Bachelor's Thesis (Academic Year 2020)

Resource Allocation Policy for Noisy Distributed Quantum Computing

A thesis presented for the degree of
Bachelor of Environment and Information Studies

Keio University
Faculty of Environment and Information Studies

Ryosuke Satoh

Bachelor's Thesis (Academic Year 2020)

Resource Allocation Policy for Noisy Distributed Quantum Computers

Multicore processing and distributed computing are crucial applications for the success of these day's computation. Since classical CPU's physical limitation gets close, as shown in Moor's law, quantum computing is expected as the next-generation computing. Similar to classical computing, multi-processing and distributed computing can be thought of for future quantum computing applications. This thesis aims to establish a resource allocation method that efficiently prepares a distributed quantum state, especially a graph state, which is an important resource for quantum computing.

A key element of this method is a decision tree composed of three different graph algorithms. By preparing three different graph algorithms, the allocator is able to deal with various types of graphs. This resource allocation method outperformed the random allocation in terms of the quality of the final state in the presence of noise. This allocator is four times better than the naive allocation method in terms of error tolerance in a simple situation. Even with a very complex graph, this allocator still keeps generating a better state than the other allocation methods. As for the scalability of this method, it works in a very straightforward way. However, the time complexity of an entire system is not small. By setting a threshold for search space, it is possible to suppress the time complexity of the entire process. These results show us the path to efficient resource allocation in a practical situation.

Keywords: 1. Quantum Computing, 2. Distributed Quantum Computing, 3. Quantum Networking, 4. Graph State 5. Graph Theory

Keio University
Faculty of Environment and Information Studies

Ryosuke Satoh

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Quantum computing has been developed as one candidate for next-generation computing. Shor's algorithm and Grover's algorithm are two well-known quantum algorithms that might outperform classical computing. Recently, the quantum Internet also attracts researchers because of its capability of quantum information but also its applicability. One of the most famous quantum Internet applications is Quantum Key Distribution that allows us to share our secrete with a completely secure process.

Distributed Quantum Computing (DQC) is an future platform on top of those two technologies. Quantum computers work cooperatively and enhance their computational power. There have not been built large quantum devices capable of DQC. However, it is predicted that quantum computers can handle several hundreds of qubits in the next decades. Under that situation, small scale DQC would be possible in the real device. This paper aims to establish an efficient way to allocate qubits over the distributed system.

1.2 Research Contribution

This project's main contribution is establishing a resource allocation method to create a quantum state for DQC. This allocation method outperformed a naive random allocation in terms of the quality of the final state.

In most cases, this thesis's allocation method outperformed a random and naive allocation method in terms of the quality measured by stabilizer measurement. The best results show that this thesis's allocation method is 3.79 times efficient than the random allocation in terms of the error tolerance. If the network topology is small enough to handle, the allocation method can deal with a reasonably large graph state (hundreds of qubits). Those results allow us to improve the resource allocations for distributed quantum computers.

