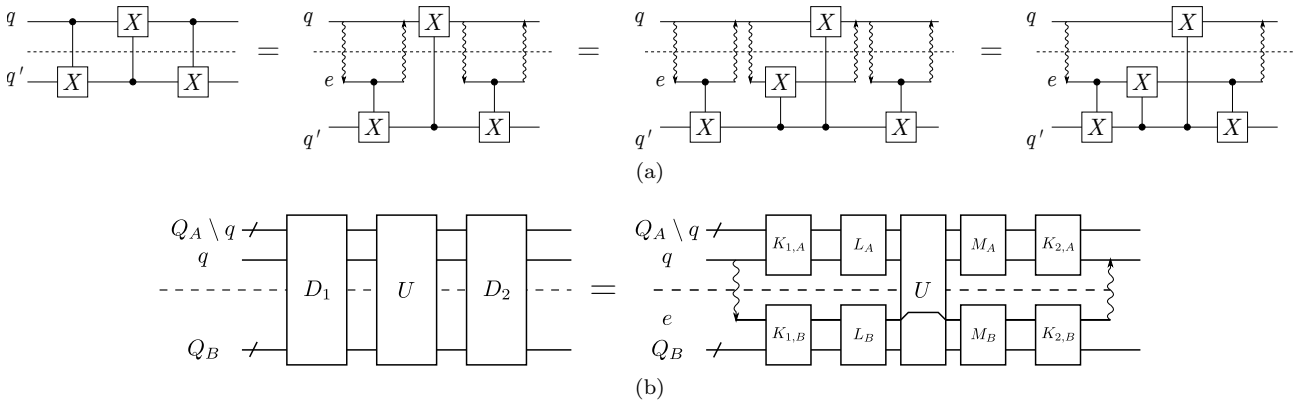


Figure 8: (a) The embedding process between two distributing processes in the DQC of swapping. (b) Packing of two distributing processes through embedding: $D_{1,2}$ are q-rooted distributable unitaries. U is q-rooted embeddable with the rule $\mathcal{B}_q(U) = M_A \otimes M_B\, U\, L_A \otimes L_B$.

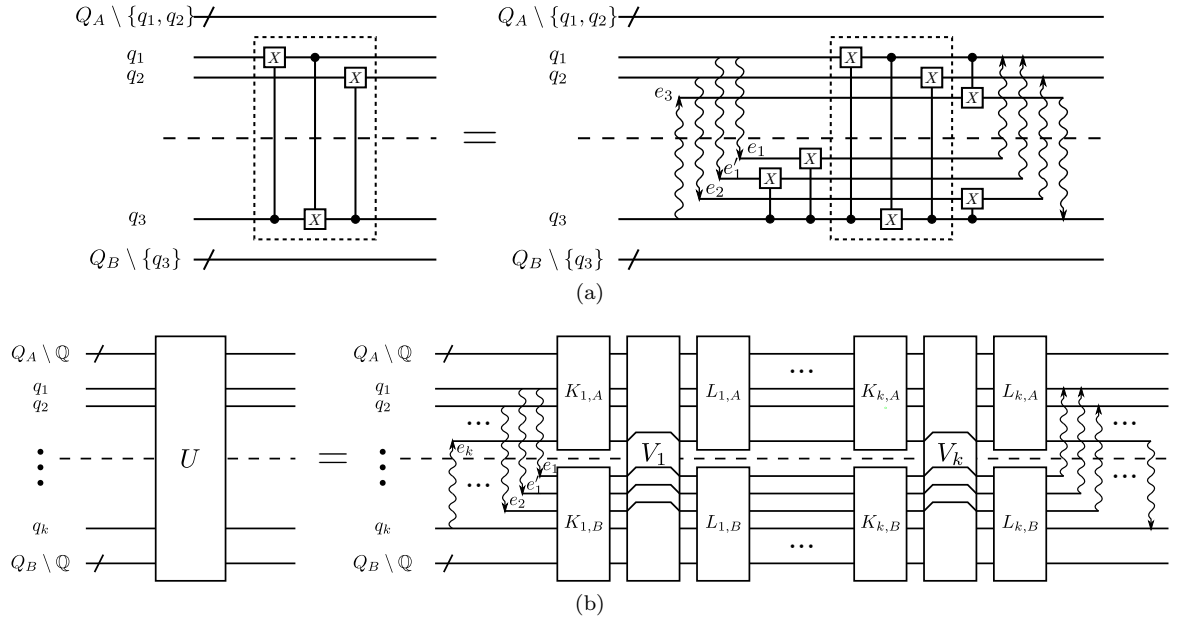


Figure 9: Joint embedding. (a) A circuit that can be jointly embedded on the qubits $\{q_1, q_1, q_2, q_3\}$. (b) A joint embedding rooted on $\mathbb{Q} = \{q_1, q_1, q_2, ..., q_k\}$ assisted with the auxiliary qubits $\mathbb{A} = \{e_1, e_1', e_2, ..., e_k\}$.

3.4 Compatibility among embeddings

Let U be embeddable on two different sets of root qubits $\mathbb{Q}$ and $\mathbb{Q}'$ with the following embedding rules, respectively

$$\mathcal{B}_{\mathbb{Q}}(U) = N_A \otimes N_B\, U\, N_A' \otimes N_B', \tag{32}$$
$$\mathcal{B}_{\mathbb{Q}'}(U) = O_A \otimes O_B\, U\, O_A' \otimes O_B'. \tag{33}$$

Suppose there are two $\mathbb{Q}$-rooted packing processes $\mathcal{P}_{\mathbb{Q},\mathbb{A}}[L]$ and $\mathcal{P}_{\mathbb{Q},\mathbb{A}}[L']$ sandwiching the unitary U. The intermediate unitary U can be packed through its $\mathbb{Q}$-rooted embedding

$$\mathcal{P}_{\mathbb{Q},\mathbb{A}}[L']\, U\, \mathcal{P}_{\mathbb{Q},\mathbb{A}}[L] = \mathcal{P}_{\mathbb{Q},\mathbb{A}}[L'\, \mathcal{B}_{\mathbb{Q}}(U)\, L]. \tag{34}$$

The same packing holds for $\mathbb{Q}'$-rooted packing processes $\mathcal{P}_{\mathbb{Q}',\mathbb{A}'}[K]$ and $\mathcal{P}_{\mathbb{Q}',\mathbb{A}'}[K']$ that sandwich the unitary U,

$$\mathcal{P}_{\mathbb{Q}',\mathbb{A}'}[K']\, U\, \mathcal{P}_{\mathbb{Q}',\mathbb{A}'}[K] = \mathcal{P}_{\mathbb{Q}',\mathbb{A}'}[K'\, \mathcal{B}_{\mathbb{Q}'}(U)\, K]. \tag{35}$$

Although one can always implement the $\mathbb{Q}$-rooted embedding of U followed by the $\mathbb{Q}'$-rooted embedding recursively,

$$U = \mathcal{P}_{\mathbb{Q}',\mathbb{A}'}\left[\mathcal{B}_{\mathbb{Q}'}\left(\mathcal{P}_{\mathbb{Q},\mathbb{A}}\left[\mathcal{B}_{\mathbb{Q}}\left(U\right)\right]\right)\right], \tag{36}$$

it is not guaranteed that one can still pack the $\mathbb{Q}$-rooted processes $\mathcal{P}_{\mathbb{Q},\mathbb{A}}[L_{1,2}]$ and $\mathbb{Q}'$-rooted processes $\mathcal{P}_{\mathbb{Q}',\mathbb{A}'}[K_{1,2}]$ simultaneously.

Fig. 10 (a) shows a $\{q_1, q_2\}$-rooted embedding of U followed by a recursive $\{q_1, q_3\}$-rooted embedding. In this example, the root qubits are $\mathbb{Q} = \{q_1, q_2\}$ and $\mathbb{Q}' = \{q_1, q_3\}$ associated with the auxiliary qubits $\mathbb{A} = \{e_1, e_2\}$ and $\mathbb{A}' = \{e_1', e_3\}$, respectively. One can simultaneously pack $\mathbb{Q}$-rooted processes and $\mathbb{Q}'$-rooted processes through the joint embedding of U, if U is $\mathbb{Q} \uplus \mathbb{Q}'$-rooted embeddable, as it is shown in Fig.

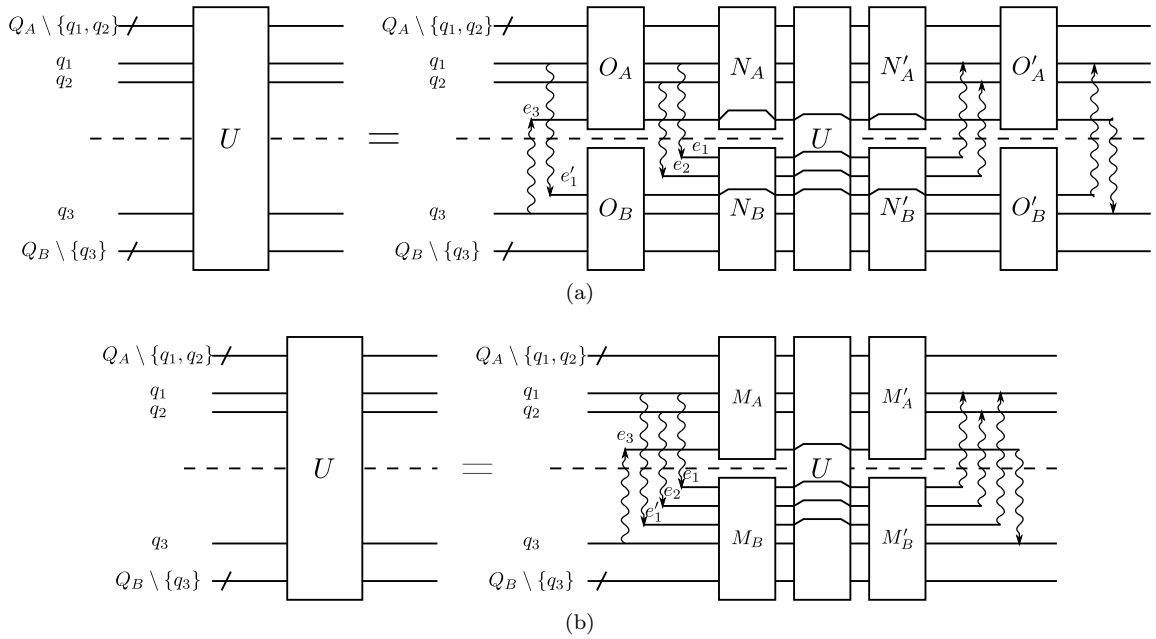


Figure 10: A unitary, which is embeddable on both $\mathbb{Q} = \{q_1, q_2\}$ and $\mathbb{Q}' = \{q_1, q_3\}$ (a) A $\mathbb{Q}$-rooted embedding followed by a $\mathbb{Q}'$-rooted embedding. (b) The $\mathbb{Q}$-rooted embedding and $\mathbb{Q}'$-rooted embedding are compatible, if U is $\mathbb{Q} \uplus \mathbb{Q}'$-rooted embeddable.

10 (b),

$$U = \mathcal{P}_{\mathbb{Q}\uplus\mathbb{Q}',\mathbb{A}\cup\mathbb{A}'}[M'_A \otimes M'_B\, U\, M_A \otimes M_B]. \tag{37}$$

Note that $\mathbb{Q} \uplus \mathbb{Q}'$ is the disjoint union of $\mathbb{Q}$ and $\mathbb{Q}'$, as the root qubits $\mathbb{Q}$ and $\mathbb{Q}'$ may share the same qubits. For example, the circuit on the left-hand side of Fig. 9 (b) is embeddable on both $\mathbb{Q} = \{q_1, q_2\}$ and $\mathbb{Q}' = \{q_1, q_3\}$. The $\mathbb{Q}$-rooted embedding and $\mathbb{Q}'$-rooted embedding of the circuit are compatible, since one can simultaneously implement these two embeddings in one joint embedding process.

As an embedding process can merge two non-sequential distributing processes, the compatibility of multiple embeddings allows us to merge multiple non-sequential distributing processes on multiple qubits. Fig. 11 demonstrates the utility of two compatible embeddings of U to merge the distributing process of $V_{1,2}$ and $W_{1,2}$. Suppose the unitary U in Fig. 11 is embeddable on both $\mathbb{Q} = \{q_1, q_2\}$ and $\mathbb{Q} = \{q_1, q_3\}$, which can be jointly embedded as shown in Fig. 10. Let $V_{1,2}$ and $W_{1,2}$ be distributable on $\mathbb{Q}'$ and $\mathbb{Q}$, respectively, while $W_{1,2}$ be $\mathbb{Q}'$-rooted embeddable with the embedding rule $\mathcal{B}_{\mathbb{Q}'}(W_{1,2}) = W_{1,2}$. As shown in Fig. 11 (a), one can firstly pack the $\mathbb{Q}'$-rooted distributing processes of $V_{1,2}$ through the $\mathbb{Q}'$-rooted embedding of $W_{1,2}$ and $\mathbb{Q} \uplus \mathbb{Q}'$-rooted embedding of U. Afterward, one can pack the $\mathbb{Q}$-rooted distributing processes of $W_{1,2}$ through the $\mathbb{Q} \uplus \mathbb{Q}'$-rooted embedding of U. In the end, the $\mathbb{Q} \uplus \mathbb{Q}'$-rooted embedding of U allows the simultaneous merging of the $\mathbb{Q}$-rooted processes and the $\mathbb{Q}'$-rooted processes.

In this example, the recursive embeddings of U rooted on $\mathbb{Q}$ and $\mathbb{Q}'$ are compatible for a simultaneous packing on $\mathbb{Q} \uplus \mathbb{Q}'$. For simultaneous packing, two embeddings must possess a recursive compatibility defined as follows.

Definition 12 (Compatible recursive embeddings) *Let $\mathcal{B}_{\mathbb{Q}}(U)$ and $\mathcal{B}_{\mathbb{Q}'}(U)$ be two embedding rules of a unitary U rooted on $\mathbb{Q}$ and $\mathbb{Q}'$, respectively. They are* compatible *for a recursive embedding on $\mathbb{Q} \uplus \mathbb{Q}'$, if U is jointly embeddable on $\mathbb{Q} \uplus \mathbb{Q}'$ with an embedding rule $\mathcal{B}_{\mathbb{Q}\uplus\mathbb{Q}'}$*

$$\mathcal{B}_{\mathbb{Q}\uplus\mathbb{Q}'}(U) = (M'_A \otimes M'_B)\; U\; (M_A \otimes M_B), \tag{38}$$

so that

$$U = \mathcal{P}_{\mathbb{Q}\uplus\mathbb{Q}',\mathbb{A}\cup\mathbb{A}'}\left[\mathcal{B}_{\mathbb{Q}\uplus\mathbb{Q}'}(U)\right]. \tag{39}$$

Since the two sets of root qubits $\mathbb{Q}$ and $\mathbb{Q}'$ can share some common elements, the compatible merging of the two embeddings $\mathcal{B}_{\mathbb{Q}}$ and $\mathcal{B}_{\mathbb{Q}'}$ takes the disjoint union set $\mathbb{Q} \uplus \mathbb{Q}'$ as its root qubits. For the example in Fig. 10 (b), the root qubits after the compatible merging are $\mathbb{Q} \uplus \mathbb{Q}' = \{q_1, q_2, q_1, q_3\}$ associated with the auxiliary qubits $\mathbb{A} \cup \mathbb{A}' = \{e_1, e_2, e'_1, e_3\}$.

As a result of Corollary 9, the compatibility of two recursive embeddings can be determined with the following sufficient condition.

Lemma 13 (Condition for recursive compatibility) *Let $\mathcal{B}_{\mathbb{Q}'}$ be a $\mathbb{Q}'$-rooted embedding rule of U,*

$$\mathcal{B}_{\mathbb{Q}'}(U) = (K'_A \otimes K'_B)\, U\, (K_A \otimes K_B). \tag{40}$$

It is recursively compatible with another $\mathbb{Q}$-rooted embedding of U, if

$$[K'_A \otimes K'_B, C_{\mathbb{Q},X_{\mathbb{A}}}] = [K_A \otimes K_B, C_{\mathbb{Q},X_{\mathbb{A}}}] = 0. \tag{41}$$

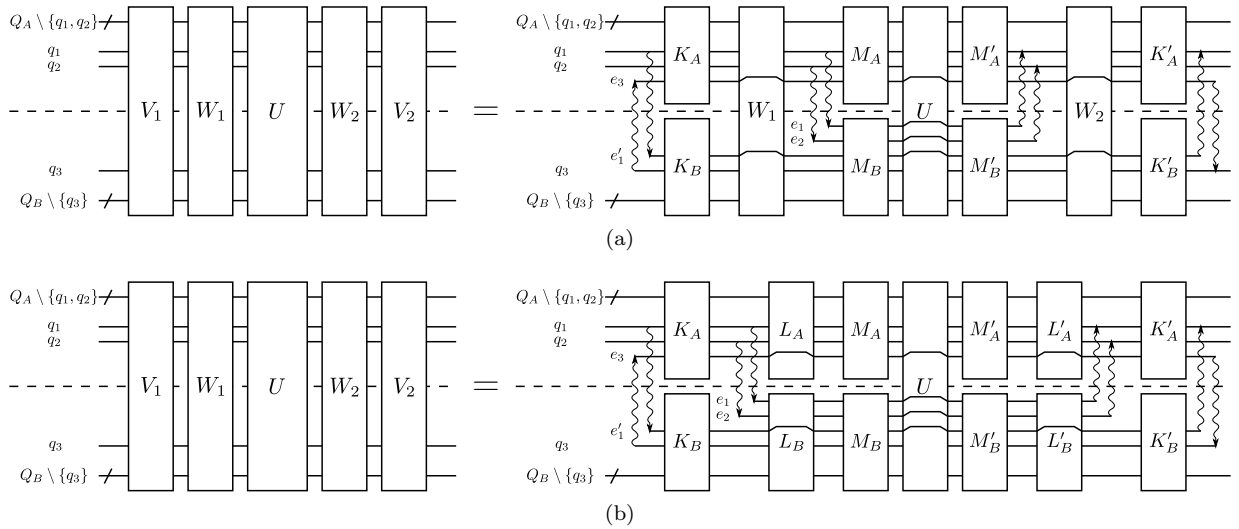


Figure 11: Simultaneous packing through compatible recursive embeddings (a) The packing of the $\mathbb{Q}'$-rooted processes including the embedding of $W_{1,2}$, embedding of U and distributing of $V_{1,2}$. (b) The simultaneous packing of $\mathbb{Q}$-rooted processes and $\mathbb{Q}'$-rooted processes that merges the $\mathbb{Q}'$-rooted embedding of $W_{1,2}$, $\mathbb{Q}'$-rooted distributing of $V_{1,2}$, and $\mathbb{Q}$-rooted distributing of $W_{1,2}$ through the $\mathbb{Q} \uplus \mathbb{Q}'$-rooted embedding of U.

The joint embedding rule is $\mathcal{B}_{\mathbb{Q}\uplus\mathbb{Q}'} = \mathcal{B}_{\mathbb{Q}'} \circ \mathcal{B}_{\mathbb{Q}}$.
Proof: See Appendix D.3 ■

As a result of Lemma 13, the embedding rule for distributable unitaries are all recursively compatible.

Corollary 14 (Recursive embedding of distributing) *The q-rooted embedding of a q-rooted distributable unitary is recursively compatible with any other embeddings.*
Proof: As a result of Lemma 13, the embedding rules in Theorem 10 are compatible for any recursive embeddings, since the additional kernel is $\mathbb{1}$ *or* X_e*, which commutes with any control-X gate* $C_{q',X_{e'}}$*,* $[\mathbb{1}, C_{q',X_{e'}}] = 0$ *and* $[X_e, C_{q',X_{e'}}] = 0$. ■

A more complicated situation of simultaneous embeddings is the nested embedding shown in Fig. 12. Suppose the unitaries $U = BA$ and $U' = CB$ are embeddable on $\mathbb{Q} = \{q_1, q_2\}$ and $\mathbb{Q}' = \{q_1, q_3\}$ with the embedding rules $\mathcal{B}_{\mathbb{Q}}$ and $\mathcal{B}_{\mathbb{Q}'}$, respectively. If we have a unitary decomposed as CBA, it is not guaranteed that one can implement the embeddings of $\mathcal{B}_{\mathbb{Q}}(BA)$ and $\mathcal{B}_{\mathbb{Q}'}(CB)$ simultaneously. If the equality in Fig. 12 holds, the nested embeddings $\mathcal{B}_{\mathbb{Q}}(BA)$ and $\mathcal{B}_{\mathbb{Q}'}(CB)$ are compatible for simultaneous implementation.

Definition 15 (Compatible nested embedding) *Let* U *and* U' *be embeddable on* $\mathbb{Q}$ *and* $\mathbb{Q}'$ *with the embedding rules* $\mathcal{B}_{\mathbb{Q}}(U)$ *and* $\mathcal{B}_{\mathbb{Q}'}(U')$*, respectively. For the decomposition of* U *and* U'*,*

$$U = BA, \;\; U' = CB. \tag{42}$$

The embedding rules $\mathcal{B}_{\mathbb{Q}}(U)$ *and* $\mathcal{B}_{\mathbb{Q}'}(U')$ *are* compatible for the nested embedding *of* CBA*, if*

$$\begin{aligned} CBA =& E_{\mathbb{Q}',\mathbb{A}'} \circ L'C \circ E_{\mathbb{Q},\mathbb{A}} \circ K'BL \\ & \circ S_{\mathbb{Q}',\mathbb{A}'} \circ AK \circ S_{\mathbb{Q},\mathbb{A}}. \end{aligned} \tag{43}$$

where $K = K_A \otimes K_B$*,* $K' = K'_A \otimes K'_B$*,* $L = L_A \otimes L_B$*, and* $L' = L'_A \otimes L'_B$ *are all local unitaries (see Fig. 12).*

3.5 Packing of distributable packets

A quantum circuit of η qubits and depth τ can be represented by a $\eta \times \tau$ grid with the nodes $\{(q,t) : q \in \{q_1, ..., q_\eta\}, t \in \{1, ..., \tau\}\}$. At each depth t, we have a unitary U_t implemented by a set of gates $g_{t,i}^{(\mathbb{Q}_i)}$ acting on disjoint sets of qubits $\mathbb{Q}_i$, which satisfy $\cup_i \mathbb{Q}_i = \{q_1, ..., q_\eta\}$ and $\mathbb{Q}_i \cap \mathbb{Q}_j = \emptyset$. As a result of Theorem 5, the distributability of U_t on a root qubit q does not depend on the bipartition $A|B$. It means that one can easily identify all the distributable nodes of a circuit without paying any attention to the partitioning. The distributable nodes indicate the potential root points of distributing processes for global gates. In the packing procedure of the distributing processes, one can neglect the circuit nodes of single-qubit distributable gates and only consider the multi-qubit gates. According to Theorem 3, two subsequential distributable nodes (q,t) and $(q,t+1)$ can be straightforwardly packed in one distributing process.

Suppose there are two non-sequential distributable nodes (q,t_1) and (q,t_2) separated by a unitary $W_{t_2;t_1}^{(q)}$, which is non-distributable but embeddable. One can still pack the two non-sequential distributing processes rooted on (q,t_1) and (q,t_2) into one packing process through the embedding of $W_{t_2;t_1}^{(q)}$ according to Theorem 3. The embeddability of $W_{t_2;t_1}^{(q)}$ therefore

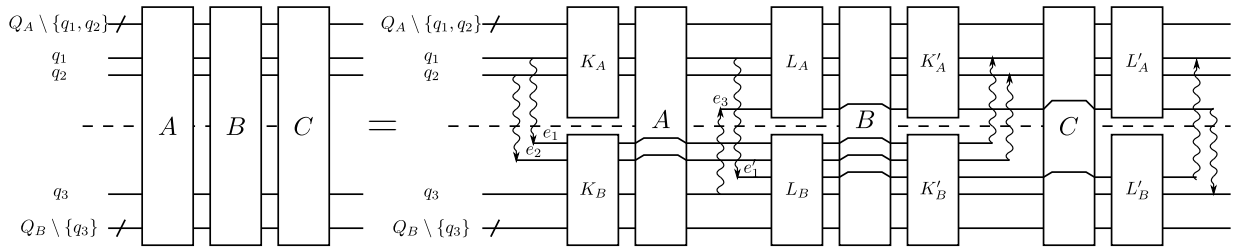


Figure 12: Compatible nested embeddings. The unitaries $U = BA$ and $U' = CB$ are embeddable on $\mathbb{Q} = \{q_1, q_2\}$ and $\mathbb{Q}' = \{q_1, q_3\}$ with the embedding rules $\mathcal{B}_{\mathbb{Q}}$ and $\mathcal{B}_{\mathbb{Q}'}$, respectively. If the equality holds, $\mathcal{B}_{\mathbb{Q}}$ and $\mathcal{B}_{\mathbb{Q}'}$ are compatible.

allows us to pack the two non-sequential distributing processes through the embedding of $W^{(q)}_{t_2;t_1}$. Such a packing of non-sequential distributing processes reduces the entanglement cost in the distributed quantum computation. To explore all the possible packing of distributing processes in a quantum circuit, we need to identify its potential embedding processes and distributing processes. The embedding processes will then merge the root points of distributing processes into *distributable packets*, which are introduced as follows.