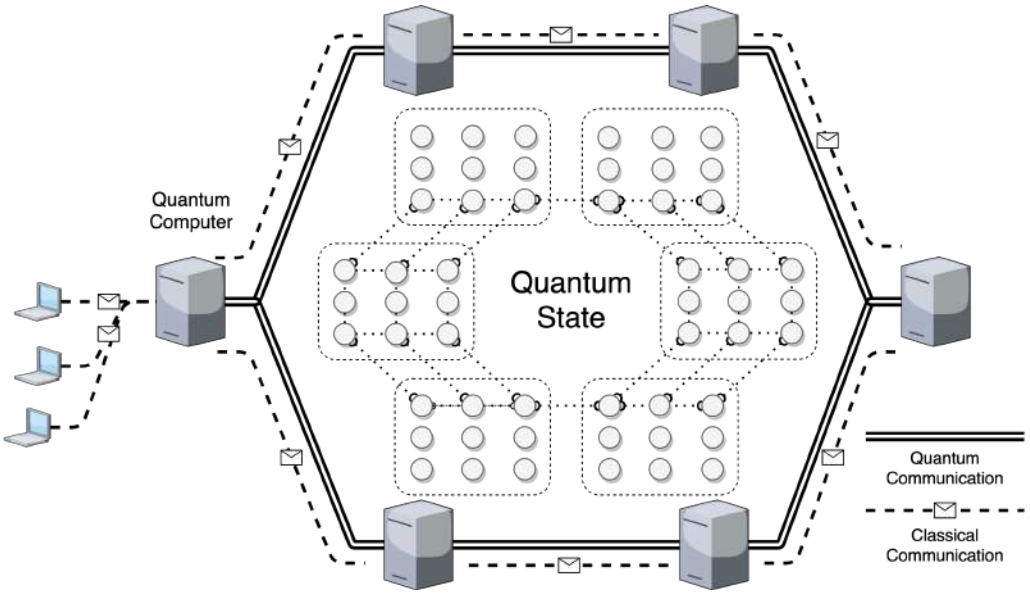


Figure 1.1: The diagram of distributed quantum computing. All quantum computer are connected by quantum links and classical links. The quantum computers work together and generate large quantum state over the distributed system.

1.3 Thesis Structure

The structure of this thesis is as follows. Background information about quantum computing is shown in Chapter 2. This chapter explains basic knowledge about what is quantum computing and networking. Several graph algorithms explained in Chapter 2 are key ideas in this thesis.

Chapter 3 explains the main proposals to establish a qubit allocation method for the distributed systems. This chapter also lists the related works of distributed quantum computing and graph state generation and introduces the key concepts of the solution. Chapter 4 shows experiments to investigate the performance and results. Three different indices evaluate the allocation method comparing to naive allocations. Finally, in chapter 5, conclude this paper and give some future perspectives.

Chapter 2

Theory of Quantum Information

2.1 History of Quantum computation and quantum information

The term "Quantum Computer" appeared for the first time in [1] by Richard P. Feynman. It was proposed as a new type of computer for simulating quantum systems. In the classical computer, we haven't been able to simulate a quantum system without approximation most of the time. However, by supposing the existence of a quantum computer based on quantum mechanics, Feynman showed we would be able to simulate it in the exact solution.

Until now, Quantum Computing has been one active research field for physicists and computer scientists. There have been many algorithms proposed on the quantum computer. Two representative algorithms that are known as superior to the known classical algorithms were Grover's search algorithm [2] and Shor's factoring algorithm [3]. Grover's algorithm allows us to search for target data from unstructured data in quadratic time comparing to the known classical algorithms for the sake of the superposition 2.5. Shor's algorithm can factor a number in polynomial time, which is an exponentially difficult task with known classical algorithms. These two algorithms gave us a big motivation for building actual quantum computers. Other than those two quantum algorithms, quantum computing is good at simulating quantum systems accurately.

As experimental achievements of quantum computing, Y. Nakamura, Yu. A. Pashkin, and J. S. Tsai were the first people that showed the real superconducting qubit [4] in 1999. In 2008, David P. DiVincenzo from IBM showed criteria for building practical quantum computers [5]. Since then, several companies such as IBM, Google, Rigetti have developed actual hardware. IBM made its hardware public, and many people have been using real devices via the cloud. Google claimed that they achieved "Quantum supremacy," an experimental proof that the quantum computer is truly superior to a classical computer in 2019 [6].

Quantum Computing is one promising direction for future computing. However, quantum devices are still far from practical. John Preskill named this era "NISQ" (Noisy Intermediate-Scale Quantum) [7], which means the size of the processor is relatively small, and the processor is very fragile to outside noise.

Classical computers protect data from errors by copying data and making it redundant. However, because of the No-cloning theorem [8], a principle of quantum mechanics that prohibit us from copying unknown quantum state. One solution to this problem is error correction that protects quantum information from noise. Error correction will be the main scheme for future quantum computing and allows us to build a large scale quantum application.