Definition 16 (distributable packet)
Let $\mathcal{V}_{q,T}$ be a set of distributable nodes (q,t) on the qubit q at the depths $t \in T$ of a quantum circuit,

$$\mathcal{V}_{q,T} := \{(q,t) : t \in T, U_t \text{ is } q\text{-rooted distributable, and } (q,t) \text{ is associated with a multi-qubit gate}\}. \tag{44}$$

Let $\mathbb{B}$ be a set of embedding rules. The set $\mathcal{V}_{q,T}$ is $\mathbb{B}$-packable with respect to a bipartition $A|B$, if the unitary between each pair of depths $\{t_1, t_2\} \subseteq T$ fulfills

$$\prod_{t:t_1<t<t_2} U_t \text{ is } q\text{-rooted embeddable} \tag{45}$$

with respect to the bipartition $A|B$ according to the embedding rules in $\mathbb{B}$. The set $\mathcal{V}_{q,T}$ is called a distributable $\mathbb{B}$-packet. *A single distributable node $\{(q,t)\}$ is a* trivial distributable $\mathbb{B}$-packet. *The set of distributable $\mathbb{B}$-packets of a circuit is denoted as $\mathbb{V}_{\mathbb{B}}$.*

If there is no ambiguity in a context, we will employ the terminology "distributable packet" without the specification of the embedding rule $\mathbb{B}$ for conciseness. In this definition, we only consider the distributable nodes, which are associated with global gates in Eq. (44), since local gates do not need to be distributed. It is worth noting that the nodes in $\mathcal{V}_{q,T}$ are not necessarily contiguous.

If two distributable packets have non-empty intersection, they can be merged into a larger set through the following corollary.

Lemma 17 (Merging of distributable packets)
Two distributable packets $\mathcal{V}_{q,T_1}$ and $\mathcal{V}_{q,T_2}$ can be merged into one packet $\mathcal{V}_{q,T_1\cup T_2}$, if they have non-empty intersection $T_1 \cap T_2 \neq \emptyset$,

$$\mathcal{V}_{q,T_1} \cup_D \mathcal{V}_{q,T_2} := \mathcal{V}_{q,T_1\cup T_2}. \tag{46}$$

The symbol $\cup_D$ represents the merging of distributable packets.
Proof: See Appendix D.4. ∎

After the merging of these processes, one obtains a *packing process on a distributable packet*, of which the kernels are given as follows.

Theorem 18 (Packing on distributable packets)
Given a distributable packet $\mathcal{V}_{q,T}$, the q-rooted distributing processes of U_t with $t \in T$ can be packed into one packing process through embedding as follows,

$$\prod_{\min T\le t\le \max T} U_t = \mathcal{P}_{q,e}[\prod_{\min T\le t\le \max T} K_t], \tag{47}$$

where the kernel K_t of the depth t is

$$K_t = \begin{cases} \mathcal{D}_{q,t} := \mathcal{D}_q(U_t), & t \in T \\ \mathcal{B}_{q,t} := \mathcal{B}_q(U_t), & t \notin T \end{cases}. \tag{48}$$

Proof: The unitary $\prod_{t\in[\min T,\max T]} U_t$ can be implemented with distributing processes for $t \in T$ and embedding processes for $t \notin T$

$$\prod_{t\in[\min T,\max T]} U_t = \prod_{t\in[\min T,\max T]} \mathcal{P}_{q,e_t}[K_t], \tag{49}$$

where the kernel K_t of the depth t is the distributing rule $\mathcal{D}_q(U_t)$ for $t \in T$ and the embedding rule $\mathcal{B}_q(U_t)$

for $t \notin T$. As a result of Theorem 3, the product of unitary-equivalent packing processes can be packed into the single packing process given in Eq. (47). ■

The set of distributable packets of a quantum circuit after all possible mergings contains only disjoint packets. The DQC of a quantum circuit can be implemented through the packing processes rooted on these distributable packets. The number of selected root packets is equal to the entanglement cost of the corresponding DQC implementation.

3.6 Conflicts among packing processes

Although distributing processes and embedding processes can be packed in a distributable packet, it is not guaranteed that the packing processes rooted on two distributable packets can be simultaneously implemented. The packing process on a distributable packet contains the distributing processes rooted on its elements, and the embedding processes between them. The implementation of these distributing processes and embedding processes may introduce additional local kernels that change the distributablity and embeddability of another packet. This may cause incompatibility among distributable packets.

To reveal the incompatibility among distributing processes and embedding processes on distributable packets, we can unpack a packing process on a packet $\mathcal{V}_{q,T}$ in Eq. (47) into a sequence of packing processes represented by distributing kernels and embedding kernels

$$\begin{gathered}\mathcal{V}_{q,T} \text{ with } T=\{t_1,...,t_k\}\\ \downarrow \text{unpack}\\ (\mathcal{D}_{q;t_1},\mathcal{B}_{q;t_1,t_2},\mathcal{D}_{q;t_2},...,\mathcal{D}_{q;t_{k-1}},\mathcal{B}_{q;t_{k-1},t_k},\mathcal{D}_{q;t_k}),\end{gathered} \tag{50}$$

where

$$\mathcal{B}_{q;t_1,t_2} := \prod_{t:t_1<t<t_2} \mathcal{B}_{q;t}. \tag{51}$$

If there exists a trivial distributable packet $\mathcal{V}_{q;t}$ with $t_1 < t < t_2$, then one can replace the embedding rules $\mathcal{B}_{q;t}$ in Eq. (51) by the distributing rule $\mathcal{D}_{q;t}$

$$\mathcal{B}_{q;t_1,t_2} \to \mathcal{B}_{q;t,t_2}\,\mathcal{D}_{q;t}\,\mathcal{B}_{q;t_1,t}. \tag{52}$$

It will add the node (q,t) into the packet $\mathcal{V}_{q,T}$ in Eq. (50) and form a larger packet $\mathcal{V}_{q,T\cup\{t\}}$. We call such embedding *decomposable*, which is defined as follows.

Definition 19 (Indecomposable embedding)
An embedding process $\mathcal{B}_{q;t_1,t_2}$ is decomposable*, if there exists only a distributable node $\mathcal{V}_{q,t}$ of a multi-qubit gate with $t_1 < t < t_2$, such that*

$$\mathcal{P}_{q,e}[\mathcal{B}_{q;t_1,t_2}] = \mathcal{P}_{q,e}[\mathcal{B}_{q;t,t_2}\,\mathcal{D}_{q;t}\,\mathcal{B}_{q;t_1,t}], \tag{53}$$

Otherwise, $\mathcal{B}_{q;t_1,t_2}$ is indecomposable*. A distributable packet $\mathcal{V}_{q;\{t_1,t_2\}}$ is indecomposable, if $\mathcal{B}_{q;t_1,t_2}$ is indecomposable.*

By definition, the embedding $\mathcal{B}_{q;t}$ on a trivial distributable node $\mathcal{V}_{q,t}$ is also an indecomposable embedding.

Given a circuit Q, the set of all indecomposable embeddings fully describes the embedding rules applied to the circuit Q,

$$\mathbb{B}(Q) = \{\mathcal{B}_{q;T} : \text{indecomposable}\}. \tag{54}$$

On the other hand, the set of all distributing kernels on trivial distributable packets fully describes the distributing rules for Q

$$\mathbb{D}(Q) = \{\mathcal{D}_{q;t} : \mathcal{V}_{q,t} \text{ is a trivial distributable packet}\}. \tag{55}$$

The incompatibility among distributing rules and embedding rules should be identified at the level of indecomposable embedding processes and trivial distributing processes, which forms a set of kernels $\mathbb{K}$ of the potential packing processes

$$\mathbb{K}(Q) := \mathbb{D}(Q) \cup \mathbb{B}(Q). \tag{56}$$

We call such a set the *indecomposable packing kernels* of Q. The incompatibility among indecomposable packing kernels can be identified with directed conflict edges

Definition 20 (Kernel-conflict edges)
Let $K_{q_1;T_1}, K_{q_2;T_2} \in \mathbb{K}(Q)$ be two indecomposable packing kernels of a circuit Q. The incompatibility between $K_{q_1;T_1}$ and $K_{q_2;T_2}$ is represented by a directed edge

$$K_{q_1;T_1} \to K_{q_2;T_2} \Leftrightarrow c(K_1,K_2) = 1, \tag{57}$$

if the implementation of K_1 prevents the implementation of K_2. If the two kernels are mutually incompatible, i.e. $c(K_1,K_2) = c(K_2,K_1) = 1$, it forms an undirected edge

$$K_{q_1;T_1} \leftrightarrow K_{q_2;T_2} := \{K_{q_1;T_1} \to K_{q_2;T_2}, K_{q_2;T_2} \to K_{q_1;T_1}\}. \tag{58}$$

The matrix $\{c(K_i,K_j)\}_{i,j}$ is the adjacency matrix of conflict edges.

A conflict caused by intrinsic incompatibility of packing processes due to the intrinsic quantum circuit structure, such as the incompatibility of embeddings, is called an *intrinsic conflict*. In practice, two simultaneous packing processes need to compete for the necessary external resources. As shown in Fig. 2, the external resources consumed in a packing process are one-ebit entangled pair, an auxiliary memory qubit e' on the root QPU and an auxiliary memory qubit e on the remote QPU, which are required in DQC implementation. The limitation on available external resources leads to conflicts among packing processes. Such a conflict caused by limited external resources is referred to as an *extrinsic conflict*.

Note that, in this paper, the auxiliary memory qubits in the root QPU are not counted as limited

external resources, since they are only needed in the starting process (see Fig. 2 (a)), and can be reset and reused immediately after the starting processes. Besides, we ignore the limitation on the coherence time of auxiliary memory qubits, as we assume that their coherence time is not worse than the local working qubits and long enough for implementing packing processes.

In a circuit, an indecomposable packing kernel can be represented by a vertex placed at its root circuit nodes with its packing rule written on it. For example, in Fig. 13, the two trivial distributable packets $\mathcal{D}_{q_1;t}$ and $\mathcal{D}_{q_2;t}$ are represented as red vertices. According to Theorem 10, every distributable unitary is also embeddable. There must be an underlying indecomposable embedding process associated with each trivial distributable packet rooted on the same circuit nodes. On the same root circuit nodes, the root vertices of distributing kernels $\mathcal{D}_{q_1;t}$ and $\mathcal{D}_{q_2;t}$ are placed over their underlying indecomposable embedding vertex $\mathcal{B}_{q_1;t}$ and $\mathcal{B}_{q_2;t}$, respectively. A further example of indecomposable packing kernel $\mathcal{B}_{q_1;t_1,t_3}$ is shown in Fig. 14 as a pincer-shaped vertex on its root circuit nodes, where its pincer-shaped geometry indicates its ability to merge two distributable packets before and after.

At the level of indecomposable kernels, there are three types of incompatibility, namely the incompatibility between two distributing processes, between a distributing process and an embedding process, and between two embedding processes, which are demonstrated in Fig. 13, 14 and 15, respectively.

DD-type conflicts (Fig. 13). The first type of conflict is the incompatibility between two distributing processes, which we call a *DD-type* conflict. As shown in Fig. 13(a), on the right-hand side, a control-Z gate can be distributed in a distributing process with the kernel $\mathcal{D}_{q_1;t}$ or $\mathcal{D}_{q_2;t}$ rooted on the node (q_1, t) or (q_2, t), respectively. If one of the distributing processes is implemented, the other process is not needed anymore. The DD-type conflict of this example is an intrinsic property of a nonlocal control-Z gate. It is therefore an instrinsic conflict. On the left-hand side of Fig. 13(a), the distributability of the control-Z gate is represented by two vertices (trivial distributable packets), where the distributing rules $\mathcal{D}_{q_1;t}$ and $\mathcal{D}_{q_2;t}$ of the control-Z gate are rooted. The incompatibility of the distributability is then represented by a red bidirected edge $(\mathcal{D}_{q_1;t} \leftrightarrow \mathcal{D}_{q_2;t})$ between these two red vertices.

For the same CZ gate, besides the distributability, it can be also embedded with the embedding rule $\mathcal{B}_{q_1;t}$ shown on the right-hand side of Fig. 13 (a). The embedding rule $\mathcal{B}_{q_1;t}$ and the distributing rule $\mathcal{D}_{q_1;t}$ compete for auxiliary memory qubits on system B, since each packing auxiliary qubit can only be assigned to a process at one time. If there is only one auxiliary memory qubit e_B available, then only either $\mathcal{B}_{q_1;t}$ or $\mathcal{D}_{q_1;t}$ can be implemented, which is an extrinsic conflict. Since a distributing process has a higher priority over an embedding process, the conflict edge $(\mathcal{B}_{q_1;t} \leftarrow \mathcal{D}_{q_1;t})$ is directed from the distributing rule represented by a green arrow in the right-hand side figure.

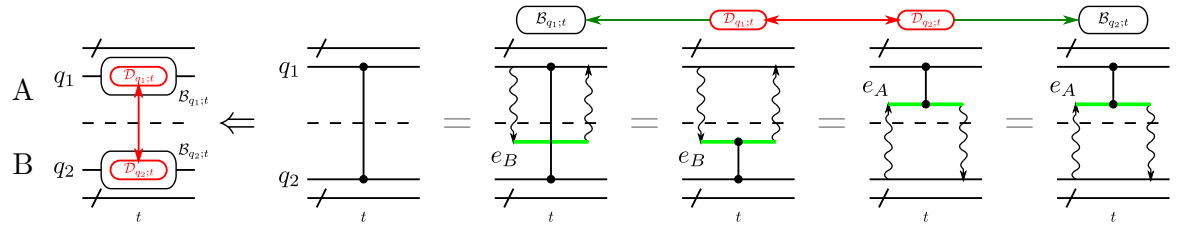
As a whole, the incompatibility among the packing processes is represented by a graph of indecomposable packing root vertices. A red bidirected edge represents the intrinsic conflict between two trivial distributing processes, while a directed green edge represents the incompatibility between a trivial distributing process and its underlying embedding. Note that the directed green edge is omitted on the left-hand side graph by default.

Other possible DD-type conflicts occur when two trivial distributable packets are placed at the same depth of a local QPU as shown in Fig. 13 (b). The conflict is caused by the competition among packing processes for external auxiliary memory qubits, which is extrinsic. The conflict between the embeddings $\mathcal{B}_{q_1;t}$ and $\mathcal{B}_{q_2;t}$ is automatically resolved if the conflicts associated with the trivial distributable packets are resolved. We therefore employ a dashed edge to represent it.

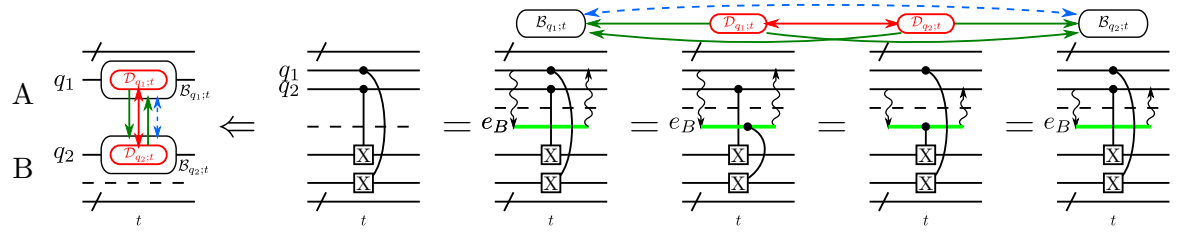
DB-type conflicts (Fig. 14). The second type of conflict is the incompatibility between a trivial distributing process and an indecomposable embedding process, which we call a *DB-type* conflict. As shown on the right side of Fig. 14, a circuit consisting of an X gate and a CNOT gate on the depth t_2 can be distributed in a distributing process with the rule $\mathcal{D}_{q_2;t_2}$ rooted on (q_2, t_2), which is also embeddable with the rule $\mathcal{B}_{q_2;t_2}$ rooted on (q_2, t_2). Another indecomposable embedding rule $\mathcal{B}_{q_1;t_1,t_3}$ can be applied on (q_1, t_2), which merges the packing processes rooted on (q_1, t_1) and (q_1, t_3).

All of the three packing processes compete for local auxiliary memory qubits, which leads to extrinsic conflicts. Since the distributing $\mathcal{D}_{q_2;t_2}$ has a higher priority than the embeddings, we employ green directed edges $\mathcal{D}_{q_2;t_2} \rightarrow \mathcal{B}_{q_2;t_2}$ and $\mathcal{D}_{q_2;t_2} \rightarrow \mathcal{B}_{q_1;t_1,t_3}$ from the distributing root vertex $\mathcal{D}_{q_2;t_2}$ to the embedding root vertices to represent the DB-type conflicts.

The conflict between the indecomposable embeddings is represented by a blue bidirected edge $\mathcal{B}_{q_2;t_2} \leftrightarrow \mathcal{B}_{q_1;t_1,t_3}$. Here, a dashed line is employed, since this type of extrinsic conflicts will be automatically resolved, once the conflicts associated with trivial distributing process are resolved. For a detailed explanation please refer to the BB-type conflicts in the next paragraph.



(a) DD-type conflict (intrinsic)



(b) DD-type conflict (extrinsic)

Figure 13: Conflict between two distributing processes. (a) Intrinsic conflict between two trivial distributable packets. (b) Extrinsic conflict between two trivial distributable packets.

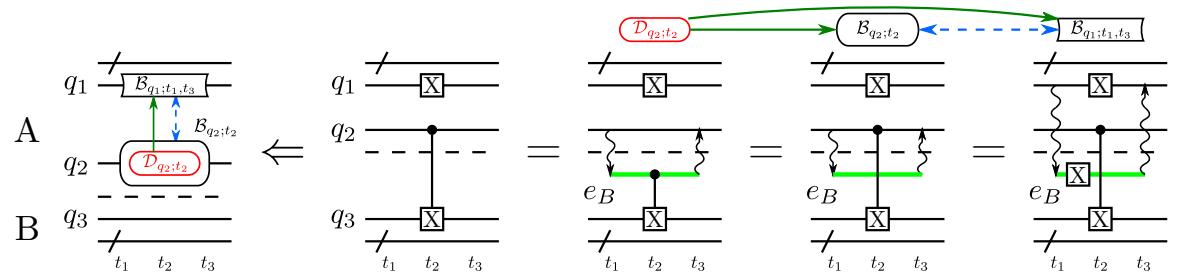
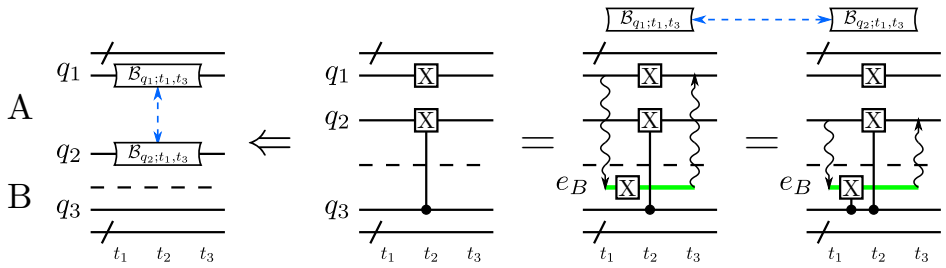


Figure 14: Conflict between a distributing process on a trivial distributable packet and an indecomposable embedding, which compete for auxilliary memory qubits on the other local system.

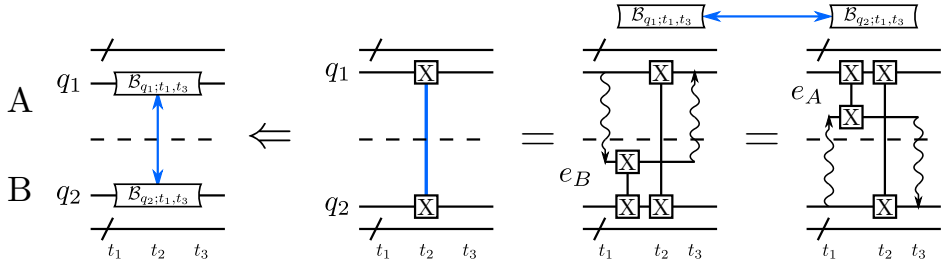
BB-type conflicts (Fig. 15). The third type of conflict is the incompatibility between two indecomposable embeddings, which we call *BB-type conflicts*. Fig 15 (a) shows a circuit example of extrinsic BB-type conflicts, in which the embedding $\mathcal{B}_{q_1;t_1,t_3}$ of the X gate on (q_1, t_2) competes for auxiliary memory qubits with the embedding $\mathcal{B}_{q_2;t_1,t_3}$ of the CNOT gate on (q_2, t_2). A blue bidirected edge $\mathcal{B}_{q_1;t_1,t_3} \leftrightarrow \mathcal{B}_{q_2;t_1,t_3}$ on the pincer-shaped vertices represents the extrinsic conflict. In DQC implementation, the utility of external resources for a packing process is meaningless unless a distributing process is involved. An indecomposable embedding must be packed inside a distributable packet. An extrinsic conflict between two indecomposable embeddings therefore implies a conflict between two distributable packets competing for the same extrinsic resource. It implies the existence of distributing processes that compete for the same resource and leads to DB-type or DD-type conflicts, which have a higher priority. As a result, such an extrinsic BB-type conflict is redundant, which is then represented by dashed lines.

Besides extrinsic BB-type conflicts, there are some conflicts between indecomposable embeddings, which are intrinsic and not resolvable by adding external resources. For the circuit example shown in Fig 15 (b), the nonlocal gate is either embeddable on $(q_1; t_2)$ or $(q_2; t_2)$ with the embedding rules $\mathcal{B}_{q_1;t_1,t_3}$ or $\mathcal{B}_{q_2;t_1,t_3}$, respective. However, these two embedding rules are not compatible for joint embedding due to fundamental limits (see Theorem 36 in Section 4.2). One has to resolve this conflict by abandoning one of the embeddings. Such an intrinsic BB-type conflict is represented as a blue bidirected edge $\mathcal{B}_{q_1;t_1,t_3} \leftrightarrow \mathcal{B}_{q_2;t_1,t_3}$ connecting the embedding root vertices. Since an intrinsic conflict is never redundant, it is always represented with a solid line.

Conflict structure and its solution. The distribution of a quantum circuit can possess all these three types at the same time. Fig. 16 shows the conflict structure of a circuit example. The kernels of trivial distributing processes and indecomposable embedding processes of the circuit are listed as follows,



(a) Extrinsic BB-type conflict



(b) Intrinsic BB-type conflict

Figure 15: Conflict between two embedding processes. (a) Extrinsic conflict between two indecomposable embeddings competing for memory qubits. (b) Intrinsic conflict between two indecomposable embeddings due to their intrinsice incompatibility.

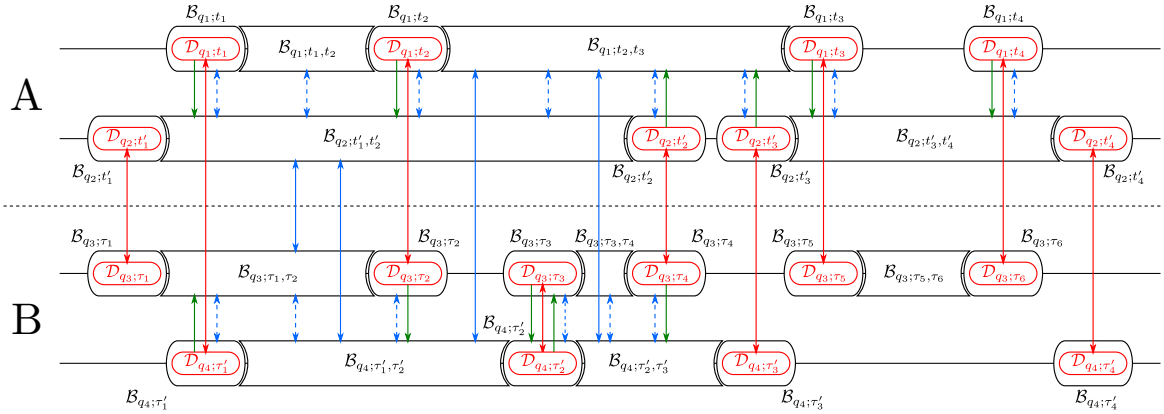


Figure 16: An example of incompatibility among distributing and embedding processes of a circuit represented by directed conflict edges. The red, green, and blue arrows represent the DD-type, DB-type, and BB-type conflicts, respectively. A dashed line indicates the redundancy of the conflict due to the existence of other conflicts associated with trivial distributable packets.

$$\mathbb{K} = \left\{ \begin{array}{cccccc} \mathcal{D}_{q_1;t_1}, & & \mathcal{D}_{q_1;t_2}, & & \mathcal{D}_{q_1;t_3}, & \\ \mathcal{B}_{q_1;t_1}, & \mathcal{B}_{q_1;t_1,t_2}, & \mathcal{B}_{q_1;t_2}, & \mathcal{B}_{q_1;t_2,t_3}, & \mathcal{B}_{q_1;t_3}, & \cdots \\ \mathcal{D}_{q_2;t'_1}, & & \mathcal{D}_{q_2;t'_2}, & \cdots & & \\ \mathcal{B}_{q_2;t'_1}, & \mathcal{B}_{q_2;t'_1,t'_2}, & \mathcal{B}_{q_2;t'_2}, & \cdots & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{array} \right\}. \tag{59}$$

With limited resources, one has to abandon conflicting processes to resolve the conflicts, the actions include the removal of distributing processes and the removal of embedding processes from a distributable packet. Let $\mathcal{V}_{q,T'}$ be a sub-packet of a distributable packet $\mathcal{V}_{q,T}$. The abandonment of the distributing processes associated with a sub-packet $\mathcal{V}_{q,T'}$ from a packet $\mathcal{V}_{q,T}$ does not affect the embeddability on (q, T'). It therefore leads to a new packet $\mathcal{V}_{q,T\setminus T'}$. The action of the removal of $\mathcal{V}_{q,T'}$ from $\mathcal{V}_{q,T}$ is defined as follows.

Definition 21 (Removal of distributing processes) *After the removal of distributing processes associated*

with $\mathcal{V}_{q,T'}$ from $\mathcal{V}_{q,T}$, where $T' \subset T$, one obtains

$$\mathcal{V}_{q,T} \setminus_D \mathcal{V}_{q,T'} := \mathcal{V}_{q,T\setminus T'}. \tag{60}$$

On the contrary, the removal of embedding processes from a distributable packet splits the packet.

Definition 22 (Removal of embedding processes) *Let $\mathcal{B}_{q;t_1,t_2}$ be an indecomposable embedding in a distributable packet $\mathcal{V}_{q;T}$ with $\{t_1,t_2\} \subseteq T$. After the removal of the embedding $\mathcal{B}_{q;t_1,t_2}$, the packet $\mathcal{V}_{q;T}$ is split into two packets*

$$\mathcal{V}_{q,T} \setminus_B \mathcal{B}_{q;t_1,t_2} := \{\mathcal{V}_{q,T_1}, \mathcal{V}_{q,T_2}\} \tag{61}$$

where $T_{1,2}$ are the two split regions of depths,

$$T_1 := \{t \in T : t \le t_1\}, \quad T_2 \quad := \{t \in T : t \ge t_2\}. \tag{62}$$

3.7 Packing graphs and conflict graphs

The conflict between packing processes can be solved through the removal of distributing processes or embedding processes. Since distributing processes have higher priority, we will first solve the DD-type conflicts among distributing processes prior to the DB-type and BB-type conflicts, which involve embedding processes. In a quantum circuit, the conflicts between two distributing processes are associated with global gates. For general multi-qubit gates, one needs hyperedges to describe the conflict. Since all multi-qubit gates can be decomposed into two-qubit gates, in this paper, we consider the circuits solely consisting of one-qubit and two-qubit gates. For a two-qubit gate which is distributable on both of its gate nodes (q_1,t) and (q_2,t), each gate node belongs to a distributable packet $\mathcal{V}_{q_1,T_1}$ and $\mathcal{V}_{q_2,T_2}$, respectively. One has the freedom to choose a root packet from $\mathcal{V}_{q_{1,2},T_{1,2}}$ to anchor a distributing process. Once a root packet is selected, the distributable node on the other packet has to be removed according to Definition 21. One can assign a packing edge to each two-qubit gate and form a packing graph.

Definition 23 (Packing edge) *Given a set of distributable $\mathbb{B}$-packets $\mathbb{V}_{\mathbb{B}}$, two distributable $\mathbb{B}$-packets $\mathcal{V}_{q_1,T_1}$ and $\mathcal{V}_{q_2,T_2}$ are* connected *by a two-qubit gate g_t at the depth t, if the nodes $\{(q_1,t),(q_2,t)\}$ of g_t belong to $\mathcal{V}_{q_1,T_1}$ and $\mathcal{V}_{q_2,T_2}$, respectively,*

$$(q_1,t) \in \mathcal{V}_{q_1,T_1} \quad \textit{and} \quad (q_2,t) \in \mathcal{V}_{q_2,T_2}. \tag{63}$$

The pair of connected distributable packets forms a packing edge *associated with the gate g_t*

$$\mathcal{E}_{g_t} := (\mathcal{V}_{q_1,T_1} \longleftrightarrow \mathcal{V}_{q_2,T_2}) := \{\mathcal{V}_{q_1,T_1}, \mathcal{V}_{q_2,T_2}\}. \tag{64}$$

The gate g_t is a packing-edge *gate. The packing edge set is denoted by $\mathbb{E}_{\mathbb{B}} = \{\mathcal{E}_{g_t}\}_{g_t}$. If $\mathcal{V}_{q_1,T_1}$ and $\mathcal{V}_{q_2,T_2}$ belong to the same local system, the edge $\mathcal{E}_{g_t}$ is* local*, otherwise* global*.*

Definition 24 (Packing graph) *A $\mathbb{B}$-packing graph of a circuit is formed by the set of distributable $\mathbb{B}$-packets and the set of packing edges,*

$$G_{\mathbb{B}} := (\mathbb{V}_{\mathbb{B}}, \mathbb{E}_{\mathbb{B}}). \tag{65}$$

To distribute all the global gates, one needs to select a set of distributable packets, which cover all packing edges. Note that the packing edges are defined at the level of distributable $\mathbb{B}$-packets, while the kernel-conflict edges in Definition 20 are defined at the level of indecomposable kernels of distributing and embedding processes.

Besides the DD-type conflict, one also needs to take the conflicts associated with embeddings into account. Each conflict edge at the level of indecomposable kernels incident to an embedding kernel also defines a conflict edge at the level of distributable packets.

$$\mathcal{V}_{q_1,T_1} \xrightarrow[C]{} \mathcal{V}_{q_2,T_2} :\Leftrightarrow \begin{cases} \exists \mathcal{D}_{q_1;t_1} \to \mathcal{B}_{q_2;\tau_1,\tau_2}, & \text{s.t. } t_1 \in T_1, \text{ and } \min T_2 \le \tau_{1,2} \le \max T_2 \\ \text{or} & \\ \exists \mathcal{B}_{q_1;t_1,t_2} \to \mathcal{B}_{q_2;\tau_1,\tau_2}, & \text{s.t. } \min T_1 \le t_{1,2} \le \max T_1, \text{ and } \min T_2 \le \tau_{1,2} \le \max T_2 \end{cases} \tag{66}$$

A quantum circuit has two sets of conflict edges at two different levels, one is the set of kernel-conflict edges at the level of indecomposable packing kernels, the other is the set of packet-conflict edges at the level of distributable packets.

Definition 25 (Conflict edge set) *The set of kernel-conflict edges $\mathbb{C}(\mathbb{K})$ is defined at the level of indecomposable packing kernels $\mathbb{K}$,*

$$\mathbb{C}(\mathbb{K}) := \{K_1 \to K_2 : K_{1,2} \in \mathbb{K}, K_2 \textit{ is an embedding}\}, \tag{67}$$

while the set of packet-conflict edges $\mathbb{C}(\mathbb{V})$ is defined at the level of distributable packets $\mathbb{V}$,

$$\mathbb{C}(\mathbb{V}) := \{\mathcal{V}_1 \to \mathcal{V}_2 : \textit{defined in Eq. (66), and } \mathcal{V}_{1,2} \in \mathbb{V}\}. \tag{68}$$

The extrinsic conflict edges caused by the competition for external resource, e.g. auxiliary qubits, can be solved by supplying sufficient external resources. The intrinsic conflict edges caused by the intrinsic incompatibility of embeddings is not solvable with external resources. The extrinsic and intrinsic edge sets are denoted by subscripts $\mathbb{C}_{in}$ and $\mathbb{C}_{ex}$, respectively. We can employ the intrinsic and extrinsic conflict graphs to represent the conflicts of packing processes.

Definition 26 (Conflict graph) *Packet-conflict graphs and kernel-conflict graphs are defined at the level of distributable $\mathbb{B}$-packets $\mathbb{V}_{\mathbb{B}}$ and indecomposable packing kernels $\mathbb{K}$, respectively, for intrinsic conflict*

$$C_{in}(\mathbb{V}_{\mathbb{B}}) := (\mathbb{V}_{\mathbb{B}}, \mathbb{C}_{in}(\mathbb{V}_{\mathbb{B}})),\ C_{in}(\mathbb{K}) := (\mathbb{K}, \mathbb{C}_{in}(\mathbb{K})), \tag{69}$$

and for extrinsic conflict

$$C_{ex}(\mathbb{V}_{\mathbb{B}}) := (\mathbb{V}_{\mathbb{B}}, \mathbb{C}_{ex}(\mathbb{V}_{\mathbb{B}})),\ C_{ex}(\mathbb{K}) := (\mathbb{K}, \mathbb{C}_{ex}(\mathbb{K})). \tag{70}$$

3.8 Packing algorithm

The packing graphs G and conflict graphs κ contain the full information of distributability, embeddability, and compatibility, which can be employed to determine the final packing of a quantum circuit. In this section, we provide heuristic algorithms for finding entanglement-efficient packing in the scenarios of unlimited and limited external resources, respectively.

Packing with unlimited auxiliary qubits. Suppose that we have unlimited packing auxiliary qubits, one can determine the ultimate packing and the required amount on packing auxiliary qubits according to Algorithm 27. In Algorithm 27, one first searches all possible minimum vertex covers $\{\mathbb{V}_i\}_i$ of the packing graph $G_{\mathbb{B}}$. From each minimum vertex cover $\mathbb{V}_i$, one obtains a set of packing kernels $\mathbb{K}_i$ rooted on the packets in $\mathbb{V}_i$. The kernel set $\mathbb{K}_i$ induces a subgraph $C_{in}(\mathbb{K}_i)$ of the intrinsic kernel-conflict graph. To solve the intrinsic conflicts in $C_{in}(\mathbb{K}_i)$ with the least amount of additional entanglement, one needs to find a minimum vertex cover $\widetilde{\mathbb{K}}_{i,j}$ of $C_{in}(\mathbb{K}_i)$. One will need to remove the embeddings in $\widetilde{\mathbb{K}}_{i,j}$ and split the corresponding packets in $\mathbb{V}_i$ to get a set of compatible root packets $\mathbb{R}_{i,j}$. The cardinality $|\mathbb{R}_{i,j}|$ is the number of packing processes and hence the entanglement cost for implementing the packing processes rooted on $\mathbb{R}_{i,j}$. One may further reduce the entanglement cost through the merging of the packets in $\mathbb{R}_{i,j}$ with an addon function "`extended_embedding`" based on the extended embedding introduced in Appendix B. The required number of packing auxiliary qubits on the local $A(B)$ system can be determined by the chromatic number $\chi_{i,j}^{(A,B)}$ of the extrinsic packet-conflict $C_{ex}(\mathbb{R}_{i,j}^{(A,B)})$. In the end, one obtains a set of compatible root packets $\mathbb{R}_{i,j}$ and their corresponding required amount of local auxiliary qubits $\chi_{i,j}^{(A,B)}$,

$$\{(\mathbb{R}_{i,j}, \chi_{i,j}^{(A)}, \chi_{i,j}^{(B)})\}_{i,j}. \tag{71}$$

Depending on explicit scenarios, one can choose either the set of root packets with least amount of ebits $\min_{i,j} |\mathbb{R}_{i,j}|$, or least number of local packing ancillas $\min_{i,j} |\chi_{i,j}^{(A,B)}|$, or some particular trade-off between the entanglement consumption and packing auxiliary qubits.

Note that enumerating all minimum vertex covers is a hard problem, as the cardinality of the set of minimum vertex covers can exponentially increase[49]. A heuristic solution is to search for a minimum vertex cover instead of enumerating all of them, which sacrifices the optimality for the entanglement efficiency. Besides, the determination of the chromatic number of a general conflict graph C_{ex} is an NP-hard problem [50]. Nevertheless, one can still efficiently estimate an upper bound on the chromatic number $\chi_{i,j}^{(A,B)}$, which implies a required number of auxiliary memory qubits that guarantees the implementation of the distributing processes. Overall, the time complexity of Algorithm 27 is estimated as $\mathcal{O}(m^2\, 3^{4m/3} 2^{2m})$, while its heuristic implementation can be efficiently implemented with the complexity $\mathcal{O}(m^{7/2})$, where m is the number of nonlocal 2-qubit control-phase gates in a circuit. (see Appendix D.5 for an explanation).

Packing with limited auxiliary qubits. For the packing with extrinsic limits of packing auxiliary qubits, one can employ Algorithm 28 to find an entanglement-efficient packing given a fixed number of auxiliary qubits. In Algorithm 28, the steps up to line 9 are solving the intrinsic conflicts. They are identical to Algorithm 27 for the packing without extrinsic limits. From line 10, the steps for solving extrinsic conflicts under the extrinsic limits must be modified accordingly. Suppose the available amount of local packing auxiliary qubits are (χ_A, χ_B). On line 12, one divides $\mathbb{R}_{i,j}^{A}$ into χ_A colors $\{\mathbb{R}_{i,j,c}^{A}\}_{c=1,\ldots,\chi_A}$ on the system A and same for the system B. Each color represents a packing auxiliary qubit. The goal is to find the optimum $\chi_{A(B)}$-color partition of $\mathbb{R}_{i,j}^{A(B)}$, such that the number of extrinsic $\mathbb{R}_{i,j}^{A(B)}$-conflict edges covered the same-color packets is minimum. A partitioning algorithm can be employed to find the minimum-conflict color partition [51], however, there is no efficient algorithm available yet. For a heuristic solution, one can simply find a $\chi_{A(B)}$-color coloring and continue to the next step.

After obtaining the c-color extrinsic conflict graph $C_{ex}(\mathbb{R}_{i,j,c}^{A(B)})$, one extracts the process kernels $\mathbb{K}_{i,j,c}^{A(B)}$ from $\mathbb{R}_{i,j,c}^{A(B)}$ and finds the minimum vertex cover $\widetilde{\mathbb{K}}_{i,j,c}^{A(B)}$ of the extrinsic $\mathbb{K}_{i,j,c}^{A(B)}$-conflict graph $C_{ex}(\mathbb{K}_{i,j,c}^{A(B)})$. To solve the conflicts, one removes the

Algorithm 27: Packing algorithm without extrinsic limits

$\{\mathbb{V}_i\}_i \leftarrow$ minimum vertex covers of $G_{\mathbb{B}}$;
foreach $\mathbb{V}_i$ **do**
  $\mathbb{K}_i \leftarrow$ the set of packing kernels $\mathbb{K}_i$ that have root packets in $\mathbb{V}_i$;
  $\{\widetilde{\mathbb{K}}_{i,j}\}_j \leftarrow$ minimum vertex covers of $\mathbb{K}_i$-induced intrinsic kernel-conflict graph $C_{in}(\mathbb{K}_i)$;
  foreach $\widetilde{\mathbb{K}}_{i,j}$ **do**
    $\mathbb{R}_{i,j} \leftarrow \bigcup_{\mathcal{V}\in\mathbb{V}_i,\mathcal{B}\in\widetilde{\mathbb{K}}_{i,j}} \mathcal{V}\setminus_B \mathcal{B}$: split the vertex cover $\mathbb{V}_i$ by removing its embedding processes included in $\widetilde{\mathbb{K}}_{i,j}$;
    /* Find possible entanglement reduction with an addon function extended_embedding based on an extended embedding. (See Appendix B for the details of extended embeddings.) */
    $\mathbb{R}_{i,j} \leftarrow$ `extended_embedding`$(\mathbb{R}_{i,j})$ looks for possible extended embeddings in $\mathbb{R}_{i,j}$;
    $\left(\mathbb{R}_{i,j}^{(A)}, \mathbb{R}_{i,j}^{(B)}\right) \leftarrow \mathbb{R}_{i,j}$: divided into two sets of local packets;
    $C_{ex}(\mathbb{R}_{i,j}^{(A,B)}) \leftarrow (\mathbb{R}_{i,j}^{(A,B)}, \mathbb{C}_{ex}(\mathbb{R}_{i,j}^{(A,B)}))$: update the extrinsic local packet-conflict graphs according to Eq. (66);
    $\chi_{i,j}^{(A,B)} \leftarrow$ chromatic number of $C_{ex}(\mathbb{R}_{i,j}^{(A,B)})$;
  end
end
Result: $\{(\mathbb{R}_{i,j}, \chi_{i,j}^{(A)}, \chi_{i,j}^{(B)})\}_{i,j}$

embedding processes $\widetilde{\mathbb{K}}_{i,j,c}^{A(B)}$ from the packet $\mathbb{R}_{i,j,c}^{A(B)}$ and obtains a set of compatible c-color packets $\widetilde{\mathbb{R}}_{i,j,c}^{A(B)}$. In the end, one obtains the candidate sets of root packets $\widetilde{\mathbb{R}}_{i,j}$ through the union of the c-color root packets $\widetilde{R}_{i,j,c}^{A(B)}$ The ultimate set of root packets is the one that has the minimum cardinality.

$$\widetilde{\mathbb{R}} = \arg\min_{\mathbb{R}_{i,j}} |\mathbb{R}_{i,j}|. \tag{72}$$

The corresponding quantum circuit can be then locally implemented with the packing process rooted on the packets in $\widetilde{\mathbb{R}}$ with the assistance of χ_A and χ_B local packing auxiliary qubits and $|\widetilde{\mathbb{R}}|$ ebits of entanglement. Note that Algorithm 28 has the same time complexity as Algorithm 27 (see Appendix D.5 for an explanation).

4 Identification of packing graphs and conflict graphs

The packing algorithms in Section 3.8 determine the distributable packets based on the distributability and embeddability of a quantum circuit, which are fully described by the packing graphs and conflict graphs of a circuit. Therefore, one needs to identify these characteristic graphs before implementing the packing algorithms. Since single-qubit and two-qubit gates form a universal set of quantum circuits, all quantum circuits can be decomposed into single-qubit and two-qubit gates. We therefore consider the identification of packing graphs and conflict graphs of quantum circuits consisting of only single-qubit and two-qubit gates. For the vertices, we need to identify the set of distributable packets, which is equivalent to the identification of indecomposable embeddings. For the edges, we need to identify the packing edges associated with two-qubit gates, the intrinsic conflict edges associated with incompatible embeddings, and the extrinsic conflict edges associated with external resource limits.

4.1 Identification of distributable packets

According to Definition 16, in a circuit consisting of 1-qubit and 2-qubit gates, the elements in a distributable packet are the gate nodes of 2-qubit gates. According to Definition 23, a packing edge between two distributable packets is always associated with a two-qubit gate, which is referred to as a packing-edge gate. A packing-edge gate of particular interest is the control-phase gate $C_V(\theta)$ in Eq. (17). One can show that the connectivity of packing graphs can be identified solely with control-phase gates, since all packing-edge gates are locally equivalent to a control-phase gate up to single-qubit distributable gates.

Lemma 29 (The connecting 2-qubit gate)
A two-qubit gate g acting on q_1 and q_2 is distributable on both of the grid points (q_1,t) and (q_2,t), if and only if it is equivalent to a control-phase gate up to single-qubit distributable gates,

$$g = (D_{q_1}\otimes D_{q_2})\, C_V(\theta)\, (\widetilde{D}_{q_1}\otimes \widetilde{D}_{q_2}), \tag{73}$$

where D_{q_i} and $\widetilde{D}_{q_i}$ are single-qubit distributable gates acting on q_i.
Proof: See Appendix D.6 ■

Algorithm 28: Packing algorithm with extrinsic limits

$\chi_A, \chi_B \leftarrow$ available amount of auxiliary qubits on each local systems;
$\{\mathbb{V}_i\}_i \leftarrow$ minimum vertex covers of $G_{\mathbb{B}}$;
foreach $\mathbb{V}_i$ **do**
  $\mathbb{K}_i \leftarrow$ the set of packing kernels $\mathbb{K}_i$ that have root packets in $\mathbb{V}_i$;
  $\{\widetilde{\mathbb{K}}_{i,j}\}_j \leftarrow$ minimum vertex covers of $\mathbb{K}_i$-induced intrinsic kernel-conflict graph $C_{in}(\mathbb{K}_i)$;
  foreach $\widetilde{\mathbb{K}}_{i,j}$ **do**
    $\mathbb{R}_{i,j} \leftarrow \bigcup_{\mathcal{V}\in\mathbb{V}_i, \mathcal{B}\in\widetilde{\mathbb{K}}_{i,j}} \mathcal{V} \setminus_B \mathcal{B}$: split the vertex cover $\mathbb{V}_i$ by removing its embedding processes included in $\widetilde{\mathbb{K}}_{i,j}$;
    `/* Find possible entanglement reduction with an addon function extended_embedding based on an extended embedding. (See Appendix B for the details of extended embeddings.) */`
    $\mathbb{R}_{i,j} \leftarrow$ `extended_embedding`$(\mathbb{R}_{i,j})$ looks for possible extended embeddings in $\mathbb{R}_{i,j}$;
    $(\mathbb{R}^{(A)}_{i,j}, \mathbb{R}^{(B)}_{i,j}) \leftarrow \mathbb{R}_{i,j}$: divided into two sets of local packets.
    $C_{ex}(\mathbb{R}^{(A,B)}_{i,j}) \leftarrow (\mathbb{R}^{(A,B)}_{i,j}, \mathbb{C}_{ex}(\mathbb{R}^{(A,B)}_{i,j}))$: update the extrinsic local packet-conflict graphs according to (66) ;
    foreach $k \in \{A, B\}$ **do**
      $\{\mathbb{R}^{(k)}_{i,j;c}\}_{c=1,...,\chi_k} \leftarrow$ divide $\mathbb{R}^{(k)}_{i,j}$ into χ_k colors, $\mathbb{R}^{(k)}_{i,j;c}$ with $c = 1, ..., \chi_k$, such that the number of all $\mathbb{R}^{(k)}_{i,j;c}$-induced conflict edges $\sum_c |\mathbb{C}_{ex}(\mathbb{R}^{(k)}_{i,j;c})|$ is minimum ;
      foreach *color* $c \in \{1, ..., \chi_k\}$ **do**
        `/* Solve the conflicts for each color` c. `*/`
        $\mathbb{K}^{(k)}_{i,j,c} \leftarrow$ the set of packing kernels $\mathbb{K}^{(k)}_{i,j,c}$ that have root packets in $\mathbb{R}^{(k)}_{i,j;c}$;
        $\{\widetilde{\mathbb{K}}^{(k)}_{i,j,c}\}_c \leftarrow$ minimum vertex covers of $\mathbb{K}^{(k)}_{i,j,c}$-induced extrinsic kernel-conflict graph $C_{ex}(\mathbb{K}^{(k)}_{i,j,c})$;
        $\widetilde{\mathbb{R}}^{(k)}_{i,j;c} \leftarrow \bigcup_{\mathcal{V}\in\mathbb{R}^{(k)}_{i,j;c}, \mathcal{B}\in\widetilde{\mathbb{K}}^{(k)}_{i,j,c}} \mathcal{V} \setminus_B \mathcal{B}$: Split the c-color vertex cover $\mathbb{R}^{(k)}_{i,j;c}$ by removing its embedding processes included in $\widetilde{\mathbb{K}}^{(k)}_{i,j,c}$;
      end
    end
    $\widetilde{\mathbb{R}}_{i,j} \leftarrow \bigcup_{k,c} \widetilde{\mathbb{R}}^{(k)}_{i,j;c}$.
  end
end
$\widetilde{\mathbb{R}} \leftarrow \arg\min_{\widetilde{\mathbb{R}}_{i,j}} |\widetilde{\mathbb{R}}_{i,j}|$;
Result: $\{\widetilde{\mathbb{R}}, \chi_A, \chi_B\}$

As a result of Lemma 29, to reveal the connectivity of distributable packets, we need to convert control unitaries to control-phase gate through

$$\begin{aligned} C_{U_0,U_1} =& |0\rangle\langle 0| \otimes U_0 + |1\rangle\langle 1| \otimes U_1 \\ =& (V(\theta_0) \otimes U_0 W) C_V(\theta_1 - \theta_0) (\mathbb{1} \otimes W^\dagger), \end{aligned} \quad (74)$$

where W is the transformation matrix for the diagonalization of $U_0^\dagger U_1$,

$$U_0^\dagger U_1 = e^{\mathbb{i}\,\theta_0} W\, V(\theta_1 - \theta_0)\, W^\dagger, \quad (75)$$

and θ_i are the eigenphases of $U_0^\dagger U_1$.

As it is shown in Fig. 17 (a), after the control-phase conversion , a circuit can be decomposed into a sequence of two-qubit blocks,

$$U = \prod_t \widetilde{U}_t, \quad (76)$$

where the elemental two-qubit block $\widetilde{U}_t$ is shown in Fig. 17 (b) and given by

$$\widetilde{U}_{qq'} = (G_q \otimes G'_{q'})\, C_V(\theta)\, (F_q \otimes F'_{q'}). \quad (77)$$

In Fig. 17 (a), we use the symbols white circles for single-qubit distributable gates, white squares for identity gates, black squares for Hadamard gates, gray squares for irrelevant gates.