2.2 Qubit

A bit is a unit of classical computation representing binary value 0 or 1. It has been deterministic which we get either 0 or 1. In quantum computation, we also have a computational unit called "Qubit." Qubit has different features from the classical bit. One common explanation is "Qubit has two different states 0 and 1 simultaneously." That feature is known as "Superposition," 2.5 which is a fundamental phenomenon in quantum mechanics. The state of the qubit is not deterministic. The only way to know the state is to measure it repeatedly. The qubit has two degrees of freedom. One is called "amplitude," and the other is called "phase." In the Bloch sphere representation shown in 2.1, the change of amplitude corresponds to the rotation on the X or Y-axis. The phase can be changed by Z-axis rotation.

Any two-level quantum system can be a qubit candidate such as polarization of photon, superconductor, and trapped ions. These qubit systems work differently in terms of the physical level, but overall notations are the same.

2.2.1 Notations

One general notation to describe a qubit is "Dirac Notation," developed by Paul Dirac [9]. In this notation, any quantum states can be represented as vectors that has same number of elements of computational basis which is in a part of Hilbert space $\mathcal{H}$. Suppose we have a qubit with two computational basis $|0\rangle$ and $|1\rangle$.

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.2.1}$$

We can describe one qubit state $|\psi\rangle$ with those two computational bases and coefficients.

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.2.2}$$

Arbitrary one qubit state can be described with these two computational bases and coefficients $\alpha, \beta \in \mathbb{C}$. α and β are arbitrary complex numbers which satisfy,

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2.2.3}$$

$|\alpha|^2$ and $|\beta|^2$ represents the probability of getting output "0" and "1" when we measure this qubit respectively.

As an example, $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ is a superposition state of $|0\rangle$ and $|1\rangle$. The probabilities of both states $|0\rangle$ and $|1\rangle$ are 0.5 and 0.5, $\left(\left|\frac{1}{\sqrt{2}}\right|^2 = \left|\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2}\right)$. We are also able to take negative and imaginary complex values as coefficients, such as $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$, $\frac{1}{\sqrt{2}}|0\rangle - \frac{i}{\sqrt{2}}|1\rangle$.

2.2.2 Bloch Sphere Representation

Bloch sphere is a visualization of single-qubit as shown in 2.1. Arbitrary one qubit state can be represented in the Bloch sphere. The north pole and south pole of the Bloch sphere corresponds to $|0\rangle$ and $|1\rangle$. All single-qubit operations correspond to the vector's rotations in the Bloch sphere with specific angles and degrees. Pauli-X gate 2.4.3 is corresponding to the 180 degree rotation on X-axis.

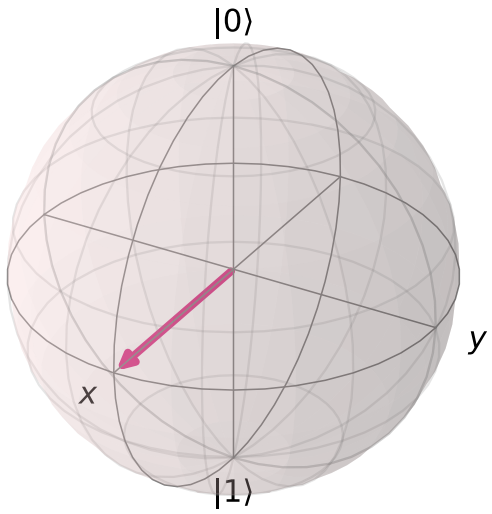


Figure 2.1: Bloch sphere for one qubit state representation. $|0\rangle$ and $|1\rangle$ are placed either side of Z axis. All one qubit operations can be expressed by the rotation of the vector in the bloch sphere. Image from [10].

2.3 Multiple qubit systems

It is possible to easily extend one qubit representation to a multi-qubit representation by taking its tensor product.

As an example, we have a two-qubit state $|\psi\rangle$. If it is initialized basis state, then we are able to write down it as,

$$|\psi\rangle = |00\rangle = |0\rangle_1 |0\rangle_2 . \tag{2.3.1}$$

where the subscripts 1 and 2 means the indices of the qubits. In the vector representation, $|\psi\rangle$ can be written as,

$$|\psi\rangle = |00\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} . \tag{2.3.2}$$

The size of the vector is 2^N, N is the number of qubits. That means the length of the vector grows exponentially as the number of qubits gets large. In other words, the entire Hilbert space gets exponentially large. The reason why classical computers can not simulate quantum computers, in general, is that the overall space is too big to calculate. The classical computers have been able to handle at most 53 qubits in a reasonable time.

2.4 Quantum Gates and Quantum Circuit Representations

A set of operations can be considered for constructing algorithms on quantum computers. In classical algorithms, we have several gates, such as AND, NOT, and NAND. All arithmetic calculations are based on these primitive gate sets. In quantum computing, completely the same gates can be assumed. However, there are gates that are unique to quantum computing.

There are essentially two types of operations, single-qubit, and two-qubit operations. The quantum operations for more than three qubits can also be built, but it is always possible to decompose it with single and two-qubit operations.

The next several subsections are examples of elementary quantum gates. All of the quantum gates are in a unitary group that has several mathematical features. Especially, the unitary matrix U satisfies $UU^\dagger = I$.

As shown in Figure 2.2, a quantum circuit is represented with wires. The circuit represents the time evolution of a quantum state from left to right.

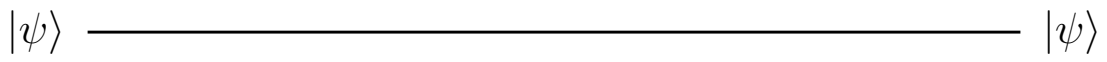


Figure 2.2: Quantum circuit representation. A Quantum state given at the left part of the wire is the initial state. The initial state can be an arbitrary quantum state, but generally, $|\psi\rangle = |0\rangle$. The number of wire represents the number of qubits.

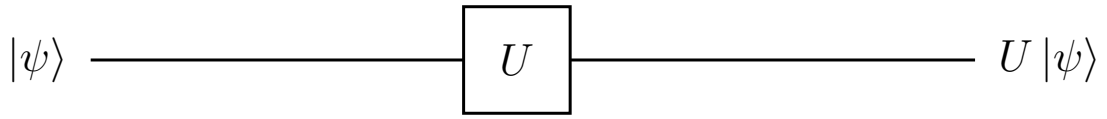


Figure 2.3: Circuit representation of single unitary gate. The output quantum state $U|\psi\rangle$ as an output sate.

2.4.1 Single qubit unitary gate

Any SU(2) matrix can be a single-qubit gate. In the circuit diagram, we usually show it with a box with U as Figure 2.3.

The definition of general SU(2) group matrix is,

$$U(\theta, \phi, \lambda) = \begin{bmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\phi}\sin(\theta/2) & e^{i\lambda+i\phi}\cos(\theta/2) \end{bmatrix} \tag{2.4.1}$$

where $0 \leq \theta, \phi, \lambda \leq 2\pi$.

By substituting appropriate angles to it, we can create any single-qubit operations. However, there are specific quantum operations that we often use in quantum algorithms. They are special cases of it and have a name.

2.4.2 Identity gate

There are four types of Pauli gate, Pauli-X, Pauli-Y, Pauli-Z, Identity. They are the most fundamental quantum gates. Identity gate is just a simple Identity matrix that does nothing to the quantum state. ($I|\psi\rangle = |\psi\rangle$)

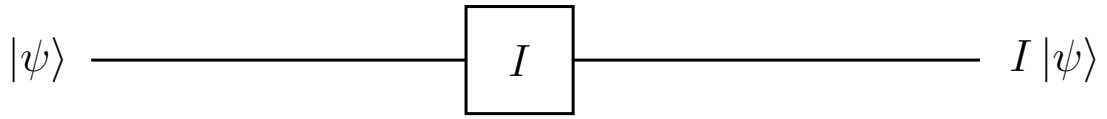


Figure 2.4: Circuit representation of Identity gate.