The control-phase nodes in Fig. 17 (a) are the nodes that we want to pack into packing processes through the embedding of the unitaries between them. If each two-qubit block $\widetilde{U}_t$ in Eq. (76) is q-rooted embeddable, then the total unitary U is also q-rooted embeddable. One can employ this sufficient condition to determine the q-rooted embeddability of U and identify the distributable packets in a circuit. Note that we do not exclude the possible embeddability of U

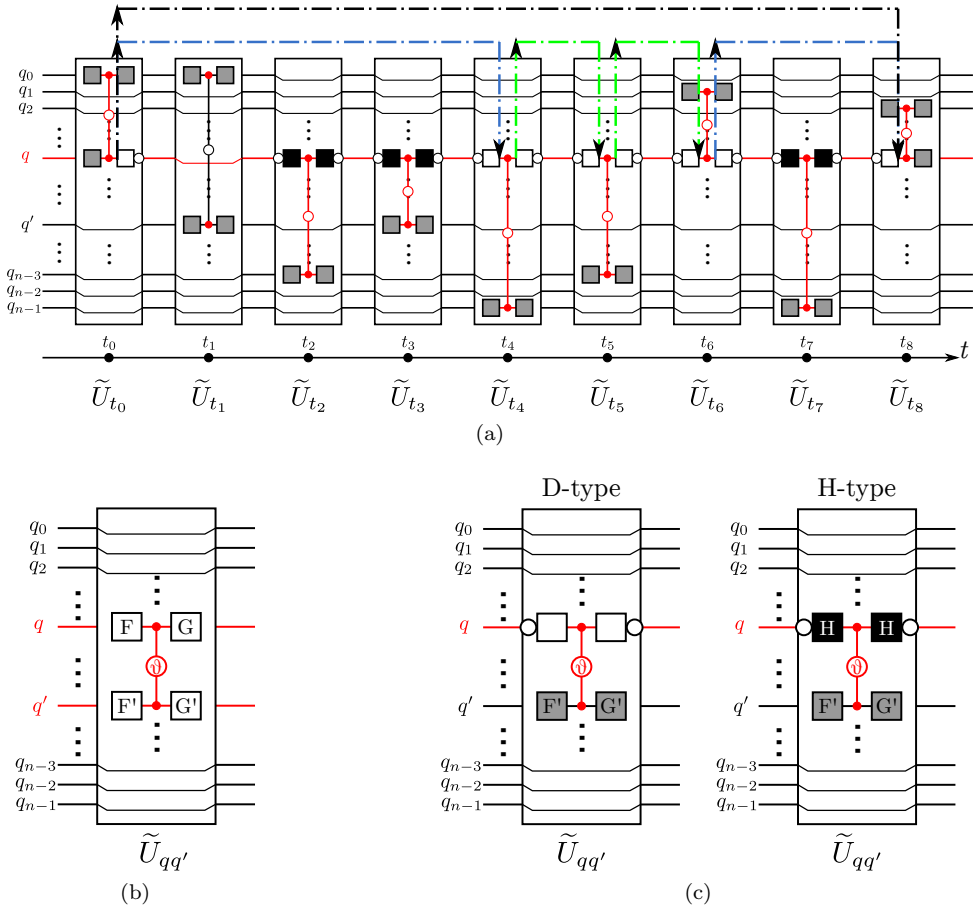


Figure 17: Identification of distributable packets: white circles represent distributable single-qubit gates, white squares represent identity gates, black square represents Hadamard gates, and gray square presents q-irrelevant gates. (a) A hopping embedding (black dot-dashed) decomposed into indecomposable hopping embeddings (blue dot-dashed) and neighbouring embeddings (green dash-dot). (b) A two-qubit block. (c) The D-type and H-type blocks.

that possesses a non-embeddable block $\widetilde{U}_t$.

After the control-phase conversion, a quantum circuit consisting of one-qubit and two-qubit gates can be decomposed into two types of embedding units, which are shown in Fig. 17 (c). The embedding rules for such a quantum circuit can be summarized as the embedding rules for two-qubit blocks $\widetilde{U}_{qq'}$ as follows.

Corollary 30 (Embedding rules for 2-qubit blocks) *A two-qubit block $\widetilde{U}_{qq'}$ in Eq. (77) is q-rooted embeddable in the following two cases*

1. *(D-type, Fig. 18 (a,b)) F_q and G_q are distributable, which means diagonal or antidiagoanl. The embedding rule is*

$$\mathcal{B}_q(\widetilde{U}_{qq'}) = \widetilde{U}_{qq'} X_e^n, \tag{78}$$

where n is the number of anti-diagonal gates in $\{G_q, F_q\}$.

2. *(Global H-type CZ, Fig. 18 (c)) The control phase gate $C_V(\theta)$ is global with a phase of $\theta = \pi$. The single-qubit gates F_q and G_q can be decomposed as*

$$F_q = H_q D_q, \quad G_q = \widetilde{D}_q H_q, \tag{79}$$

where D_q and $\widetilde{D}_q$ are diagonal or anti-diagonal. The embedding rule is

$$\mathcal{B}_q(\widetilde{U}_{qq'}) = (G'_{q'} C_{q',X_e} X_e^n G_{q'}^{'\dagger})\, \widetilde{U}_{qq'} \tag{80}$$

$$= \widetilde{U}_{qq'}\, (F_{q'}^{'\dagger} C_{q',X_e} X_e^n F'_{q'}), \tag{81}$$

where n is the number of anti-diagonal gates in $\{D_q, \widetilde{D}_q\}$.

The block $\widetilde{U}_{qq'}$ is not embeddable in the following case,

3. *(Local H-type, Fig. 18 (d)) $\widetilde{U}_{qq'}$ is local H-type.*

Proof: For a D-type block, $\widetilde{U}_{qq'}$ is distributable on q, and has embedding rules in Eq. (27) and (28) according to Theorem 10. For a global H-type CZ block, one can derive the primitive embedding rule through

$$C_{q,X_e} \widetilde{U}_{qq'} C_{q,X_e} = (G'_{q'} C_{q',X_e} X_e^k G_{q'}^{'\dagger})\, \widetilde{U}_{qq'} \tag{82}$$

$$= \widetilde{U}_{qq'}\, (F_{q'}^{'\dagger} C_{q',X_e} X_e^k F'_{q'}). \tag{83}$$

For a local H-type block, it is not distributable according to Theorem 5. As a result of Theorem 10, a q-rooted non-distributable local unitary can never be q-rooted embeddable. ■

Now, we have the embedding rules for single-qubit gates $\mathbb{B}_1$ (Theorem 10) and two-qubit gates $\mathbb{B}_2$ (Corollary 30), which form a set of embedding rules $\mathbb{B}$ for the packing of gate nodes of control-phase gates,

$$\mathbb{B} = \{\mathbb{B}_1 := \text{Theorem 10}, \mathbb{B}_2 := \text{Corollary 30}\}. \quad (84)$$

Employing the set of embedding rules $\mathbb{B}$, one can identify the set of distributable $\mathbb{B}$-packet $\mathbb{V}_{\mathbb{B}}$ of a quantum circuit.

As it is shown Fig. 17 (a), to determine the $\mathbb{B}$-packability of the nodes (q, t_0) and (q, t_8), one needs to determine the q-rooted embeddability of the unitary between them

$$W_{t_8;t_0} = F_{q,t_8}\widetilde{U}_{t_7}\widetilde{U}_{t_6}\widetilde{U}_{t_5}\widetilde{U}_{t_4}\widetilde{U}_{t_3}\widetilde{U}_{t_2}\widetilde{U}_{t_1}G_{q,t_0}. \quad (85)$$

In this example, the embedding of $W_{t_8;t_0}$ can be decomposed into four indecomposable embeddings $W_{i=1,\ldots,4}$ between control-phase gates,

$$W_{t_8;t_0} = W_4\, C_{q,V}^{(t_6)}\, W_3\, C_{q,V}^{(t_5)}\, W_2\, C_{q,V}^{(t_4)}\, W_1. \quad (86)$$

where

$$W_1 = F_{q,t_4}\widetilde{U}_{t_3}\widetilde{U}_{t_2}\widetilde{U}_{t_1}G_{q,t_0}, \quad (87)$$
$$W_2 = F_{q,t_5}G_{q,t_4}, \quad (88)$$
$$W_3 = F_{q,t_6}G_{q,t_5}, \quad (89)$$
$$W_4 = F_{q,t_8}\widetilde{U}_{t_7}G_{q,t_6}. \quad (90)$$

The embeddings of W_2 and W_3 (green dot-dashed lines) do not include any embedding unit. Their embeddability can be easily determined with the embedding rules $\mathbb{B}_1$ for single qubit gates (Theorem 10). These two embeddings do not contain embedding units. They identify the packet of neighbouring control-phase nodes $\mathcal{V}_{q,\{t_4,t_5\}}$ and $\mathcal{V}_{q,\{t_5,t_6\}}$, respectively. We call such an embedding a *neighbouring embedding*, and its corresponding distributable packet a *neighbouring distributable packet*.

On the other hand, the embeddings of W_1 and W_4 (blue dot-dashed lines) contain embedding units, which allow the packing of remote control-phase nodes $\mathcal{V}_{q,\{t_0,t_4\}}$ and $\mathcal{V}_{q,\{t_6,t_8\}}$, respectively. We call such an embedding a *hopping embedding*, and its corresponding distributable packet a *hopping distributable packet*. One need the $\mathbb{B}_2$ embedding rules (Corollary 30) to identify hopping embeddings.

The embedding of $W_{t_8;t_0}$ in Eq. (85) is also a hopping embedding, which forms the hopping distributable packet $\mathcal{V}_{q,\{t_0,t_8\}}$. However, according to Definition 19, $W_{t_8;t_0}$ is decomposable. In this example, one can see that the indecomposable embeddings $\{W_1, ..., W_4\}$ lead to the largest distributable packet $\mathcal{V}_{q,\{t_0,t_4,t_5,t_6,t_8\}}$, which is merged according to Lemma 17,

$$\begin{aligned}\mathcal{V}_{q,\{t_0,t_4,t_5,t_6,t_8\}} =& \mathcal{V}_{q,\{t_0,t_4\}} \cup_D \mathcal{V}_{q,\{t_4,t_5\}} \\ & \cup_D \mathcal{V}_{q,\{t_5,t_6\}} \cup_D \mathcal{V}_{q,\{t_6,t_8\}}. \end{aligned} \quad (91)$$

It is therefore necessary to identify all indecomposable embeddings of a quantum circuit to fully explore its packability. The identification of distributable packets of a circuit is therefore equivalent to the identification of all the indecomposable embeddings.

For neighbouring embeddings, the identification according to $\mathbb{B}_1$ (Theorem 10) always returns indecomposable embeddings. For indecomposable hopping embeddings constructed with the embedding units in Corollary 30, one can search them with the following condition.

Lemma 31 (Indecomposable hopping embedding) *Two distributable nodes $\{(q, t_i), (q, t_j)\}$ with $i \leq j - 2$ form an indecomposable hopping distributable $\mathbb{B}$-packet $\mathcal{V}_{q,\{t_i,t_j\}}$, if and only if the unitary $W_{t_j;t_i}$ between (q, t_i) and (q, t_j) can be decomposed as*

$$W_{t_j;t_i} = D'_{q,t_j}\left(\prod_{k:i<k<j}\widetilde{U}_{t_k}\right)D_{q,t_i}, \quad (92)$$

where D'_{q,t_j} and D_{q,t_i} are diagonal or anti-diagonal, and each embedding unit $\widetilde{U}_{t_k}$ is global H-type with the control phase $\theta = \pi$.
Proof: See Appendix D.7. ■

This lemma implies that a control-phase gate, which is local or has a control phase $\theta \neq \pi$, cannot be a part of indecomposable hopping embedding.

Corollary 32 *Let $\mathcal{V}_{q,\{t_i,t_j\}}$ be an indecomposable distributable packet. All the control-phase gates $C_V^{q,q_t}(\theta_t)$ with $t_i < t < t_j$ between the nodes (q, t_i) and (q, t_j) must be global and $\theta_t = \pi$.*
Proof: This is a direct result of Lemma 31. ■

Now, we have all the essential tools to develop an identification algorithm for distributable packets, which is summarized in Algorithm 33. To implement this algorithm, one needs two functions, *neighbouring()* and *hopping()*, to determine the q-rooted neighbouring and hopping embeddings, respectively. The neighbouring embeddings can be determined according to Algorithm 34 through the embedding rules $\mathbb{B}_1$ in Theorem 10, which leads to a set of distributable $\mathbb{B}_1$-packets $\mathbb{V}_{\mathbb{B}_1}$. Meanwhile the indecomposable hopping embeddings can be determined according to Algorithm 35 through the embedding rules $\mathbb{B}_2$ in Corollary 30. The indecomposable hopping embeddings determined by Algorithm 35 remotely connect two neighbouring distributable packets in $\mathbb{V}_{\mathbb{B}_1}$. One can then merge two remote neighbouring distributable packets through the indecomposable hopping embeddings in $\mathbb{B}_2$ and obtain the

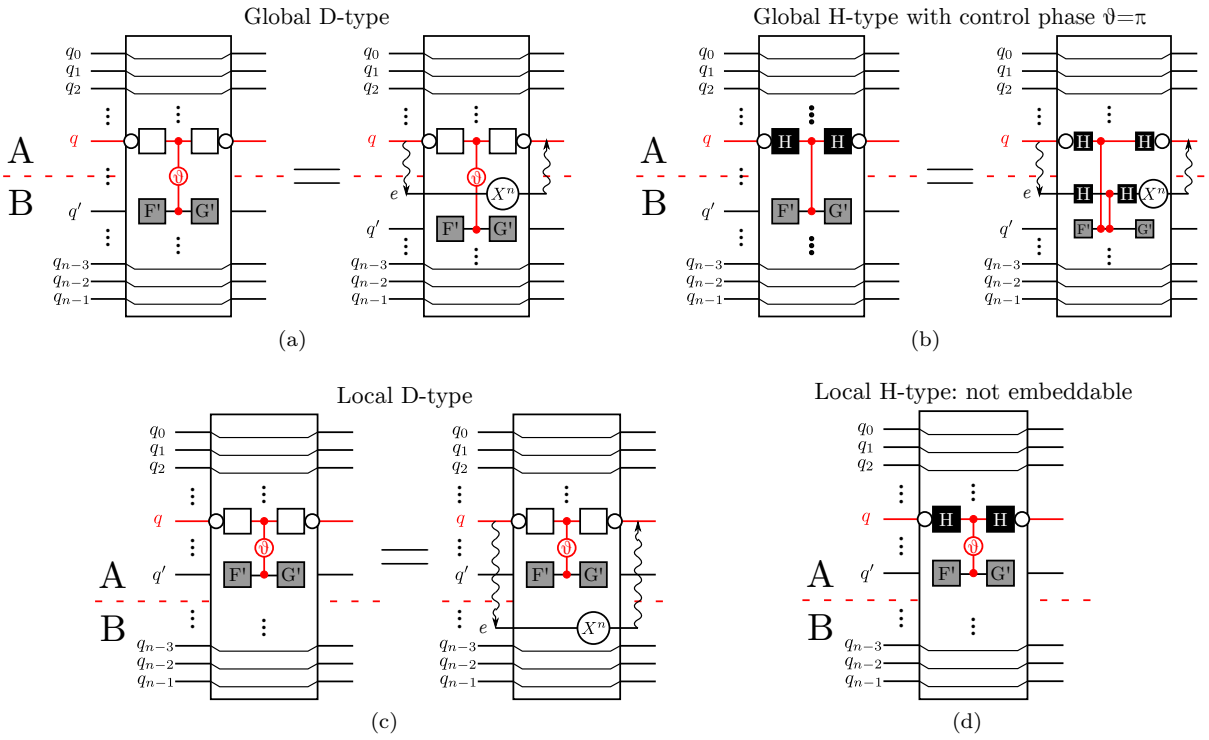


Figure 18: Embedding rules of two-qubit embedding units in Corollary 30. (a) Global D-type. (b) Global H-type with phase $\theta = \pi$. (c) Local D-type. (d) Local H-type (non-embeddable).

ultimate set of distributable packets $\mathbb{V}_{\mathbb{B}}$. The complexity of the identification of distributable packets is estimated to be $\mathcal{O}(nd^2)$, where n and d are the total qubit number and depth of a circuit (see Appendix D.8 for an explanation).

In the identification algorithm for indecomposable hopping embeddings (Algorithm 35), from line 10 to 28, the ultimate goal is to shift the Hadamard gates between two control-phase nodes into an embedding block through commuting them with the single-qubit distributable gates to form H-type embedding units, which is demonstrated by the examples in Fig. 19. In Fig. 19 (a), it shows a quantum circuit after the control-phase conversion. In this example, we look at the packing on the qubit q. The white squares are identity gates.

The single qubit gates (red diamonds) between the control phase nodes are converted into three types according to Eq. (94). In Fig 19 (b), these single-qubit gates are then grouped into $\mathbb{S}$ for starting-gate candidates, $\mathbb{E}$ for ending-gate candidates, $\mathbb{M}$ for intermediate-gate candidates, and $\mathbb{D}$ for distributable intermediate-gate candidates according to line 3 to 6 in Algorithm 35. The gates in the groups $\mathbb{S}$ and $\mathbb{E}$ are the potential starting and ending single-qubit gates of an indecomposable hopping embedding, since they can be decomposed into gates containing only one Hadamard gate. The gates in the group $\mathbb{M}$ can only be the intermediate single-qubit gates in an indecomposable hopping embedding, since they can only be converted into the gates containing two Hadamard gates. Note that all gates in $\mathbb{S}$, $\mathbb{E}$, $\mathbb{D}$ and $\mathbb{M}$ can serve as single-qubit intermediate gates in an indecomposable hopping emdedding, since they can be decomposed into a gate that contains two Hadamards.

In Fig. 19 (b), the red-dashed two-qubit blocks $\widetilde{U}_{t_0}$, $\widetilde{U}_{t_6}$, and $\widetilde{U}_{t_9}$ contain a local control-phase gate or a global control-phase gate with the phase $\theta \neq \pi$, which can never be involved in an indecomposable hopping embedding according to Lemma 31. One can therefore divide the starting-gate and ending-gate groups into different divisions, such as $\mathbb{S}^{(q)}_{1,2}$ and $\mathbb{E}^{(q)}_{1,2}$. In each division, one can then construct a set of candidate q-rooted packets $\mathbb{P}^{(q)}_k$ from $\mathbb{S}^{(q)}_k$ and $\mathbb{E}^{(q)}_k$. These steps are summarized on line 7 to 9 in Algorithm 35.

For each candidate packet $\{(q,t_i),(q,t_j)\} \in \mathbb{P}^{(q)}_k$, one can then check the q-rooted embeddability following line 10 to 28. For the example of the candidate packet $\{(q,t_1),(q,t_6)\}$ in Fig. 19 (b), the single-qubit gate $W_{t_4;t_3}$ is an $\mathbb{S}$-type and $\mathbb{E}$-type gate, while $W_{t_5;t_4}$ is $\mathbb{D}$-type. One first converts these gates into an $\mathbb{M}$-type gate following line 12 to 23. In the next step, one checks the embeddability of $W_{t_6;t_1}$ by shifting the red squares into the embedding units according to line 25 to 28, which is as shown in Fig. 19 (c), If there are no non-embeddable gates remaining outside the embedding units, then $\{(q,t_1),(q,t_6)\}$ forms a $\mathbb{B}$-distributable packet $\mathcal{V}_{q,\{t_1,t_6\}}$. Explicitly, the condition for a successful construction of sequential embedding units is given on line 25. A counterexample

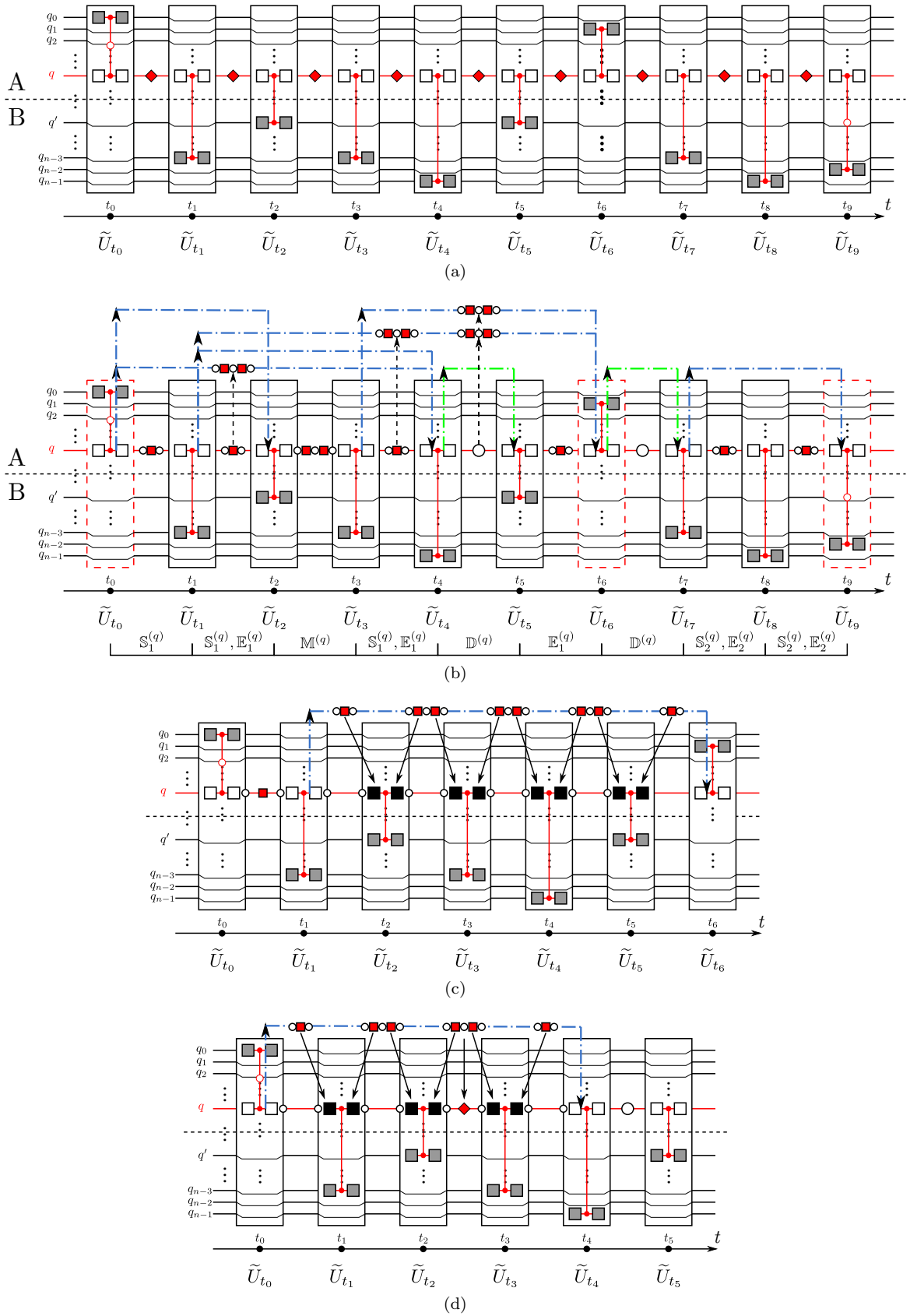


Figure 19: Algorithm for the identification of distributable packets. (a) A circuit after control-phase conversion. The white squares are identity gates, the white circles are single-qubit distributable gates, and the red diamonds are the single-qubit non-distributable gates. (b) The circuit after the 1st step in Algorithm 33. The red squares are Hadamard gates. (c) The determination of the packability of $\{(q, t_1), (q, t_6)\}$. (d) The determination of the packability of $\{(q, t_0), (q, t_4)\}$.