The identity matrix is,

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{2.4.2}$$

2.4.3 Pauli-X gate

There are three types of Pauli gate, Pauli-X, Pauli-Y, and Pauli-Z. They are one of the most fundamental quantum gates.

Pauli-X gate acts on one qubit quantum state and works similarly to the NOT gate in classical. When we apply the Pauli-X gate to $|0\rangle$, then $X|0\rangle = |1\rangle$. Otherwise,

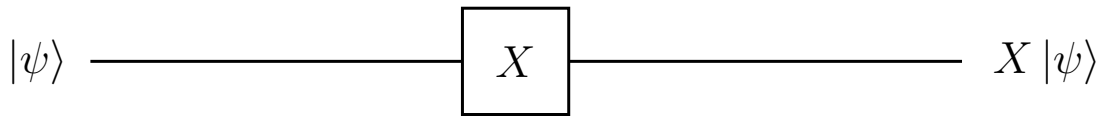


Figure 2.5: Circuit representation of Pauli-X gate.

$X|1\rangle = |0\rangle$. The matrix of Pauli-X gate is,

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{2.4.3}$$

This matrix rotation corresponds to the π rotation on the X-axis in the Bloch sphere.

2.4.4 Pauli-Y gate

Pauli-Y gate also acts on one qubit quantum state. The matrix of the Pauli-Y gate includes an imaginary number i, unlike the classical gate.

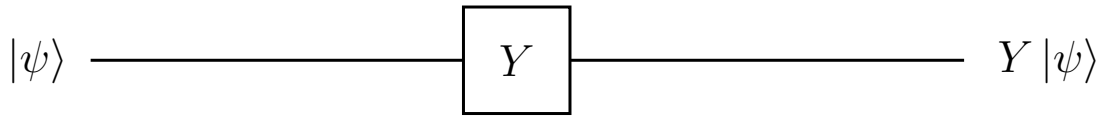


Figure 2.6: Circuit representation of Pauli-Y gate.

The matrix of Pauli-Y gate is,

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}. \tag{2.4.4}$$

This matrix rotation corresponds to the π rotation on the Y-axis in the Bloch sphere.

2.4.5 Pauli-Z gate

Pauli-Z gate is also called the phase rotation gate because it changes the phase of a quantum state. When we apply Pauli-Z gate to $|0\rangle$, then nothing happens $Z|0\rangle = |0\rangle$. However, when we apply Pauli-Z gate to $|1\rangle$, then it becomes $Z|1\rangle = -|1\rangle$.

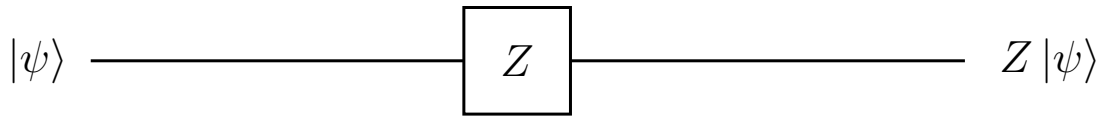


Figure 2.7: Circuit representation of Pauli-Z gate.

The matrix representation of Pauli-Z gate is,

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{2.4.5}$$

.

This matrix rotation corresponds to the π rotation on the Z-axis in the Bloch sphere.

2.4.6 Hadamard gate

Hadamard gate plays an important role in creating a superposition of a qubit. Figure 2.8 shows the circuit representation of the Hadamard gate. If the input state is $|0\rangle$, the state becomes $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$. Otherwise, if the input state is $|1\rangle$, the state becomes $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. Those $|+\rangle$ and $|-\rangle$ are called "superposition" state. See Sec. 2.5 for more details about superposition.