Algorithm 33: Identification of distributable packets

$Q \leftarrow$ initialize a circuit with the control-phase conversion (Eq. (74));
$\mathbb{V}_0 \leftarrow$ the set of trivial distributable packets of the circuit Q;

$$\mathbb{V}_0 := \{\mathcal{V}_{q,t} = (q,t,\theta) : (q,t) \text{ is a node of a control-phase gate } C_V(\theta).\} \tag{93}$$

foreach *subsequential nodes* $\{(q,t_i,\theta_i),(q,t_{i+1},\theta_{i+1})\} \subseteq \mathbb{V}_0$ **do**
  $\{W_{t_{i+1;t_i}}\}_i \leftarrow$ conversion of single-qubit gates $W_{t_{i+1};t_i}$;

$$W_{t_{i+1};t_i} = \begin{cases} D, & D \text{ diagonal or anti-diagonal;} \\ V(\gamma)\,H\,V(\alpha), & Z\text{-rotation-equivalent to } H; \\ V(\gamma)\,H\,V(\beta)\,H\,V(\alpha), & \text{else, Euler decomposition.} \end{cases} \tag{94}$$

end
foreach *qubit* q *in* Q **do**
  $\{\mathbb{B}_1^{(q)}, \mathbb{V}_{\mathbb{B}_1}^{(q)}\} \leftarrow$ `neighbouring`$(q,\ \mathbb{V}_0,\ \{W_{t_{i+1;t_i}}\}_i)$: identification of neighbouring distributable packets;
  $\{\mathbb{B}_2^{(q)}, \mathbb{H}^{(q)}\} \leftarrow$ `hopping`$(q,\ \mathbb{V}_0,\ \{W_{t_{i+1;t_i}}\}_i)$: identification of hopping distributable packets;
  $\mathbb{B}^{(q)} \leftarrow \mathbb{B}_1^{(q)} \cup \mathbb{B}_2^{(q)}$;
  $\mathbb{V}_{\mathbb{B}}^{(q)} \leftarrow \mathbb{V}_{\mathbb{B}_1}^{(q)}$ merged by $\mathbb{H}^{(q)}$ according to Lemma 17;
end
Result: $\{\mathbb{B}, \mathbb{B}_1, \mathbb{B}_2, \mathbb{V}_{\mathbb{B}}, \mathbb{V}_{\mathbb{B}_1}\}$

Algorithm 34: Identification of neighbouring distributable packets on q

Function `neighbouring`$(q,\ \mathbb{V}_0,\ \{W_{t_{i+1},t_i}\}_i)$:
  $\mathbb{D}^{(q)} \leftarrow \{\mathcal{V}_{q;t_i,t_{i+1}} = \{(q,t_i,\theta_i),(q,t_{i+1},\theta_{i+1})\} : W_{t_{i+1};t_i} = D\}$: the set of neighbouring distributable packets;
  $\mathbb{B}_1^{(q)} \leftarrow \{(\mathcal{V}_{q;t_i,t_{i+1}}, \mathcal{B}_q(W_{t_{i+1};t_i})) : W_{t_{i+1};t_i} = D\}$: the set of neighbouring embedding kernels determined by Theorem 10;
  $\mathbb{V}_{\mathbb{B}_1}^{(q)} \leftarrow$ the set of distributable packets merged from $\mathbb{D}^{(q)}$ according to Lemma (17);
  return $\{\mathbb{B}_1^{(q)}, \mathbb{V}_{\mathbb{B}_1}^{(q)}\}$;

of embeddability for the candidate $\{(q,t_0),(q,t_4)\}$ is shown in Fig. 19 (d), in which a non-embeddable gate (red diamond) is lying between the embedding units $\widetilde{U}_{t_2}$ and $\widetilde{U}_{t_3}$. Accordingly, $\{(q,t_0),(q,t_4)\}$ is not embeddable.

In the end, Algorithm 35 returns the set of indecomposable hopping embeddings $\mathbb{B}_2$ and the set of their corresponding indecomposable distributable packets $\mathbb{H}$.

4.2 Identification of packing and conflict edges

The packing edges can be straightforwardly assigned to control-phase gates. A global control-phase gate $C_V^{(q,q')}(\theta_t)$ acting on $\{q,q'\}$ at the depth t defines an undirected edge between two distributable packets $\mathcal{V}_{q,T_{\mathbb{B}}}$ and $\mathcal{V}_{q',T'_{\mathbb{B}}}$ in $\mathbb{V}_{\mathbb{B}}$, where $t \in T_{\mathbb{B}} \cap T'_{\mathbb{B}}$,

$$\mathcal{E}^{(\mathbb{B})}_{q,q';t} := \{\mathcal{V}_{q,T_{\mathbb{B}}}, \mathcal{V}_{q',T'_{\mathbb{B}}}\} \text{ with } t \in T_{\mathbb{B}} \cap T'_{\mathbb{B}}. \tag{100}$$

These edges form an edge set of the $\mathbb{B}$-packing graph,

$$G_{\mathbb{B}} = (\mathbb{V}_{\mathbb{B}}, \mathbb{E}_{\mathbb{B}}) \text{ with } \mathbb{E}_{\mathbb{B}} := \{\mathcal{E}_{q,q';t} : \exists C_V^{(q,q')}(\theta_t)\}. \tag{101}$$

For the examples in Fig. 19 (c,d), if $\mathcal{V}_1 = \{(q,t_0),(q,t_4)\}$ and $\mathcal{V}_2 = \{(q,t_1),(q,t_5)\}$ are both distributable $\mathbb{B}$-packets, they are nested and identified by two nested hopping embeddings. If the two nested hopping embeddings are incompatible, only one of them can be packed in one packing process. It means that one can not simultaneously pack two distributable packets identified by incompatible embeddings. The compatibility of two distributable packets can be determined through the following theorem.

Theorem 36 (Compatibility of embedding)
For a circuit consisting of single-qubit and two-qubit gates, two indecomposable embeddings $\mathcal{B}_{q;\tau_1,\tau_2}$ and $\mathcal{B}_{q';\tau'_1,\tau'_2}$ are incompatible, if and only if there exists a global control-Z gate $C_Z(q,q';t)$ acting on $\{q,q'\}$ at the depth t with $\tau_1 < t < \tau_2$ and $\tau'_1 < t < \tau'_2$.
Proof: See Appendix D.10 ■

Algorithm 35: Identification of hopping distributable packets on q

Function `hopping(`q, $\{W_{t_{i+1},t_i}\}_i$`)`**:**

$\{\mathbb{B}_2^{(q)}, \mathbb{H}^{(q)}\} \leftarrow \{\emptyset, \emptyset\}$;

$\mathbb{S}^{(q)} \leftarrow \Big\{\{(q,t_i,\theta_i),(q,t_{i+1},\theta_j)\} : W_{t_{i+1};t_i} = V(\gamma_i')\,H\,V(\alpha_i'), \text{and } \theta_{i+1} = \pi\Big\}$: the set of single-qubit starting-gate candidates.

$\mathbb{E}^{(q)} \leftarrow \Big\{\{(q,t_i,\theta_i),(q,t_{i+1},\theta_j)\} : W_{t_{i+1};t_i} = V(\gamma_i')\,H\,V(\alpha_i'), \text{and } \theta_i = \pi\Big\}$: the set of single-qubit ending-gate candidates.

$\mathbb{M}^{(q)} \leftarrow \Big\{\{(q,t_i,\theta_i),(q,t_{i+1},\theta_{i+1})\} : W_{t_{i+1};t_i} = V(\gamma_i)\,H\,V(\beta_i)\,H\,V(\alpha_i), \text{and } \theta_i = \theta_{i+1} = \pi\Big\}$: the set of single-qubit intermediate-gate candidates.;

$\mathbb{D}^{(q)} \leftarrow \Big\{\{(q,t_i,\theta_i),(q,t_{i+1},\theta_{i+1})\} : W_{t_{i+1};t_i} = D, \text{and } \theta_i = \theta_{i+1} = \pi\Big\}$: the set of single-qubit intermediate-gate candidates, which are distributable.;

$\{\mathbb{S}_k^{(q)}, \mathbb{E}_k^{(q)}\}_k \leftarrow$ division of $\mathbb{S}^{(q)}$ and $\mathbb{E}^{(q)}$ by local control-phase gates and global control-phase with $\theta \neq \pi$ (Corollary 32). Let $T(\mathbb{P})$ be the set of node depths in $\mathbb{P}$, $T(\mathbb{P}) := \{t : \exists p \in \mathbb{P}, s.t.(q,t,\theta_t) \in p\}$

$$\Big\{\mathbb{S}_k^{(q)} \subseteq \mathbb{S} : C_V^{(q,q_t')}(\theta_t) \text{ is global and } \theta_t = \pi \text{ for all } \min T(\mathbb{S}_k^{(q)}) < t < \max T(\mathbb{S}_k^{(q)})\Big\}_k. \tag{95}$$

$$\Big\{\mathbb{E}_k^{(q)} \subseteq \mathbb{E} : C_V^{(q,q_t')}(\theta_t) \text{ is global and } \theta_t = \pi \text{ for all } \min T(\mathbb{E}_k^{(q)}) < t < \max T(\mathbb{E}_k^{(q)})\Big\}_k. \tag{96}$$

foreach *division* k **do**

$\mathbb{P}_k^{(q)} \leftarrow$ get the set of potential indecomposable distributable packets

$$\mathbb{P}_k^{(q)} = \Big\{\{(q,t_i,\theta_i),(q,t_j,\theta_j)\} : \{(q,t_i,\theta_i),(q,t_{i+1},\theta_{i+1})\} \in \mathbb{S}_k^{(q)}, \{(q,t_{j-1},\theta_{j-1}),(q,t_j,\theta_j)\} \in \mathbb{E}_k^{(q)}.\Big\} \tag{97}$$

foreach $\{(q,t_i),(q,t_j)\} \in \mathbb{P}_k^{(q)}$ **do**

`/* Check the` q`-rooted embeddability of` $W_{t_j;t_i}$ `according to Lemma 31. */`

$\widetilde{\mathbb{M}} \leftarrow \emptyset$ **foreach** $k \in [i+1, j-2]$ **do**

switch $\{(q,t_k),(q,t_{k+1})\}$ **do**

case $\{(q,t_k),(q,t_{k+1})\} \in \mathbb{S}^{(q)}$ *or* $\{(q,t_k),(q,t_{k+1})\} \in \mathbb{E}^{(q)}$ **do**

Convert the $\mathbb{S}$-type or $\mathbb{E}$-type gates into $\mathbb{M}$-type gates through

$$W_{t_{k+1};t_k} = V(\gamma_k')\,H\,V(\alpha_k') = V(\gamma_k)\,H\,V(\beta_k)\,H\,V(\alpha_k), \tag{98}$$

where $(\alpha_k, \beta_k, \gamma_k) = (\alpha_k' + \frac{1}{2}\pi, \frac{1}{2}\pi, \gamma_k' + \frac{1}{2}\pi)$;

end

case $\{(q,t_k),(q,t_{k+1})\} \in \mathbb{D}^{(q)}$ **do**

Convert $\mathbb{D}$-type gates into $\mathbb{M}$-type gates through

$$W_{t_{k+1};t_k} = H\,V(n\pi)\,H\,V(\alpha_k') = V(\gamma_k)\,H\,V(\beta_k)\,H\,V(\alpha_k), \tag{99}$$

where $(\alpha_k, \beta_k, \gamma_k) = (\alpha_k', n\pi, 0)$ or $(0, n\pi, (-1)^n\alpha_k')$;

$\widetilde{\mathbb{M}} \leftarrow \widetilde{\mathbb{M}} \cup \{(\alpha_k, \gamma_k)\}$;

end

case $\{(q,t_k),(q,t_{k+1})\} \in \mathbb{M}^{(q)}$ **do**

$\widetilde{\mathbb{M}} \leftarrow \widetilde{\mathbb{M}} \cup \{(\alpha_k, \gamma_k)\}$;

end

end

end

`/* Determine the` q`-embeddability of` $W_{t_j;t_i}$ `with the following necessary and sufficient condition. For the proof, see Appendix D.9. */`

if $\gamma_k + \alpha_{k+1} = n\pi$, *for all* $k \in [i, j-1]$ *and* $(\alpha_k, \gamma_k) \in \widetilde{\mathbb{M}}$ **then**

$\mathbb{B}_2 \leftarrow \mathbb{B}_2 \cup \{(\mathcal{V}_{q;t_i,t_j}, \mathcal{B}_q(W_{t_j;t_i}))\}$;

$\mathbb{H}^{(q)} \leftarrow \mathbb{H}^{(q)} \cup \{\mathcal{V}_{q;t_i,t_j}\}$;

end

end

end

return $\{\mathbb{B}_2^{(q)}, \mathbb{H}^{(q)}\}$;

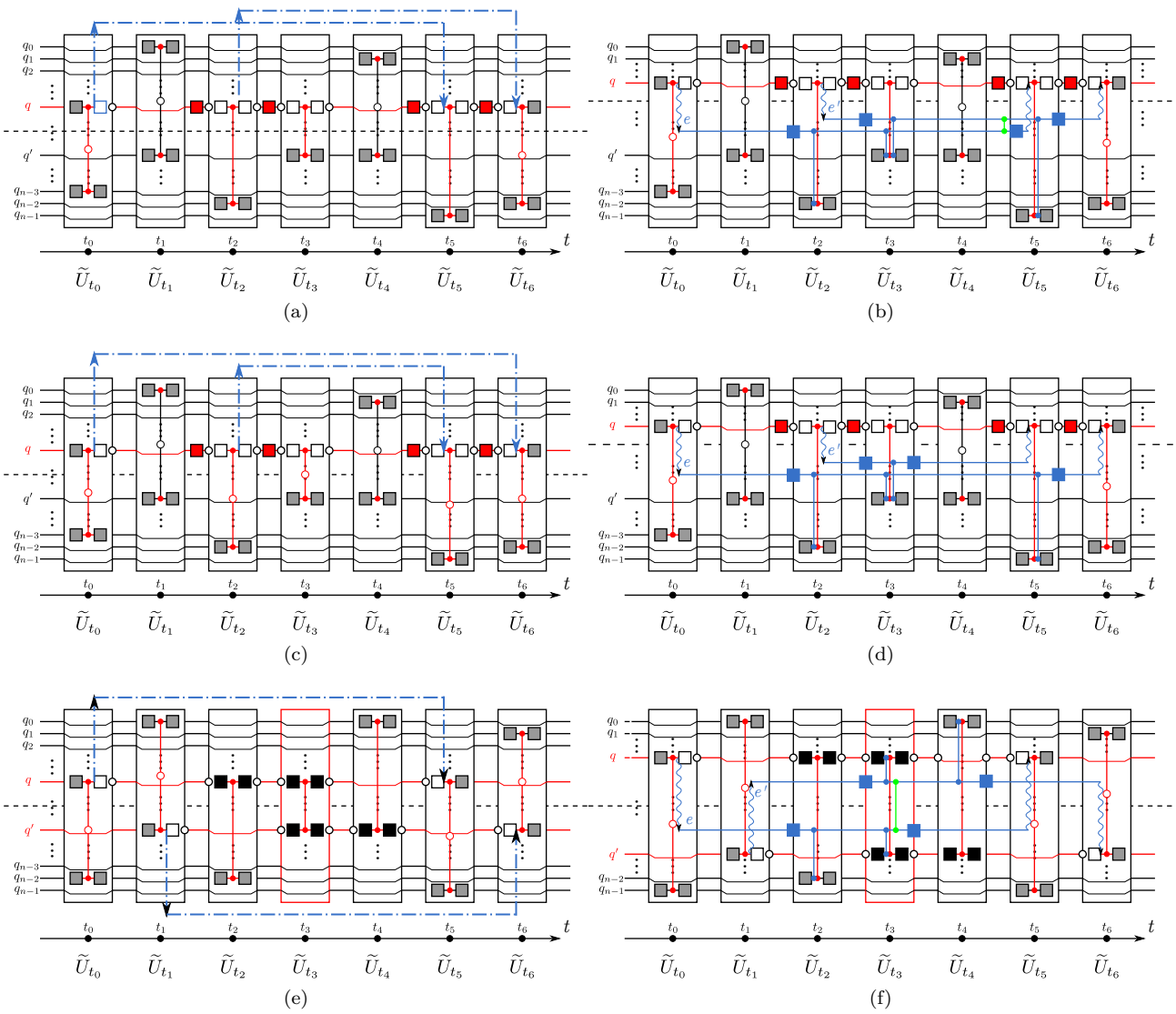


Figure 20: Nested embeddings in elemental blocks. (a) Nested embeddings on the same qubit q. (b) Implementation of the embedding in (a). (c) Inclusive nested embeddings on the same qubit q. (d) Implementation of the embedding in (c). (e) Nested embeddings on two qubits $\{q, q'\}$ that belong to different local systems. (f) Incompatibility of the two nested embedding due to the additional global gate $C_Z^{(e,e')}$.

Fig. 20 shows the three nested embeddings in a circuit consisting of single-qubit and two-qubit gates. The figures (b),(d),(f) are the implementation of the nested embeddings indicated by the blue dot-dashed lines in the figures (a),(c),(e), respectively. The nested embeddings in Fig. 20 (e) are not compatible, since their implementation introduces a global control-Z gate (green).

Theorem 36 implies that an indecomposable neighbouring embedding is always compatible with any other indecomposable embeddings. The incompatibility of indecomposable embeddings exists only for the indecomposable hopping embeddings, which contain a global control-Z gate. Such incompatibility is intrinsic and independent from external resources. After the identification of the intrinsic incompatibility between two indecomposable hopping embeddings, one can represent the incompatibility associated with a global control-Z gate by a conflict edge at the level of a set of distributable packets $\mathbb{V}$.

Definition 37 (Intrinsic conflict edges in $\mathbb{V}$)
In a set of distributable packets $\mathbb{V}$, a global control-Z gate $C_Z^{(q,q';t)}$ defines a conflict edge

$$\{\mathcal{V}_{q,T}, \mathcal{V}_{q',T'}\}, \tag{102}$$

where $\mathcal{V}_{q,T}, \mathcal{V}_{q',T'} \in \mathbb{V}$ are the distributable packets that cover the control-Z gate, namely $\min T < t < \max T$, $\min T' < t < \max T'$. The set of intrinsic conflict edges in $\mathbb{V}$ is denoted by

$$\mathbb{C}_{in}(\mathbb{V}) := \{\{\mathcal{V}_{q,T}, \mathcal{V}_{q',T'}\} : \mathcal{V}_{q,T}, \mathcal{V}_{q',T'} \in \mathbb{V}, \exists C_Z^{(q,q';t)} \text{ covered by } \mathcal{V}_{q,T}, \mathcal{V}_{q',T'}.\} \tag{103}$$

Since each indecomposable embedding in $\mathbb{B}_2$ corresponds to a unique distributable packet in $\mathbb{H}$, the intrinsic conflict edges between indecomposable embeddings introduced in Eq. (67) are equivalent to the intrinsic conflict edges between the distributable packets in $\mathbb{H}$. One can therefore construct the intrinsic packet-conflict and kernel-conflict graphs as

$$C_{in}(\mathbb{V}_{\mathbb{B}}) = (\mathbb{V}_{\mathbb{B}}, \mathbb{C}_{in}(\mathbb{V}_{\mathbb{B}})) \tag{104}$$

and

$$C_{in}(\mathbb{H}) = (\mathbb{H}, \mathbb{C}_{in}(\mathbb{H})), \tag{105}$$

respectively.

For the extrinsic kernel-conflict graphs, one will need to include the trivial distributable packets $\mathbb{V}_0$ in the vertices, $\mathbb{K} = \mathbb{V}_0 \cup \mathbb{H}$. The extrinsic kernel-conflict graph $C_{ex}(\mathbb{K})$ will be then constructed at the level of $\mathbb{K}$, from which one can construct the extrinsic packet-conflict graph $C_{ex}(\mathbb{V}_{\mathbb{B}})$ according to Eq. (66).

5 Applications of embedding-enhanced distributed quantum computing

In this section, we demonstrate our packing algorithms for DQC of unitary coupled-cluster (UCC) circuit [45, 46], with which one implements a quantum variational eigensolver to find the eigenvalue of an observable that simulates a chemical system. The implementation of our algorithm can be found in an open-source package pytket-dqc [1], which is developed for multipartite DQC based on our embedding-enhanced method [39].

5.1 DQC of a 4-qubit UCC circuit

A 4-qubit UCC circuit after the control-phase conversion is shown in Fig. 24 in Appendix C. It contains 64 global control-Z gates. The qubits are partitioned in two local systems $A = \{q_0, q_1\}$ and $B = \{q_2, q_3\}$. Employing Algorithm 33, 34 and 35, one obtains a packing graph $G_{\mathbb{B}}$ in Fig. 21 (a). The colored vertices are the $\mathbb{B}$-distributable packets, while the white vertices stretching from the colored vertices represent the hopping embeddings that merge the neighbouring packets to form a colored vertex. According to Algorithm 27, one first determines a minimum vertex cover $\mathbb{V}_{\text{mvc}}$ of the packing graph $G_{\mathbb{B}}$, which are highlighted as red vertices. Note that in this example, we only heuristically find a minimum vertex cover instead of searching for all minimum vertex covers.

According to Section 4.2, one can obtain the corresponding packet-conflict graph in Fig. 21 (b). The conflict edges are defined at the level of distributable packets. One has to solve the conflicts (red edges) induced by the minimum vertex cover $\mathbb{V}_{\text{mvc}}$. To this end, we need to expand the packet-conflict graph to a kernel-conflict graph, as it is shown in Fig. 21 (c). According to Algorithm 27, we select the set of kernels $\mathbb{K}_{\text{mvc}}$ that are induced by the minimum vertex cover $\mathbb{V}_{\text{mvc}}$, and highlight them in red. The $\mathbb{K}_{\text{mvc}}$-induced kernel-conflict graph $C_{in}(\mathbb{K}_{\text{mvc}})$ representing the conflicts that need to be resolved. We then find the minimum vertex cover $\widetilde{\mathbb{K}}_{\text{mvc}}$ of the conflict graph $C_{in}(\mathbb{K}_{\text{mvc}})$, and highlight them in blue. To solve the conflicts, we remove the blue-highlighted hopping embeddings $\widetilde{\mathbb{K}}_{\text{mvc}}$ from the minimum-vertex-covering packets $\mathbb{V}_{\text{mvc}}$.

In total, we find 10 distributable packets in $\mathbb{V}_{\text{mvc}}$ that cover all the packing edges and remove 7 hopping embedding kernels $\widetilde{\mathbb{K}}_{\text{mvc}}$ from the selected packets. In the end, we need only 17 distributing processes to implement a DQC of the 4-qubit UCC circuit, which contains 64 global control-Z gates. Our method therefore saves 47 ebits in total.

5.2 Comparison with other protocols

We use the heuristic implementation of the packing algorithm (Algorithm 27) to analyze the bipartite distribution of UCC circuits compared with the G^*-simple and G^*-LP algorithms based on the "Migration Selection" method introduced in [35]. In the "Migration Selection" method, neighboring nonlocal CZ gates can also be packed together, but its packing stops when the process encounters single-qubit gate or local CZ gates, no matter whether embeddable or not. In our embedding-enhanced packing algorithm, one packs control-phase gates and uses embedding to merge non-sequential distributing processes over embeddable single-qubit and two-qubit gates. The G^*-algorithm can therefore be understood as a special packing algorithm for CZ-compiled circuits without embedding.

Our benchmarking is performed on the UCC circuits with an even number of qubits that are uniformly distributed over two QPUs, each of which has the same specifications. To account for the entanglement cost that depends on qubit allocation, we benchmark all possible bipartitions with an equal number of local qubits for 4-qubit and 6-qubit circuits. The result is shown in Fig. 22 (a). It shows that embedding can significantly reduce entanglement costs under every possible bipartition. The entanglement efficiencies of these three protocols are further compared for UCC circuits up to a qubit number of 20 in Fig. 22 (b). Since the number of bipartitions increases exponentially with respect to the qubit number, in this comparison, we fixed a bipartition for each qubit number. The result shows that the entanglement efficiency of embedding-enhanced distributing is also significantly improved for circuits with large qubit numbers.

[1] https://github.com/CQCL/pytket-dqc

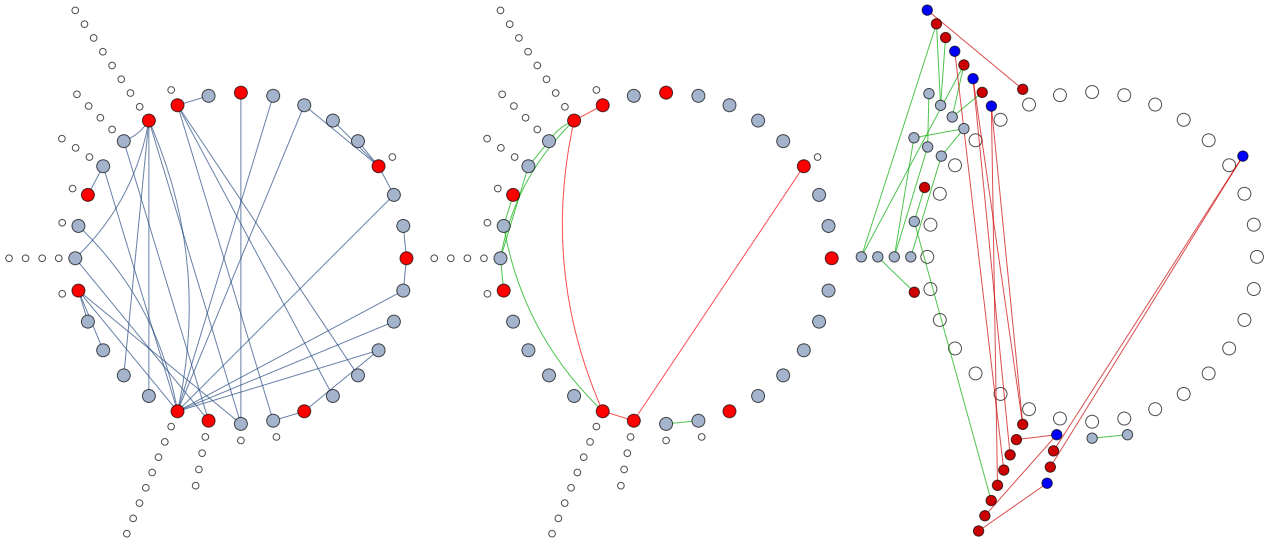


Figure 21: The distributability and embeddability structure of a 4-qubit UCC circuit. (a) The packing graph: Each colored vertex is a distributable packet, which is formed by the merging of the neighbouring packets through indecomposable hopping packets. The indecomposable hopping packets are depicted as white vertices and aligned in a line stretching from each colored vertex. The edges are packing edges. The red vertices are a minimum vertex cover $\mathbb{V}_{\text{mvc}}$ of the packing graph. The order of $\mathbb{V}_{\text{mvc}}$ is 10. (b) Intrinsic packet-conflict graph: the conflict graph defined at the level of distributable packets has the same vertex set as the packing graph. The conflict edges between packets are induced from the intrinsic conflict edges between kernels according to Eq. (66). The red edges are the conflict edges induced by the minimum vertex cover $\mathbb{V}_{\text{mvc}}$. (c) Intrinsic kernel-conflict graph defined at the level of kernels of packing processes. The colored vertices are the indecomposable hopping packets representing the kernels of hopping embeddings, while the white circles are the distributable packet formed from these embedding kernels. The red vertices $\mathbb{K}_{\text{mvc}}$ are the indecomposable hopping packets that included in the minimum vertex cover $\mathbb{V}_{\text{mvc}}$. The red edges are the conflict edges induced by $\mathbb{K}_{\text{mvc}}$. The blue vertices $\widetilde{\mathbb{K}}_{\text{mvc}}$ are a minimum vertex cover of the $\mathbb{K}_{\text{mvc}}$-induced conflict graph, which are the embeddings that needs to be removed. The order of $\widetilde{\mathbb{K}}_{\text{mvc}}$ is 7.

5.3 Embedding-enhanced multipartite-distributed quantum computing

The extension of our embedding-enhanced packing protocol to multipartite DQC requires an extended definition of distributable packets (Definition 16) through an extension of the bipartite embedding rules to multipartite systems. Once we get the extended packing graphs and conflict graphs constructed by the extended distributable packets as their vertices, one can follow Algorithms 27 and 28 to compile the DQC. However, the identification of packing edges and conflict edges is challenging. For multipartite embeddings, for instant in an $A|B|C$ system, their compatibility has to be considered under all possible bipartite subsystems, since the additional local correction gates in an embedding process of a nonlocal gate between two parties $A|B$ may destroy its embeddability between the other party $A|C$. The incompatibility of embeddings is therefore not more describable by a bipartite conflict edge.

A straightforward but not optimal solution is to extend the notion of distributable packets to a set of gate nodes that are associated with two-qubit gates acting on the same local systems, such that each distributable packet is associated with only two parties. With this solution, one can circumvent the multipartite conflict problem of embeddings and introduce the embedding method to existing compilers for multipartite DQC [34] to allow entanglement-efficient DQC on multipartite quantum computing networks [39].

6 Conclusion

In this paper, we have developed a theoretical architecture for entanglement-efficient distributed quantum computing over two local QPUs with the assistance of entanglement. The building blocks of the DQC architecture are the *entanglement-assisted packing processes* (Definition 2), which are extended from the EJPP protocol [26]. An entanglement-assisted packing process packs a sequence of gates as its kernel sandwiched by the entanglement-assisted *starting and ending processes* (Definition 1). Two types of entanglement-assisted packing processes are essential, namely the *distributing processes* (Definition 4) and the *embedding processes* (Definition 7). The ultimate goal of distributed quantum computing with entanglement-assisted packing processes is to implement a quantum circuit with distributing processes, in which the kernels are all local gates. An embed-

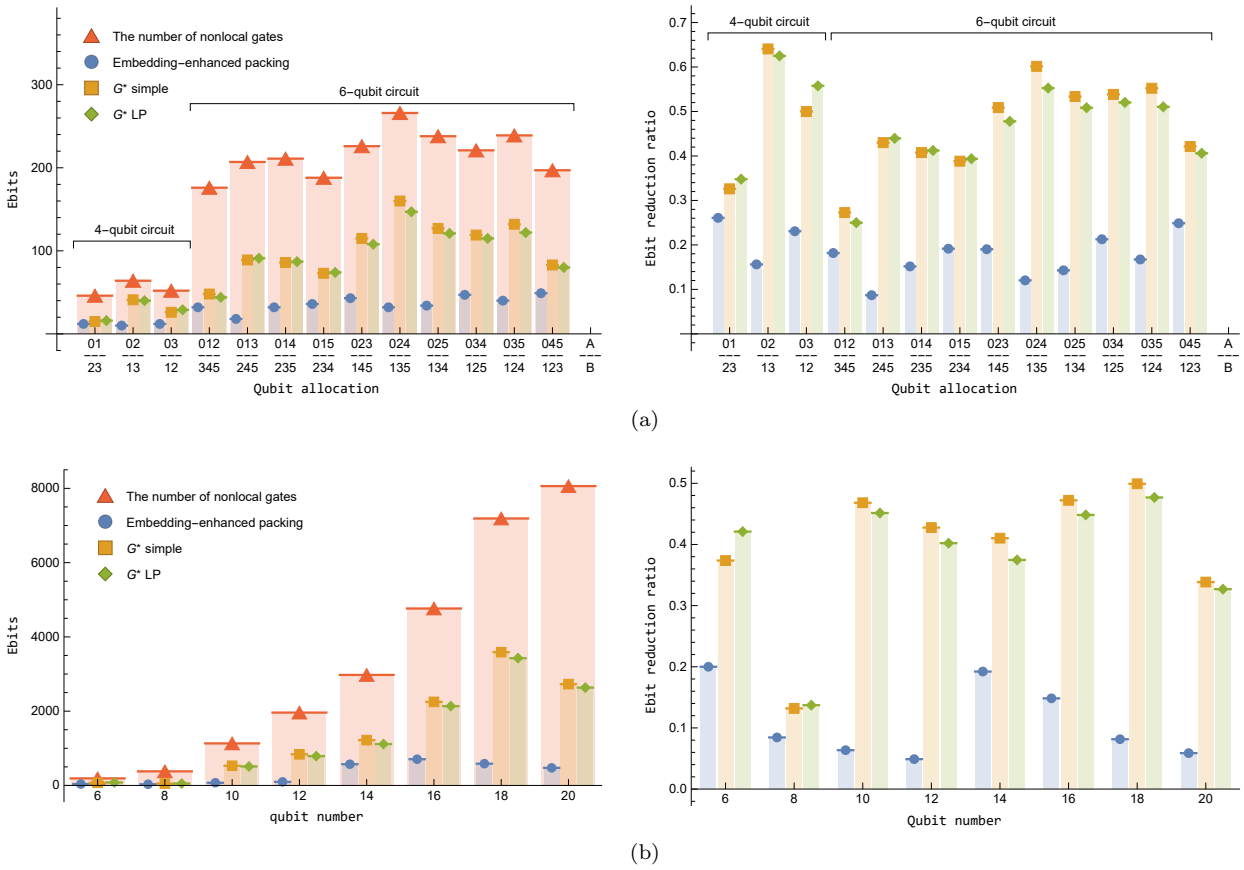


Figure 22: The entanglement costs of different DQC protocols for UCC circuits (left column) and their reduction ratios (right column). (a) The entanglement cost for 4-qubit and 6-qubit UCC circuits under different bipartitions. (b) The entanglement cost for UCC circuits with a qubit number from 6 up to 20 under a fixed bipartition.

ding process allows the packing of two non-sequential distributing processes and hence save one ebit of entanglement. We therefore explore the embeddability of quantum circuits to save the entanglement cost in DQC.

The distributability of a unitary can be determined by Theorem 5. Each distributing process consumes an ebit of entanglement. To save entanglement, we introduced the embedding processes to merge two non-sequential distributing processes into one single packing process. The embeddability of a unitary between two distributing processes can be determined with a sufficient condition, from which one can derive the primitive embedding rules (Corollary 9, Theorem 10, and Corollary 11). These embedding rules join the nodes of distributable gates into *distributable packets* (Definition 16 and Lemma 17), on each of which one can implement the gates locally assisted with one-ebit entanglement in a single distributing process (Theorem 18).

The *trivial distributing processes* and *indecomposable embedding processes* (Definition 19) associated with different packets may be incompatible for simultaneous implementation. We therefore introduce *packing edges* between distributable packets (Definition 23) and conflict edges incident to embedding processes (Definition 20 and 25) to represent the incompatibility among distributing and embedding processes. With these edges, one can construct the *packing graphs* and *conflict graphs* of a circuit, which contain the full information of distributability, embeddability and incompatability in a circuit. Based on these graphs, one can determine the packing of a circuit with heuristic Algorithm 27 (or Algorithm 28) for the scenario of unlimited (or limited) external resources.

In particular, we consider circuits consisting of only one-qubit and two-qubit gates, which are universal for quantum computing. The distributability structure (packing edges) of such a circuit is completely represented by the control-phase gates (Lemma 29) in its control-phase conversion. In a control-phase conversion, a circuit is decomposed as a sequence of two-qubit building blocks. We have derived the primitive embedding rules for these building blocks in Corollary 30. The embedding rules for single-qubit gates (Theorem 10) and two-qubit blocks (Corollary 30) identify the indecomposable neighbouring embeddings (Algorithm 34) and hopping embeddings (Lemma 31 and Algorithm 35), respectively. Based on these embedding rules, one can then identify the set of distributable packets of a circuit with Algorithm 33. Fi-

nally, one obtains the ultimate packing graph through the association of packing edges to global control-phase gates, and the ultimate conflict graphs through the association of conflict edges to incompatible embeddings (Theorem 36).

These algorithms are demonstrated in the distributed implementation of a 4-qubit UCC circuit, which contains 64 global control-Z gates. The ultimate number of packing processes in the distributed implementation is 17, hence requiring 17 ebits of entanglement and saving 47 ebits. We benchmark the algorithms with further instants of the UCC circuits up to 20 qubits. The comparison with other protocols shows a significant enhancement of entanglement efficiency by embeddings.

The packing method in this paper can be summarized as "distributing enhanced by embedding". It incorporates the extrinsic limit of quantum resources, such as the number of available local memory qubits, in the conflict edges. It can be therefore employed to study the scalability of distributed quantum computing. One can further extend our method to multipartite scenarios and adopt quantum network topology to establish entanglement-efficient DQC over quantum internet [39].

The DQC architecture of a quantum circuit based on entanglement-assisted packing processes revealed by the packing algorithms is a constructive method to determine an upper bound on the entanglement cost of the entanglement-assisted LOCCs of a circuit. Benefit from embeddings, one can obtain a tighter upper bound on the entanglement cost approaching the lower bound, which is determined by the operator Schmidt rank [38]. One can therefore exploit the packing method to explore the unitaries that have the optimal entanglement cost implemented with entanglement-assisted packing processes. The packing protocol in this paper is therefore practical for finding an entanglement-efficient solution for distributed quantum computing, and also useful for the fundamental study of the optimality in entanglement-assisted LOCC.

Acknowledgments

JYW is supported by National Science and Technology Council, Taiwan, under Grant no. NSTC 110-2112-M-032-005-MY3, 111-2923-M-032-002-MY5, 111-2119-M-008-002, and 112-2112-M-032-008-MY3. KM, TF, AS, and MM are supported by MEXT Quantum Leap Flag-ship Program (MEXT QLEAP) JPMXS0118069605, JPMXS0120351339, Japan Society for the Promotion of Science (JSPS) KAKENHI Grant No. 18K13467, 21H03394.

A The representation of an entanglement-assisted packing process

With a general kernel K, the quantum operations of a perfect packing process is described by

$$\rho = \mathrm{tr}_e \left(C_{q,X_e} K C_{q,X_e} \left|\psi, 0_e\right\rangle \left\langle \psi, 0_e\right| C_{q,X_e} K^\dagger C_{q,X_e} \right). \tag{106}$$

A packing process is therefore a set of Kraus operators

$$\mathcal{P}_{q,e}[K](|\psi\rangle) = \frac{1}{2}\mathcal{K}_+ \left|\psi\right\rangle \left\langle\psi\right| \mathcal{K}_+^\dagger + \frac{1}{2}\mathcal{K}_- \left|\psi\right\rangle \left\langle\psi\right| \mathcal{K}_-^\dagger, \tag{107}$$

where

$$\mathcal{K}_\pm = \sqrt{2}\left\langle \pm_e | C_{q,X_e} K C_{q,X_e} | 0_e \right\rangle. \tag{108}$$

For some special kernels, the Kraus operators $\mathcal{K}_\pm$ are equivalent up to a global phase,

$$\mathcal{K}_+ = e^{\mathbb{i}\,\varphi}\mathcal{K}_-. \tag{109}$$

In such a case, a packing process is equivalent to a unitary.

Lemma 38 (U-equivalent packing process) *Let $\mathcal{P}_{q,e}[K]$ be an EJPP process. It is equivalent to a unitary U,*

$$U = \mathcal{P}_{q,e}[K], \tag{110}$$

if and only if there exists a single-qubit X-rotation on the auxiliary qubit e,

$$R_X^{(e)}(\varphi) := \left|+_e\right\rangle \left\langle +_e\right| + e^{\mathbb{i}\,\varphi} \left|-_e\right\rangle \left\langle -_e\right|, \tag{111}$$

such that $R_X^{(e)}(\varphi)K$ admits the following form in the computational basis $\{|i_q i'_e\rangle\}_{i,i'}$ of q and e.

$$\begin{aligned} R_X^{(e)}(\varphi)K &= \left\{ \left\langle i_q i'_e \right| R_X^{(e)}(\varphi) K \left| j_q j'_e \right\rangle \right\}_{ii',jj'} \\ &= \begin{pmatrix} U_{00}^{(\bar{q})} & 0 & 0 & U_{01}^{(\bar{q})} \\ 0 & * & * & 0 \\ 0 & * & * & 0 \\ U_{10}^{(\bar{q})} & 0 & 0 & U_{11}^{(\bar{q})} \end{pmatrix}, \end{aligned} \tag{112}$$

where $U_{ij}^{(\bar{q})} := \langle i_q | U | j_q \rangle$. The phase φ is called the auxiliary phase.

Proof: A packing process is U-equivalent, if and only if the Kraus operator $\mathcal{K}_\pm$ in Eq. (108) fulfills Eq. (109). Introducing a local phase calibration $R_X^{(e)}(\varphi)$ on the auxiliary qubit e to eliminate the phase φ in Eq. (109)

$$\widetilde{K} := R_X^{(e)}(\varphi)K, \tag{113}$$

one obtains

$$\left\langle +_e \right| \widetilde{K}\, C_{q,X_e} \left| 0_e \right\rangle = \left\langle -_e \right| Z_q\, \widetilde{K}\, C_{q,X_e} \left| 0_e \right\rangle. \tag{114}$$

This equality is equivalent to

$$0 = \left\langle +_e \right| \left(\mathbb{1}_{qe} - Z_q Z_e \right) \widetilde{K} \left(\mathbb{1}_{qe} + Z_q Z_e \right) \left| +_e \right\rangle. \tag{115}$$

In the computational basis of q and e, this equality is equivalent to

$$0 = \langle i_q, (i \oplus 1)_e | \widetilde{K} | i_q, i_e \rangle \quad \text{for } i = 0, 1. \tag{116}$$

This proves that a packing process is equivalent to a unitary if and only if the $R_X^{(e)}(\varphi)$-calibrated kernel $\widetilde{K}$ has the following form

$$\widetilde{K} = \begin{pmatrix} \widetilde{K}^{(\bar{q},\bar{e})}_{00,00} & * & * & \widetilde{K}^{(\bar{q},\bar{e})}_{00,11} \\ 0 & * & * & 0 \\ 0 & * & * & 0 \\ \widetilde{K}^{(\bar{q},\bar{e})}_{11,00} & * & * & \widetilde{K}^{(\bar{q},\bar{e})}_{11,11} \end{pmatrix}, \tag{117}$$

where $\widetilde{K}^{(\bar{q},\bar{e})}_{ii',jj'} := \langle i_q, i'_e | \widetilde{K} | j_q, j'_e \rangle$. Since the $R_X^{(e)}(\varphi)$ calibration only introduce a global phase to the Kraus operator $\mathcal{K}_-$ of a packing process, $\mathcal{P}_{q,e}[K]$ and $\mathcal{P}_{q,e}(\widetilde{K})$ implement the same unitary U. The unitary U implemented by $\mathcal{P}_{q,e}[K]$ is therefore given by

$$U = \langle +_e | \widetilde{K} (\mathbb{1} + Z_q Z_e) | +_e \rangle . \tag{118}$$

As a result of Eq. (116), the representation of U in the computational basis of q is therefore

$$\begin{aligned} \langle i_q | U | j_q \rangle &= \langle i_q, +_e | \widetilde{K} (\mathbb{1} + Z_q Z_e) | j_q, +_e \rangle \\ &= \langle i_q, i_e | \widetilde{K} | j_q, j_e \rangle , \end{aligned} \tag{119}$$

which leads to

$$\widetilde{K} = \begin{pmatrix} \widetilde{U}^{(\bar{q})}_{00} & \widetilde{K}^{(\bar{q},\bar{e})}_{00,01} & \widetilde{K}^{(\bar{q},\bar{e})}_{00,10} & \widetilde{U}^{(\bar{q})}_{01} \\ 0 & \widetilde{K}^{(\bar{q},\bar{e})}_{01,01} & \widetilde{K}^{(\bar{q},\bar{e})}_{01,10} & 0 \\ 0 & \widetilde{K}^{(\bar{q},\bar{e})}_{10,01} & \widetilde{K}^{(\bar{q},\bar{e})}_{10,10} & 0 \\ \widetilde{U}^{(\bar{q})}_{10} & \widetilde{K}^{(\bar{q},\bar{e})}_{11,01} & \widetilde{K}^{(\bar{q},\bar{e})}_{11,10} & \widetilde{U}^{(\bar{q})}_{11} \end{pmatrix}. \tag{120}$$

Let W and V be two matrices defined as

$$\begin{aligned} W &= \begin{pmatrix} \widetilde{K}^{(\bar{q},\bar{e})}_{00,01} & \widetilde{K}^{(\bar{q},\bar{e})}_{00,10} \\ \widetilde{K}^{(\bar{q},\bar{e})}_{11,01} & \widetilde{K}^{(\bar{q},\bar{e})}_{11,10} \end{pmatrix}, \\ V &= \begin{pmatrix} \widetilde{K}^{(\bar{q},\bar{e})}_{01,01} & \widetilde{K}^{(\bar{q},\bar{e})}_{01,10} \\ \widetilde{K}^{(\bar{q},\bar{e})}_{10,01} & \widetilde{K}^{(\bar{q},\bar{e})}_{10,10} \end{pmatrix}. \end{aligned} \tag{121}$$

Since $\widetilde{K}$ and U are both unitary, it holds then

$$V^\dagger V = \mathbb{1}, \text{ and } W V^\dagger = 0, \tag{122}$$

which leads to

$$W = W V^\dagger V = 0. \tag{123}$$

As a result,

$$\widetilde{K} = \begin{pmatrix} \widetilde{U}^{(\bar{q})}_{00} & 0 & 0 & \widetilde{U}^{(\bar{q})}_{01} \\ 0 & * & * & 0 \\ 0 & * & * & 0 \\ \widetilde{U}^{(\bar{q})}_{10} & 0 & 0 & \widetilde{U}^{(\bar{q})}_{11} \end{pmatrix}. \tag{124}$$

This completes the proof. ■

Corollary 39 (Sufficient condition for U-equivalent) *An entanglement-assisted process with the kernel $C_{q,X_e} U C_{q,X_e}$ is canonical and equivalent to the unitary U,*

$$U = \mathcal{P}_{q,e}[C_{q,X_e} U C_{q,X_e}]. \tag{125}$$

The kernel $C_{q,X_e} U C_{q,X_e}$ is the primitive kernel *of U.*
Proof: According to Eq. (108), the Kraus operator of $\mathcal{P}_{q,e}[C_{q,X_e} U C_{q,X_e}]$ are $\mathcal{K}_\pm = \sqrt{2} \langle \pm_e | U | 0_e \rangle = U$. ■

B Extended embedding

The embedding introduced in Definition 7 can merge two non-sequential distributing processes without changing global gates in the kernel. The packing edges associated with global gates in a packing graph are therefore not affected by an embedding process. Although an embedding process may introduce conflicts to some other embeddings, one can solve these conflicts through the removal of embeddings, which split their original distributable packets. It means that the distributability structure of global gates represented by packing edges in the packing graph is not changed by ordinary embeddings. Since the global gates in a kernel are not changed, the entanglement cost for distributing processes in the kernel does not change. The merging of two non-sequential distributing processes will then save 1 ebit.

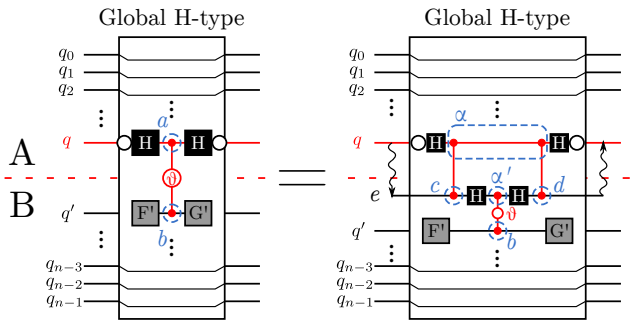


Figure 23: An extended embedding rule for general global H-type embeddig unit.

In addition to the ordinary H-type embedding rule in Corollary 30 for a two-qubit embedding unit with a control-Z gate, we introduce an extended embedding rule for the H-type embedding unit with general controlling phase $\theta \neq \pi$.

Lemma 40 (Extended embedding) *An H-type 2-qubit embedding unit can be packed with the following rule (Fig. 23)*

$$\begin{aligned} & H_q U^{(q,q')} H_q \\ =& \mathcal{P}_{q,e} \left[H_q C_Z^{(q,e)} H_e U^{(e,q')} H_e C_Z^{(q,e)} H_q \right]. \end{aligned} \tag{126}$$

Proof: Employing the primitive embedding rule in Corollary 9, we know that

$$\begin{aligned} &H_q U^{(q,q')} H_q \\ =&\mathcal{P}_{q,e}\left[C_{q',X_e} H_q U^{(q,q')} H_q C_{q',X_e}\right]. \end{aligned} \tag{127}$$

One can then show that

$$\begin{aligned} &C_{q',X_e} H_q U^{(q,q')} H_q C_{q',X_e} \\ =&H_q C_Z^{(q,e)} H_e U^{(e,q')} H_e C_Z^{(q,e)} H_q. \end{aligned} \tag{128}$$

■

As it is shown in Fig. 23, this extended embedding of the global H-type block localizes the control-phase gate to the B system with two additional global control-Z gates. On the left-hand side, the original distributable packets $\mathcal{V}_a$ and $\mathcal{V}_b$ (blue dashed circles) form a packing edge ($\mathcal{V}_a \leftrightarrow \mathcal{V}_b$). After the extended embedding, the packet $\mathcal{V}_a$ is replaced by $\mathcal{V}_\alpha$, and the packing edge ($\mathcal{V}_a \leftrightarrow \mathcal{V}_b$) is localized to ($\mathcal{V}_{\alpha'} \leftrightarrow b$) on the right-hand side with two additional packets $\mathcal{V}_c$ and $\mathcal{V}_d$ globally connected to $\mathcal{V}_\alpha$.

$$(\mathcal{V}_a \leftrightarrow \mathcal{V}_b) \underset{\text{embedding}}{\longrightarrow} (\mathcal{V}_c \leftrightarrow \mathcal{V}_\alpha \leftrightarrow \mathcal{V}_d, \mathcal{V}_{\alpha'} \leftrightarrow \mathcal{V}_b). \tag{129}$$

The global packing edges of the kernel are now changed to $\mathcal{V}_c \leftrightarrow \mathcal{V}_\alpha \leftrightarrow \mathcal{V}_d$. The corresponding graph has the minimum vertex cover $\{\mathcal{V}_a\}$. If $\mathcal{V}_a$ is already selected as one of the root packets for the distributing processes, then after the extend embedding, one can replace the original packet $\mathcal{V}_a$ in the original vertex cover by $\mathcal{V}_\alpha$. Such a replacement does not change the entanglement cost in the kernel. It therefore leads to a possible reduction of entanglement by one ebit through the merging of two non-sequential distributing processes hopping over the H-type unit.

However, if $\mathcal{V}_b$ is selected in the vertex cover of the original packing graph rather than $\mathcal{V}_a$, the 1-ebit entanglement saved by the extended embedding hopping over $\mathcal{V}_\alpha$ has to be consumed for the distributing process rooted at $\mathcal{V}_\alpha$. It means that the extended embedding does not bring any reduction of entanglement cost.

It is therefore necessary to select $\mathcal{V}_a$ as a root packet for the extended embedding to save one ebit. Suppose that $\mathcal{V}_a$ is a root packet. After the extended embedding, the new local control phase gate $C_V^{(\alpha',b)}$ may intrinsically prevent the other hopping embeddings. This happens when there is a selected root packet merged by the q'-rooted hopping embedding over the packet $\mathcal{V}_b$. In this case, one has to remove the q'-rooted hopping embedding, and split the corresponding packet, which will reuse the 1-ebit entanglement saved by the extended embedding. Such a conflict is the only intrinsic conflict that prevents the reduction of entanglement cost through the extended embedding. As a result, the necessary condition for reduction of entanglement cost by the extended embedding rule can be summarized as follows.

Corollary 41 (Entanglement reduction condition) *The q-rooted extended embedding of an H-type embedding unit can save 1-ebit, only if*

1. *the distributable packet on q in the unit is already selected as a root packet, and*
2. *the local gate $C_V^{(\alpha',b)}$ does not conflict with a selected hopping embedding.*

This necessary condition is also sufficient, if there is no extrinsic limits on external resources.

After solving the intrinsic conflicts and identifying the root packets $\mathbb{P}_{i,j}$ in Algorithm 27 (line 6) and Algorithm 28 (line 7), one can introduce an addon function "*extended_ embedding()*" to further reduce the entanglement cost by checking the two conditions in Corollary 41. The algorithm for the addon function is given in Algorithm 42. Note that for Algorithm 28 with extrinsic limits, after the extended embedding, one has to update the extrinsic conflict graph on line 9.

Algorithm 42: Extended embedding

1 **Function** `extended_embedding` *($\mathbb{R}$)*:
2 $\quad\widetilde{\mathbb{H}} \leftarrow \emptyset$;
3 $\quad\mathbb{H}_{\mathbb{R}} \leftarrow$ Get the hopping embeddings in $\mathbb{R}$;
$\quad$`/* Get the neighbouring distributable packets from` $\mathbb{R}$ `*/`
4 $\quad\mathbb{R}_D \leftarrow$ Get the neighbouring packets from $\mathbb{R}$ through removal of the hopping embeddings;
5 $\quad$**foreach** $\mathcal{V}_{q,T} \in \mathbb{R}_D$ **do**
6 $\quad\quad$**if** $\exists$ *sequential* $(\mathcal{V}_{q,T_1}, \mathcal{V}_{q,T}, \mathcal{V}_{q,T_2}) \subseteq \mathbb{R}_D$ *such that* $\nexists$ *control phase gates on q between T_1, T, and T_2* **then**
7 $\quad\quad\quad(t_1, t_2) \leftarrow (\max T_1, \min T_2)$ (Condition 1 in Corollary 41);
8 $\quad\quad\quad$**if**
1. $(q; t_1, t_2) \notin \mathbb{H}_{\mathbb{R}}$, and
2. $W_{q;t_2,t_1}$ is extended embeddable according to Lemma 40, and
3. $\nexists (q'; t_a, t_b) \in \mathbb{H}_{\mathbb{R}}$, such that $\exists a$ global control phase gate $C_V^{(q,q')}(\theta_t)$ at a depth $t \in (t_1, t_2)$ and $t \in (t_a, t_b)$ (Condition 2 in Corollary 41);
9 $\quad\quad\quad$**then**
10 $\quad\quad\quad\quad\widetilde{\mathbb{H}} \leftarrow \widetilde{\mathbb{H}} \cup \{q; (t_1, t_2)\}$;
11 $\quad$**end**
12 $\quad\widetilde{\mathbb{R}} \leftarrow$ merge $\widetilde{\mathbb{R}}$ by $\widetilde{\mathbb{H}}$ according to Lemma 17;
13 $\quad$**return** $\{\widetilde{\mathbb{R}}, \widetilde{\mathbb{H}}\}$

C UCC circuits

We employ the "pytket-dqc" package to generate a 4-qubit UCC circuit. The 4-qubit circuit is converted according to the control-phase conversion in Eq. (76) and (77). The converted circuit is shown in Fig. 24. There are 64 2-qubit blocks. Every block contains one global control-phase gate. After the control-phase conversion, one employs Algorithm 33, 34 and 34 to identify the distributable packets and the corresponding packing graph and conflict graphs, which are shown in Fig. 21.

D Proof of theorems and lemmas

D.1 Proof of Theorem 3

According to Lemma 38, the matrices of the kernels K_i in the computational basis are

$$K_i = \begin{pmatrix} U_{i,00}^{(\bar{q})} & 0 & 0 & U_{i,01}^{(\bar{q})} \\ 0 & * & * & 0 \\ 0 & * & * & 0 \\ U_{i,10}^{(\bar{q})} & 0 & 0 & U_{i,11}^{(\bar{q})} \end{pmatrix}, \tag{130}$$

where $U_{i,jk}^{(\bar{q})} := \langle j_q|U_i|k_q\rangle$. It holds then

$$K_2K_1 = \begin{pmatrix} \widetilde{U}_{00}^{(\bar{q})} & 0 & 0 & \widetilde{U}_{01}^{(\bar{q})} \\ 0 & * & * & 0 \\ 0 & * & * & 0 \\ \widetilde{U}_{10}^{(\bar{q})} & 0 & 0 & \widetilde{U}_{11}^{(\bar{q})} \end{pmatrix}, \tag{131}$$

where

$$\widetilde{U}_{ij}^{(\bar{q})} = \sum_k U_{2,ik}^{(\bar{q})} U_{1,kj}^{(\bar{q})}. \tag{132}$$

As a result of Lemma 38, one obtains

$$U_2U_1 = \mathcal{P}_{q,e}[K_2K_1]. \tag{133}$$

This completes the proof.

D.2 Proof of Theorem 5

Let U be a q-rooted distributable unitary with q being on the local system A. According to Lemma 38, there exists a canonical kernel $K_A \otimes K_B$ such that

$$K_A \otimes K_B = \begin{pmatrix} \widetilde{U}_{00}^{(\bar{q})} & 0 & 0 & \widetilde{U}_{01}^{(\bar{q})} \\ 0 & * & * & 0 \\ 0 & * & * & 0 \\ \widetilde{U}_{10}^{(\bar{q})} & 0 & 0 & \widetilde{U}_{11}^{(\bar{q})} \end{pmatrix}. \tag{134}$$

Since $\langle i,j|K_A \otimes K_B|k,l\rangle = \langle i|K_A|k\rangle \otimes \langle j|K_B|l\rangle$, it holds then

$$0 = \langle i|K_A|k\rangle \otimes \langle j|K_B|l\rangle \text{, for all } i \oplus k \neq j \oplus l. \tag{135}$$

It means that K_A and K_B must be diagonal or anti-diagonal at the same time,

$$K_A = \sum_{i,j\in\{0,1\}} \Delta_{ij} |i\rangle \langle j|_q \otimes V_j^{(Q_A\setminus q)}, \tag{136}$$

$$K_B = \sum_{i,j\in\{0,1\}} \Delta_{ij} |i\rangle \langle j|_e \otimes W_j^{(Q_B)}. \tag{137}$$

where $\Delta_{ij} = \delta_j^i$ or $\Delta_{ij} = \delta_j^{i\oplus 1}$. As a result,

$$K_A \otimes K_B = \begin{pmatrix} \Delta_{00}^2 V_0 \otimes W_0 & 0 & 0 & \Delta_{01}^2 V_1 \otimes W_1 \\ 0 & \Delta_{00}\Delta_{11} V_0 \otimes W_1 & \Delta_{11}\Delta_{00} V_1 \otimes W_0 & 0 \\ 0 & \Delta_{11}\Delta_{00} V_1 \otimes W_0 & \Delta_{00}\Delta_{11} V_0 \otimes W_1 & 0 \\ \Delta_{10}^2 V_0 \otimes W_0 & 0 & 0 & \Delta_{11}^2 V_1 \otimes W_1 \end{pmatrix}. \tag{138}$$

It means that the following equality is a necessary condition for a q-rooted distributable unitary,

$$U = \sum_{i,j} \Delta_{ij} |i\rangle \langle j|_q \otimes V_j^{(Q_A\setminus q)} \otimes W_j^{(Q_B)}. \tag{139}$$

This equality is also sufficient for a q-rooted distributable unitary, which can be implemented with the kernels given in Eq. (136).

D.3 Proof of Lemma 13

Since U is embeddable on $\mathbb{Q}$ and $\mathbb{Q}'$ with the embedding rules $\mathcal{B}_{\mathbb{Q}}$ and $\mathcal{B}_{\mathbb{Q}'}$, it holds

$$U = \mathcal{P}_{\mathbb{Q}',\mathbb{A}'}\left[\mathcal{B}_{\mathbb{Q}'}\left(\mathcal{P}_{\mathbb{Q},\mathbb{A}}\left[\mathcal{B}_{\mathbb{Q}(U)}\right]\right)\right]. \tag{140}$$

According to Eq. (108), the unitary is equivalent to

$$\begin{aligned} U =& \sqrt{2}\,\langle \pm_{e_1}, \pm_{e_2}, ..., \pm_{e_k}|\, C_{\mathbb{Q}',X_{\mathbb{A}'}}(K'_A \otimes K'_B) \\ & \times C_{\mathbb{Q},X_{\mathbb{A}}} \mathcal{B}_{\mathbb{Q}'}(U) C_{\mathbb{Q},X_{\mathbb{A}}} \\ & \times (K_A \otimes K_B) C_{\mathbb{Q}',X_{\mathbb{A}'}} |0_{e_1}, 0_{e_2}, ..., 0_{e_k}\rangle, \end{aligned} \tag{141}$$

where $\mathbb{A} \cup \mathbb{A}' = \{e_1, ..., e_k\}$ is the disjoint set union of the auxiliary qubits $\mathbb{A}$ and $\mathbb{A}'$. As a result of the commutation relation in Eq. (41), the unitary U is then joint embeddable on $\mathbb{Q} \uplus \mathbb{Q}'$ with

$$\begin{aligned} U =& \sqrt{2}\,\langle \pm_{e_1}, \pm_{e_2}, ..., \pm_{e_k}|\, C_{\mathbb{Q}',X_{\mathbb{A}'}} C_{\mathbb{Q},X_{\mathbb{A}}} \\ & \times (K'_A \otimes K'_B) \mathcal{B}_{\mathbb{Q}}(U) (K_A \otimes K_B) \\ & \times C_{\mathbb{Q},X_{\mathbb{A}}} C_{\mathbb{Q}',X_{\mathbb{A}'}} |0_{e_1}, 0_{e_2}, ..., 0_{e_k}\rangle \\ =& \mathcal{P}_{\mathbb{Q}\uplus\mathbb{Q}',\mathbb{A}\uplus\mathbb{A}'}\left[\mathcal{B}_{\mathbb{Q}'} \circ \mathcal{B}_{\mathbb{Q}}(U)\right]. \end{aligned} \tag{142}$$

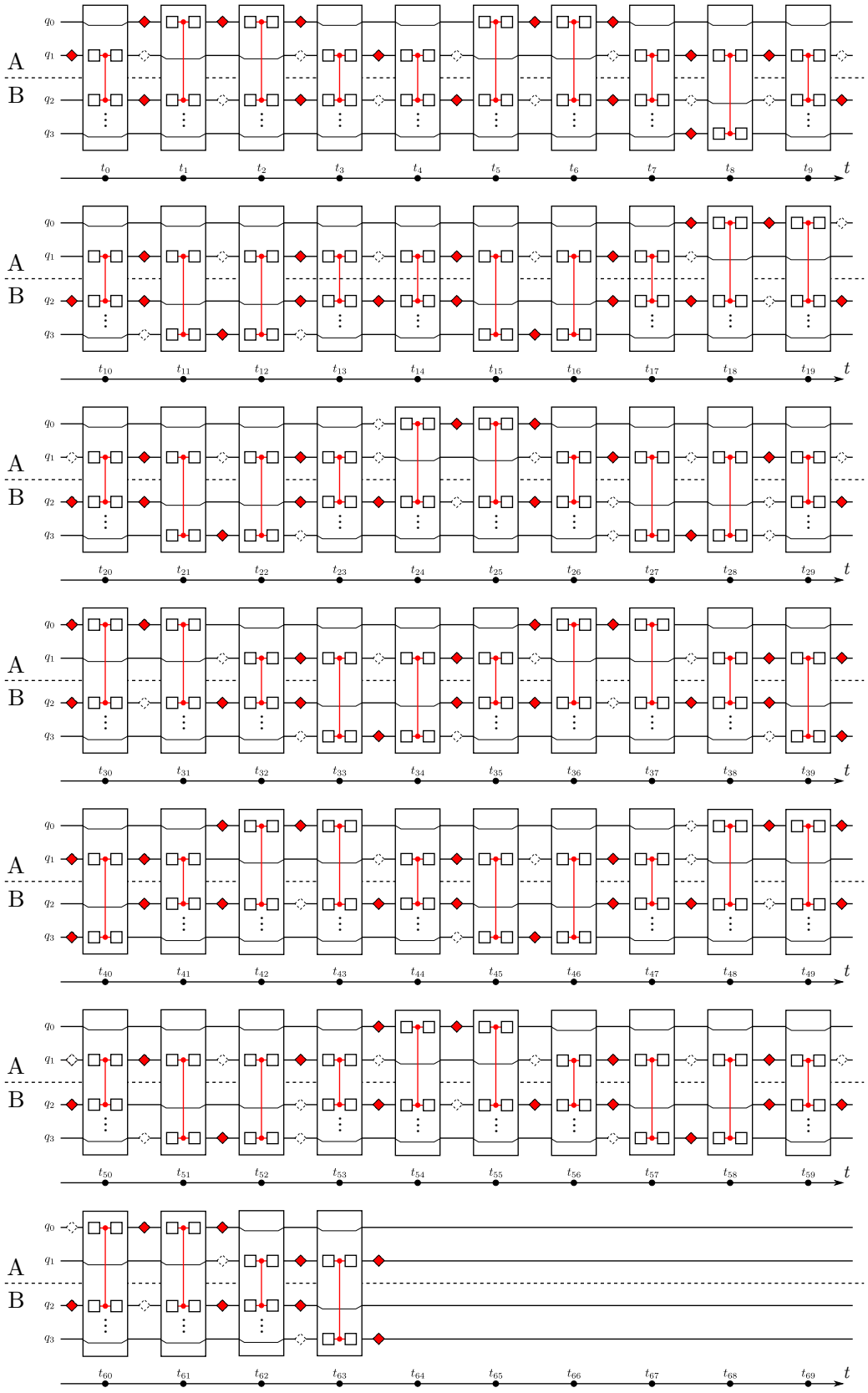


Figure 24: The 2-qubit blocks of a 4-qubit UCC circuit after the control-phase conversion.

D.4 Proof of Lemma 17

First of all, the unitaries U_t with $t \in T_1 \cup T_2$ are all distributable by definition.

Second, for any $t_{1,2} \in T_1 \cup T_2$ with $t_1 < t_2$, we need to prove that the unitary

$$W_{t_2;t_1} := \prod_{t:t_1<t<t_2} U_t \tag{143}$$

between t_1 and t_2 is q-rooted embeddable. If $(q, t_{1,2}) \in \mathcal{V}_{q,T_i}$ belong to the same distributable packet, $W_{t_1;t_2}$ is q-rooted embeddable by definition. For $t_1 \in T_1$ and $t_2 \in T_1$, let $t_0 \in T_1 \cap T_2$ be a common node in both $\mathcal{V}_{q,T_1}$ and $\mathcal{V}_{q,T_2}$. There are three possible position for t_0, namely, $t_0 < t_1 < t_2$, $t_1 < t_0 < t_2$, and $t_1 < t_2 < t_0$. For $t_0 < t_1 < t_2$, the unitary $W_{t_1;t_0}$ and $W_{t_2;t_0}$ are both embeddable, since $\{(q, t_0), (q, t_i)\} \subseteq \mathcal{V}_{q,T_i}$. It holds then

$$W_{t_2,t_1} = W_{t_2,t_0} W^\dagger_{t_1,t_0} U^\dagger_{t_1}, \tag{144}$$

which is also q-rooted embeddable. Analogously, one can prove the q-rooted embeddability of $W_{t_2;t_1}$ for $t_1 < t_0 < t_2$ and $t_1 < t_2 < t_0$. This completes the proof.

D.5 The time complexity of Algorithm 27 and 28

The time complexity of Algorithm 27 is estimated as follows. Given a packing graph $G_{\mathbb{B}} = (\mathbb{V}, \mathbb{E})$, there are three main steps in the algorithm,

1. enumerate all minimum vertex covers $\mathbb{K}_i$.
2. enumerate all minimum vertex covers $\{\mathbb{R}_{i,j}\}_j$ of the intrinsice kernel-conflict graph $C_{in}(\mathbb{K}_i)$.
3. determine the chromatic number of extrinsic packet-conflict graphs $C_{ex}(\mathbb{R}^{(A,B)}_{i,j})$

The determination of minimum vertex covers is equivalent to the determination of maximum matching according to the Köning theorem [52]. The complexity of enumerating all minimum vertex covers of a graph can be therefore estimated as $O(|\mathbb{V}|^{1/2}|\mathbb{E}| + |\mathbb{V}|N_{\mathrm{MVC}})$ according to Theorem 2 in [49], where N_{MVC} is the number of all minimum vertex covers.

Let m be the number of nonlocal control phase gates in the control-phase conversion of a quantum circuit (Eq. (74)). The vertex number $|\mathbb{V}|$ and edge number $|\mathbb{E}|$ of a packing graph are upper bounded by $2m$ and m, respectively. The complexity of the algorithm is then estimated as

$$\mathcal{O}(\sqrt{2}m^{3/2} + 2mN_{\mathrm{MVC}}). \tag{145}$$

The complexity of the first two steps is

$$\mathcal{O}(\sqrt{2}m^{3/2}(N_1 + 1) + 2m(N_2 + 1)N_1) \tag{146}$$

where $N_1 = |\{\mathbb{K}_i\}_i|$ is the number of the minimum vertex covers in step 1, while $N_2 = \max_i |\{\mathbb{R}_{i,j}\}_j|$ is the maximum cardinality of the set of minimum vertex covers obtained in step 2.

The number of minimum vertex covers can be upper bounded by the number of minimal vertex covers, which is equal to the number of maximal independent sets and upper bounded by $3^{|\mathbb{V}|/3} \leq 3^{2m/3}$ according to [53]. In the worst case, the complexity of the first two steps is $O(m\, 3^{4m/3})$.

In the last step, one needs to determine the chromatic number of the extrinsic conflict graph $C_{ex}(\mathbb{R}^{(A,B)}_{i,j})$. In general, the determination of the chromatic number has a complexity of $\mathcal{O}(2^{|\mathbb{V}|}|\mathbb{V}|)$ [51], which is NP-hard. Overall, to obtain the full information about all the options of packing strategies $\{(\mathbb{R}_{i,j}, \chi^{(A)}_{i,j}, \chi^{(B)}_{i,j})\}_{i,j}$ with Algorithm 27, the complexity is roughly upper bounded by

$$\mathcal{O}(m^2\, 3^{4m/3} 2^{2m}), \tag{147}$$

which is exponentially increasing.

For a heuristic implementation of the algorithm, one can simply find a minimum vertex cover without enumerating all of them. The complexity of finding a minimum vertex cover can be estimated by $O(|\mathbb{V}|^{1/2}|\mathbb{E}|)$ according to the Hopcroft–Karp algorithm [54] Besides, an upper bound on the chromatic number of a conflict graph can also be efficiently determined by $\chi \leq \max_{v\in\mathbb{V}}(d_v + 1)$, where d_v is the degree of a vertex, with a time complexity of $\mathcal{O}(|\mathbb{V}|^2)$. As a whole, the heuristic implementation of Algorithm 27 has a time complexity of

$$\mathcal{O}(m^{7/2}), \tag{148}$$

which is polynomial.

In Fig. 25, the runtimes t of Algorithm 27 for different UCC circuits are plotted with respect to the number of the nonlocal gates m in the circuits. It shows a polynomial increasing of the runtime t with respect to the nonlocal gates m in the order of 3.157.

Algorithm 28 only differs from Algorithm 27 on step 3, where one searches for χ_A and χ_B partitioning on the extrinsic graph $C_{ex}(\mathbb{R}^{(A)}_{i,j})$ and $C_{ex}(\mathbb{R}^{(B)}_{i,j})$, respectively, instead of the determination of their chromatic numbers. The complexity of $\chi_{A,B}$ partitioning is equal to the determination of the chromatic number [51]. The complexity of Algorithm 28 is therefore the same as Algorithm 27, which is given in Eq. (147). The complexity of the heuristic solution for Algorithm 28 is the same as the one given in Eq. (148).

D.6 Proof of Lemma 29

According to Definition 23, a connecting gate g is distributable on both q_1 and q_2. As a result of Theorem

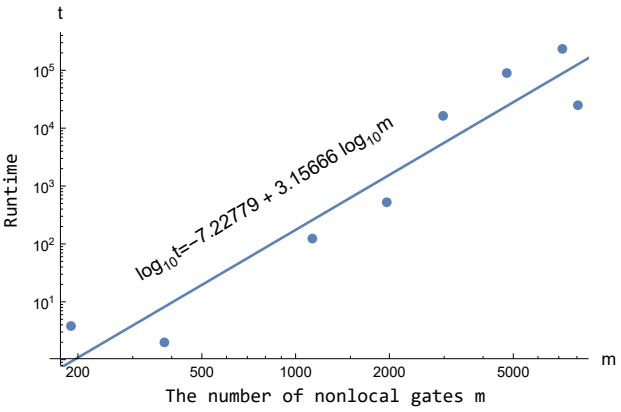


Figure 25: The log-log plot of the runtimes of the packing algorithm (Algorithm 27) for different UCC circuits.

5, the gate g can be decomposed as

$$g = \sum_{i,j} u_{ij} |i\rangle \langle j| \otimes U_j \tag{149}$$

$$= \sum_{i,j} v_{ij} V_j \otimes |i\rangle \langle j|, \tag{150}$$

where $\{u_{ij}\}_{i,j}$ and $\{v_{ij}\}_{i,j}$ are diagonal or antidiagonal. Let D_1 and D_2 be diagonal or antidiagonal single-qubit gates such that $\{u_{ij}\}_{i,j} = D_1.V(\varphi_1)$ and $\{\tilde{u}_{ij}\}_{i,j} = D_2.V(\varphi_2)$ is diagonal. It holds then

$$\begin{aligned}
&(D_1^\dagger \otimes D_2^\dagger) g \\
=& |0\rangle \langle 0| \otimes D_2^\dagger U_0 + |1\rangle \langle 1| \otimes (e^{\mathbb{i}\,\varphi_1} D_2^\dagger U_1) && (151)\\
=& D_1^\dagger V_0 \otimes |0\rangle \langle 0| + (e^{\mathbb{i}\,\varphi_2} D_1^\dagger V_1) \otimes |1\rangle \langle 1| && (152)
\end{aligned}$$

It follows that $D_1^\dagger V_i$ and $D_2^\dagger V_i$ are diagonal unitary. Furthermore, $D_1^\dagger V_0$ and $D_2^\dagger U_0$ fulfill $\langle 0|D_1^\dagger V_0|0\rangle = \langle 0|D_2^\dagger U_0|0\rangle$. It follows that

$$\begin{aligned}
&(D_1^\dagger \otimes D_2^\dagger) G (\tilde{D}_1^\dagger \otimes \tilde{D}_2^\dagger) \\
=& |0\rangle \langle 0| \otimes \mathbb{1} + && (153)\\
& |1\rangle \langle 1| \otimes (e^{\mathbb{i}\,\varphi_1} \langle 0|D_1^\dagger V_0|0\rangle \langle 1|D_1^\dagger V_0|1\rangle^* D_2^\dagger U_1 U_0^\dagger D_2) \\
=& \mathbb{1} \otimes |0\rangle \langle 0| + && (154)\\
& (e^{\mathbb{i}\,\varphi_2} \langle 0|D_2^\dagger U_0|0\rangle \langle 1|D_2^\dagger U_0|1\rangle^* D_1^\dagger V_1 V_0^\dagger D_1) \otimes |1\rangle \langle 1|
\end{aligned}$$

where

$$\tilde{D}_1 = \langle 0|D_1^\dagger V_0|0\rangle^* D_1^\dagger V_0 \tag{155}$$

$$\tilde{D}_2 = D_2^\dagger U_0. \tag{156}$$

Furthermore, one can then show the following equality

$$\begin{aligned}
& e^{\mathbb{i}\,\varphi_1} \langle 0|D_1^\dagger V_0|0\rangle \langle 1|D_1^\dagger V_0|1\rangle^* D_2^\dagger U_1 U_0^\dagger D_2 \\
=& e^{\mathbb{i}\,\varphi_2} \langle 0|D_2^\dagger U_0|0\rangle \langle 1|D_2^\dagger U_0|1\rangle^* D_1^\dagger V_1 V_0^\dagger D_1 \\
=& |0\rangle \langle 0| + e^{\mathbb{i}\,\theta} |1\rangle \langle 1| = V(\theta). && (157)
\end{aligned}$$

As a result,

$$(D_1^\dagger \otimes D_2^\dagger) g (\tilde{D}_1^\dagger \otimes \tilde{D}_2^\dagger) = |0\rangle \langle 0| \otimes \mathbb{1} + |1\rangle \langle 1| \otimes V(\theta). \tag{158}$$

D.7 Proof of Lemma 31

Two remote control-phase nodes $\{(q,t_i),(q,t_j)\}$ with $i \neq j-2$ form a $\mathbb{B}$-distributable packet $\mathcal{V}_{q,\{t_i,t_j\}}$, if and only if the unitary $W_{t_j;t_i}$ between them is $\mathbb{B}$-embeddable through a hopping embedding consisting of several embedding units. According to the embedding rules $\mathbb{B}$ in Theorem 10 and Corollary 30, all embedding rules for embedding units are the global D-type, local D-type and global H-type and as shown in Fig. 18 (a,b,c). The unitary $W_{t_j;t_i}$ between $\{(q,t_i),(q,t_j)\}$ is $\mathbb{B}$-embeddable, if and only if it can be decomposed as

$$W_{t_j;t_i} = F_{q,t_j} \left(\prod_{k:i<k<j} \widetilde{U}_{t_k} \right) G_{q,t_i}, \tag{159}$$

where $\widetilde{U}_{t_k}$ are D-type or H-type embedding units.

Suppose there exists a D-type embedding unit U_{t_l}, one can then decompose $W_{t_j;t_i}$ into two embeddings

$$W_{t_j;t_i} = W_{t_j;t_l} C_V^{q,q'_l}(\theta_l) W_{t_l;t_i}, \tag{160}$$

where $W_{t_j;t_l}$ and $W_{t_l;t_i}$ are both $\mathbb{B}$-embeddable given by

$$W_{t_j;t_l} = F_{q,t_j} \left(\prod_{k:i<k<l} \widetilde{U}_{t_k} \right) G_{q,t_l}, \tag{161}$$

$$W_{t_l;t_i} = F_{q,t_l} \left(\prod_{k:l<k<i} \widetilde{U}_{t_k} \right) G_{q,t_i}. \tag{162}$$

As a result, the distributable packet $\mathcal{V}_{q,\{t_i,t_j\}}$ is indecomposable, only if the embedding units in Eq. (159) are all H-type. This completes the proof.

D.8 The time complexity of Algorithm 33

Up to line 8, the complexity is equal to $\mathcal{O}(m_1+m_2)$, where m_1 and m_2 are the numbers of single-qubit and two-qubit gates, respectively. For each qubit, the identification of neighboring distributable packets has the complexity $\mathcal{O}(d)$, where d is the depth of a circuit. For the identification of hopping distributable packets in Algorithm 35, the complexity is determined by the examination of the embeddability of $W_{t_j;t_i}$ on line 10, which is upper bounded by $\mathcal{O}(d(d-1))$. As a result, the complexity of Algorithm 33 is estimated by $\mathcal{O}(m_1+m_2+n\times d(d-1))$, where n is the number of total qubits from all parties. Since $m_{1,2} \leq nd$, the complexity is upper bounded by $\mathcal{O}(nd^2)$.

D.9 Proof of line 25 in Algorithm 35

We need to prove that a unitary $W_{t_j;t_i}$ given in the following form is q-rooted embeddable according to

the embedding rules $\mathbb{B}$, if and only if $\alpha_{k+1}+\gamma_k = n\pi$ for all k,

$$\begin{aligned} W_{t_j;t_i} &= V(\gamma_{j-1})\, H\, V(\alpha_{j-1})\, C_V^{(q,q_{j-1})} \\ &\times V(\gamma_{j-2})\, H\, V(\beta_{j-2})\, H\, V(\alpha_{j-2})\, C_V^{(q,q_{j-2})} \\ &\times \cdots \\ &\times V(\gamma_{i+1})\, H\, V(\beta_{i+1})\, H\, V(\alpha_{i+1})\, C_V^{(q,q_{i+1})} \\ &\times V(\gamma_i)\, H\, V(\alpha_i). \end{aligned} \tag{163}$$

Since the phase gates in unitary $W_{t_j;t_i}$ commute with C_V gates, one can merge them together as follows

$$\begin{aligned} W_{t_j;t_i} &= V(\gamma_{j-1})\, H\, C_V^{(q,q_{j-1})} \\ &\times V(\alpha_{j-1}+\gamma_{j-2})\, H\, V(\beta_{j-2})\, H\, C_V^{(q,q_{j-2})} \\ &\times \cdots \\ &\times V(\alpha_{i+1}+\gamma_{i+1})\, H\, V(\beta_{i+1})\, H\, C_V^{(q,q_{i+1})} \\ &\times V(\alpha_{i+1}+\gamma_i)\, H\, V(\alpha_i). \end{aligned} \tag{164}$$

According to the embedding rules $\mathbb{B}_2$ in Corollary 30, $W_{t_j;t_i}$ is q-embeddable only if the gate C_V must form a H-type embedding unit. To satisfy this necessary condition, we insert $H\,H$ into $W_{t_j;t_i}$ to construct H-type embedding units HC_VH,

$$\begin{aligned} W_{t_j;t_i} &= V(\gamma_{j-1})\, H\, C_V^{(q,q_{j-1})} H \\ &\times HV(\alpha_{j-1}+\gamma_{j-2})\, H\, V(\beta_{j-2})\, H\, C_V^{(q,q_{j-2})} H \\ &\times \cdots \\ &\times HV(\alpha_{i+1}+\gamma_{i+1})\, H\, V(\beta_{i+1})\, H\, C_V^{(q,q_{i+1})} H \\ &\times HV(\alpha_{i+1}+\gamma_i)\, H\, V(\alpha_i). \end{aligned} \tag{165}$$

This unitary is q-embeddable according to $\mathbb{B}_1$ in Theorem 10 and $\mathbb{B}_2$ in Corollary 30, if and only if $HV(\alpha_{k+1}+\gamma_k)HV(\beta_k)$ is embeddable for all k. It is equivalent to the following condition

$$\alpha_{k+1}+\gamma_k = n\pi. \tag{166}$$

This completes the proof.

D.10 Proof of Theorem 36

In this section, we prove the compatibility of the primitive embedding rules in a circuit consisting of single-qubit and two-qubit gates. Fig. 20 shows all the three possible cases of nested distributable packets. In Fig. 20 (a,b), the two distributable packets are on the same qubit q.

For (a), it shows the distributable packets $\mathcal{V}_{q,\{t_0,t_5\}}$ and $\mathcal{V}_{q,\{t_1,t_6\}}$ identified by two nested embeddings. We first implement the primitive embedding of $W_{t_5;t_0}$. According to the embedding rules $\mathbb{B}$ in Eq. (84), the additional kernels are all gates acting on system B, which are irrelevant for the embedding of $W^{(q)}_{t_6;t_2}$ on q. For the embedding of $W^{(q)}_{t_6;t_2}$ on q, the only additional gate is the control-X gate C_{q,X_e} in the ending process of embedding of $W^{(q)}_{t_5;t_0}$, which acts before (q,t_5). One can shift the control-X gate C_{q,X_e} into the block $\widetilde{U}_{t_5}$

$$\widetilde{U}_{t_5} = (\widetilde{D}_q \otimes G_{q_{n-1}})\, C_{q,X_e} C_V(\theta_5)\, (D_q \otimes F_{q_{n-1}}) \tag{167}$$

For the indecomposable embedding of $W^{(q)}_{t_6;t_2}$, one can find a decomposition of local gates on q that forms a H-type embedding unit for $\widetilde{U}_{t_5}$.

$$\widetilde{U}_{t_5} = (\widetilde{D}_q H_q \otimes G_{q_{n-1}})\, C_{q,X_e} C_V(\theta_5)\, (H_q D_q \otimes F_{q_{n-1}}). \tag{168}$$

The unitary $\widetilde{U}_{t_5}$ can be decomposed into two H-type embedding units by inserting $H_q^2 \otimes \mathbb{1}$ between the two control gates C_{q,X_e} and $C_V(\theta_5)$, as follow

$$\begin{aligned} \widetilde{U}_{t_5} &= (\widetilde{D}_q H_q \otimes G_{q_{n-1}})\, C_{q,X_e} (H_q \otimes \mathbb{1}_B) \\ &\times (H_q \otimes \mathbb{1}_B) C_V(\theta_5)\, (H_q D_q \otimes F_{q_{n-1}}). \end{aligned} \tag{169}$$

It means that one can implement the embedding of $W_{t_6;t_2}$ with an additional local gate $(H_e \otimes H_{e'}) C_Z^{(e,e')} (H_e \otimes H_{e'})$ acting on the two auxiliary qubit $\{e,e'\}$. Removing duplicated Hadamard gates between two control-Z gate on the auxiliary qubits $\{e,e'\}$, the implementation of embedding is shown in Fig. 20 (b), where the additional local gate $C_Z^{(e,e')}$ is highlighted in green. This proves the compatibility of two nested embedding shown in Fig. 20 (a).

For the case shown in Fig. 20 (c), one of the embedding $W^{(q)}_{t_5;t_2}$ is included in the embedding of $W^{(q)}_{t_6;t_0}$. One can first implement the embedding $W^{(q)}_{t_6;t_0}$, of which the local kernels of embedding all acts on B and are all irrelevant with the embedding of $W_{t_5;t_2}$. It means that the embedding of $W^{(q)}_{t_5;t_2}$ is not affected by the embedding of $W^{(q)}_{t_6;t_0}$. These two embeddings are therefore compatible. The implementation of the two embeddings are shown in Fig. 20 (d).

For the case shown in Fig. 20 (e), the two nested embeddings are rooted on two qubits $\{q,q'\}$, which are on two different local systems. The embeddings of $W^{(q)}_{t_5;t_0}$ can be decomposed into two embedding units $\widetilde{U}_{t_2}$ and $\widetilde{U}_{t_3}$, while $W^{(q')}_{t_6;t_1}$ can be decomposed into $\widetilde{U}_{t_3}$ and $\widetilde{U}_{t_4}$. The two hopping embeddings share the same embedding unit $\widetilde{U}_{t_3}$. If one wants to implement the two hopping embedding together, one has to implement the joint embedding of $\widetilde{U}_{t_3}$ on $\{q,q'\}$. The primitive recursive embedding of $\widetilde{U}_{t_3}$ will lead to a global control-Z gate $C_Z^{(e,e')}$ acting on the two auxiliary $\{e,e'\}$,

$$\begin{aligned} &C_{q,X_e} C_{q',X_{e'}}\, (H_q \otimes H_{q'})\, C_V(\theta)\, (H_q \otimes H_{q'})\, C_{q',X_{e'}} C_{q,X_e} \\ &= (H_e \otimes H_{e'})\, C_Z^{(q,e')} C_Z^{(q',e)} C_V(\theta) C_Z^{(e,e')}\, (H_e \otimes H_{e'})\,. \end{aligned} \tag{170}$$

The incompatibility of the nested embeddings is caused by the incompatibility of the recursive embedding of the gate embedding unit $\widetilde{U}_{t_3}$.

The distributable packets $\mathcal{V}_{q,\{t_i,t_j\}}$ and $\mathcal{V}_{q',\{t_k,t_l\}}$ are incompatible, if and only if $\{q,q'\}$ belong to two local systems, and there is an embedding unit $(H_q \otimes H_{q'})\, C_V^{(q,q')}(\theta)\, (H_q \otimes H_{q'})$ located on t with $t_i < t < t_j$ and $t_k < t < t_l$. Existence of such an embedding unit for two distributable packets is equivalent to the existence of a control-phase gate acting on $\{q,q'\}$ at a depth of t with $t_i < t < t_j$ and $t_k < t < t_l$. This completes the proof.

References

[1] J. Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2018 2:79. DOI: 10.22331/q-2018-08-06-79.

[2] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn, A. Kandala, A. Mezzacapo, P. Müller, W. Riess, G. Salis, J. Smolin, I. Tavernelli, and K. Temme. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, 2018 3(3):030503. DOI: 10.1088/2058-9565/aab822.

[3] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta. Validating quantum computers using randomized model circuits. *Physical Review A*, 2019 100:032328. DOI: 10.1103/PhysRevA.100.032328.

[4] S. Bose, P. L. Knight, M. B. Plenio, and V. Vedral. Proposal for teleportation of an atomic state via cavity decay. *Physical Review Letters*, 1999 83:5158–5161. DOI: 10.1103/PhysRevLett.83.5158.

[5] C. Cabrillo, J. I. Cirac, P. García-Fernández, and P. Zoller. Creation of entangled states of distant atoms by interference. *Physical Review A*, 1999 59:1025–1033. DOI: 10.1103/PhysRevA.59.1025.

[6] D. E. Browne, M. B. Plenio, and S. F. Huelga. Robust creation of entanglement between ions in spatially separate cavities. *Physical Review Letters*, 2003 91:067901. DOI: 10.1103/PhysRevLett.91.067901.

[7] L.-M. Duan, B. Blinov, D. Moehring, and C. Monroe. Scalable trapped ion quantum computation with a probabilistic ion-photon mapping. *Quantum Information and Computation*, 2004 4:165–173. DOI: 10.48550/arXiv.quant-ph/0401020.

[8] Y. L. Lim, A. Beige, and L. C. Kwek. Repeat-until-success linear optics distributed quantum computing. *Physical Review Letters*, 2005 95: 030505. DOI: 10.1103/PhysRevLett.95.030505.

[9] L.-M. Duan, M. J. Madsen, D. L. Moehring, P. Maunz, R. N. Kohn, and C. Monroe. Probabilistic quantum gates between remote atoms through interference of optical frequency qubits. *Physical Review A*, 2006 73:062324. DOI: 10.1103/PhysRevA.73.062324.

[10] Z.-q. Yin, W. L. Yang, L. Sun, and L. M. Duan. Quantum network of superconducting qubits through an optomechanical interface. *Physical Review A*, 2015 91:012333. DOI: 10.1103/PhysRevA.91.012333.

[11] K. Koshino, K. Inomata, Z. R. Lin, Y. Tokunaga, T. Yamamoto, and Y. Nakamura. Theory of deterministic entanglement generation between remote superconducting atoms. *Physical Review Applied*, 2017 7:064006. DOI: 10.1103/PhysRevApplied.7.064006.

[12] D. L. Moehring, P. Maunz, S. Olmschenk, K. C. Younge, D. N. Matsukevich, L.-M. Duan, and C. Monroe. Entanglement of single-atom quantum bits at a distance. *Nature*, 2007 449(7158): 68–71. DOI: 10.1038/nature06118.

[13] L. Slodička, G. Hétet, N. Röck, P. Schindler, M. Hennrich, and R. Blatt. Atom-atom entanglement by single-photon detection. *Physical Review Letters*, 2013 110(8):083603. DOI: 10.1103/physrevlett.110.083603.

[14] S. Ritter, C. Nolleke, C. Hahn, A. Reiserer, M. Neuzner, Andreas andUphoff, M. Mucke, E. Figueroa, J. Bochmann, , and G. Rempe. An elementary quantum network of single atoms in optical cavities. *Nature*, 2012 484:195–200. DOI: 10.1038/nature11023.

[15] J. Hofmann, M. Krug, N. Ortegel, L. Gérard, M. Weber, W. Rosenfeld, and H. Weinfurter. Heralded entanglement between widely separated atoms. *Science*, 2012 337(6090):72–75. DOI: 10.1126/science.1221856.

[16] H. Bernien, B. Hensen, W. Pfaff, G. Koolstra, M. S. Blok, L. Robledo, T. H. Taminiau, M. Markham, D. J. Twitchen, L. Childress, and R. Hanson. Heralded entanglement between solid-state qubits separated by three metres. *Nature*, 2013 497(7447):86–90. DOI: 10.1038/nature12016.

[17] A. Delteil, Z. Sun, W. bo Gao, E. Togan, S. Faelt, and A. Imamoğlu. Generation of heralded entanglement between distant hole spins. *Nature Physics*, 2015 12(3):218–223. DOI: 10.1038/nphys3605.

[18] R. Stockill, M. J. Stanley, L. Huthmacher, E. Clarke, M. Hugues, A. J. Miller, C. Matthiesen, C. Le Gall, and M. Atatüre. Phase-tuned entangled state generation between distant spin qubits. *Physical Review Letters*, 2017 119:010503. DOI: 10.1103/PhysRevLett.119.010503.

[19] A. Narla, S. Shankar, M. Hatridge, Z. Leghtas, K. M. Sliwa, E. Zalys-Geller, S. O. Mundhada, W. Pfaff, L. Frunzio, R. J. Schoelkopf, and M. H. Devoret. Robust concurrent remote entanglement between two superconducting

qubits. *Physical Review X*, 2016 6:031036. DOI: 10.1103/PhysRevX.6.031036.

[20] L. J. Stephenson, D. P. Nadlinger, B. C. Nichol, S. An, P. Drmota, T. G. Ballance, K. Thirumalai, J. F. Goodwin, D. M. Lucas, and C. J. Ballance. High-rate, high-fidelity entanglement of qubits across an elementary quantum network. *Physical Review Letters*, 2020 124:110501. DOI: 10.1103/PhysRevLett.124.110501.

[21] D. Hucul, I. V. Inlek, G. Vittorini, C. Crocker, S. Debnath, S. M. Clark, and C. Monroe. Modular entanglement of atomic qubits using photons and phonons. *Nature Physics*, 2014 11(1):37–42. DOI: 10.1038/nphys3150.

[22] H. J. Kimble. The quantum internet. *Nature*, 453:1023–1030, 2008. DOI: 10.1038/nature07127.

[23] A. Soeda, Y. Kinjo, P. S. Turner, and M. Murao. Quantum computation over the butterfly network. *Physical Review A*, 2011 84:012333. DOI: 10.1103/PhysRevA.84.012333.

[24] D. Gottesman and I. L. Chuang. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature*, 402:390–393, 1999. DOI: 10.1038/46503.

[25] X. Zhou, D. W. Leung, and I. L. Chuang. Methodology for quantum logic gate construction. *Physical Review A*, 2000 62:052316. DOI: 10.1103/PhysRevA.62.052316.

[26] J. Eisert, K. Jacobs, P. Papadopoulos, and M. B. Plenio. Optimal local implementation of nonlocal quantum gates. *Physical Review A*, 2000 62: 052317. DOI: 10.1103/PhysRevA.62.052317.

[27] S. F. Huelga, J. A. Vaccaro, A. Chefles, and M. B. Plenio. Quantum remote control: Teleportation of unitary operations. *Physical Review A*, 2001 63:042303. DOI: 10.1103/PhysRevA.63.042303.

[28] S. F. Huelga, M. B. Plenio, and J. A. Vaccaro. Remote control of restricted sets of operations: Teleportation of angles. *Physical Review A*, 2002 65:042316. DOI: 10.1103/PhysRevA.65.042316.

[29] L. Jiang, J. M. Taylor, A. S. Sørensen, and M. D. Lukin. Distributed quantum computation based on small quantum registers. *Physical Review A*, 2007 76:062323. DOI: 10.1103/PhysRevA.76.062323.

[30] R. V. Meter, W. Munro, K. Nemoto, and K. M. Itoh. Arithmetic on a distributed-memory quantum multicomputer. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 3 (4):1–23, 2008. DOI: 10.1145/1324177.1324179.

[31] M. Caleffi, M. Amoretti, D. Ferrari, D. Cuomo, J. Illiano, A. Manzalini, and A. S. Cacciapuoti. Distributed quantum computing: a survey, 2022. DOI: 10.48550/arXiv.2212.10609.

[32] K. S. Chou, J. Z. Blumoff, C. S. Wang, P. C. Reinhold, C. J. Axline, Y. Y. Gao, L. Frunzio, M. H. Devoret, L. Jiang, and R. J. Schoelkopf. Deterministic teleportation of a quantum gate between two logical qubits. *Nature*, 2018 561 (7723):368–373. DOI: 10.1038/s41586-018-0470-y.

[33] Y. Wan, D. Kienzler, S. D. Erickson, K. H. Mayer, T. R. Tan, J. J. Wu, H. M. Vasconcelos, S. Glancy, E. Knill, D. J. Wineland, A. C. Wilson, and D. Leibfried. Quantum gate teleportation between separated qubits in a trapped-ion processor. *Science*, 364(6443):875–878, 2019. DOI: 10.1126/science.aaw9415.

[34] P. Andrés-Martínez and C. Heunen. Automated distribution of quantum circuits via hypergraph partitioning. *Physical Review A*, 2019 100: 032308. DOI: 10.1103/PhysRevA.100.032308.

[35] R. G. Sundaram, H. Gupta, and C. R. Ramakrishnan. Efficient Distribution of Quantum Circuits. In S. Gilbert, editor, *35th International Symposium on Distributed Computing (DISC 2021)*, volume 209 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 41:1–41:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-210-5. DOI: 10.4230/LIPIcs.DISC.2021.41.

[36] D. Cuomo, M. Caleffi, K. Krsulich, F. Tramonto, G. Agliardi, E. Prati, and A. S. Cacciapuoti. Optimized compiler for distributed quantum computing. *ACM Transactions on Quantum Computing*, 2023 4(2). ISSN 2643-6809. DOI: 10.1145/3579367.

[37] R. G. Sundaram, H. Gupta, and C. R. Ramakrishnan. Distribution of quantum circuits over general quantum networks. In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 415–425, 2022 Los Alamitos, CA, USA. IEEE Computer Society. DOI: 10.1109/QCE53715.2022.00063.

[38] D. Stahlke and R. B. Griffiths. Entanglement requirements for implementing bipartite unitary operations. *Physical Review A*, 2011 84:032316. DOI: 10.1103/PhysRevA.84.032316.

[39] P. Andres-Martinez, T. Forrer, D. Mills, J.-Y. Wu, L. Henaut, K. Yamamoto, M. Murao, and R. Duncan. Distributing circuits over heterogeneous, modular quantum computing network architectures. DOI: 10.48550/arXiv.2305.14148.

[40] J. M. Baker, C. Duckering, A. Hoover, and F. T. Chong. Time-sliced quantum circuit partitioning for modular architectures. In *Proceedings of the 17th ACM International Conference on Computing Frontiers*, CF '20, page 98–107, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379564. DOI: 10.1145/3387902.3392617.

[41] S. DiAdamo, M. Ghibaudi, and J. Cruise. Distributed quantum computing and network control for accelerated vqe. *IEEE Transactions*

on Quantum Engineering, 2:1–21, 2021. DOI: 10.1109/TQE.2021.3057908.

[42] D. Ferrari, A. S. Cacciapuoti, M. Amoretti, and M. Caleffi. Compiler design for distributed quantum computing. *IEEE Transactions on Quantum Engineering*, 2:1–20, 2021. DOI: 10.1109/TQE.2021.3053921.

[43] A. Ovide, S. Rodrigo, M. Bandic, H. Van Someren, S. Feld, S. Abadal, E. Alarcon, and C. G. Almudever. Mapping quantum algorithms to multi-core quantum computing architectures. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2023. DOI: 10.1109/ISCAS46773.2023.10181589.

[44] D. Ferrari, S. Carretta, and M. Amoretti. A modular quantum compilation framework for distributed quantum computing. *IEEE Transactions on Quantum Engineering*, 2023 4(01):1–13. ISSN 2689-1808. DOI: 10.1109/TQE.2023.3303935.

[45] A. G. Taube and R. J. Bartlett. New perspectives on unitary coupled-cluster theory. *International Journal of Quantum Chemistry*, 106(15): 3393–3401, 2006. DOI: 10.1002/qua.21198.

[46] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 2014 5(1). DOI: 10.1038/ncomms5213.

[47] D. Jonathan and M. B. Plenio. Entanglement-assisted local manipulation of pure quantum states. *Physical Review Letters*, 1999 83:3566–3569. DOI: 10.1103/PhysRevLett.83.3566.

[48] A. Yimsiriwattana and S. J. Lomonaco. Generalized GHZ states and distributed quantum computing, 2004. DOI: 10.48550/ARXIV.QUANT-PH/0402148.

[49] T. Uno. Algorithms for enumerating all perfect, maximum and maximal matchings in bipartite graphs. In *Algorithms and Computation*, pages 92–101. Springer Berlin Heidelberg. DOI: 10.1007/3-540-63890-3_11.

[50] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Mathematical Sciences Series. Freeman, 1979. ISBN 9780716710448.

[51] A. Björklund, T. Husfeldt, and M. Koivisto. Set partitioning via inclusion-exclusion. *SIAM Journal on Computing*, 39(2):546–563, 2009. DOI: 10.1137/070683933.

[52] R. Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, 2005 2005 edition. DOI: 10.1007/978-3-662-53622-3.

[53] J. W. Moon and L. Moser. On cliques in graphs. *Israel journal of Mathematics*, 3:23–28, 1965. DOI: 10.1007/BF02760024.

[54] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. 2(4):225–231. DOI: 10.1137/0202019.