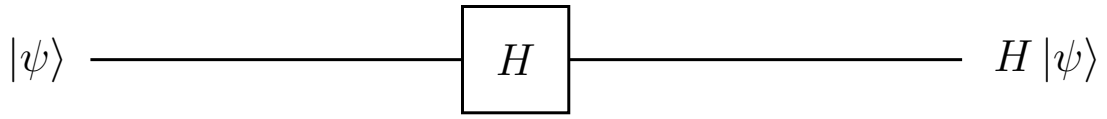


Figure 2.8: Circuit representation of Hadamard gate.

The matrix representation of the Hadamard gate is,

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{2.4.6}$$

The rotation of this matrix corresponds to the π rotation on the $\frac{\pi}{2}$ axis in the Bloch Sphere.

2.4.7 S gate

Following two gates (S gate and T gate) are gates that rotate phase. S gate rotates phase for $\frac{\pi}{2}$. We are also able to describe it as a square root of the Pauli-Z gate $S = \sqrt{Z}$. Thus the matrix representation of the S gate is also the square root of the Pauli-Z gate.

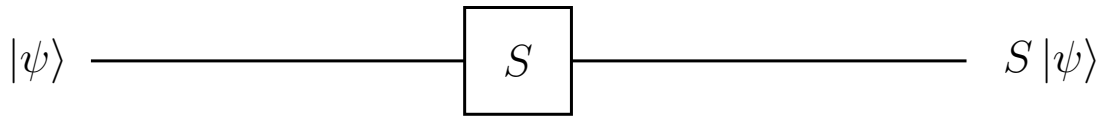


Figure 2.9: Circuit representation of S gate.

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}. \tag{2.4.7}$$

.

2.4.8 T gate

T gate rotates phase $\frac{\pi}{4}$. T gate can be represented as $T = \sqrt{S}$ as well as S gate.

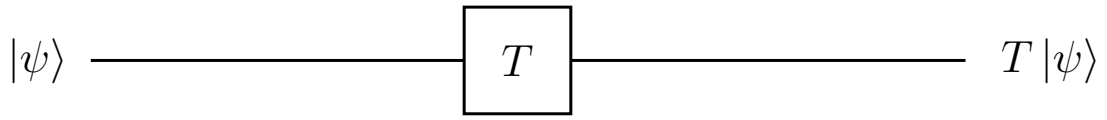


Figure 2.10: Circuit representation of T gate.

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix}. \tag{2.4.8}$$

.

A generalized form of single phase rotation gate is,

$$R_z(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}. \tag{2.4.9}$$

When the rotation angles θ are $\pi, \frac{\pi}{2}, \frac{\pi}{4}$, corresponding phase rotation gates are $R_z(\pi) = Z$, $R_z(\frac{\pi}{2}) = S$, $R_z(\frac{\pi}{4}) = T$ respectively.

2.4.9 Two-qubit gate

Two-qubit gates also play an essential role in creating "Entanglement," a fundamental phenomenon unique to quantum mechanics (See 2.6). Two-qubit gates act on two qubits, control, and target. When the control qubit is $|1\rangle$, a quantum gate works on the target qubit. Otherwise, it doesn't work. Figure 2.11 shows the circuit representation of a two-qubit unitary gate. $CU_{i,j}$ represents the controlled-unitary gate from qubit i to qubit j. The size of a single-qubit gate matrix is two by two unitary matrix, but the two-qubit gate matrix has a four by four unitary matrix.

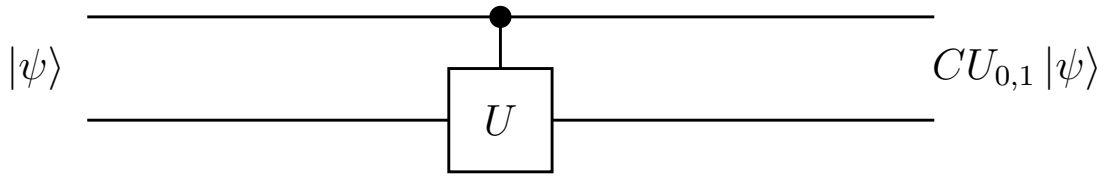


Figure 2.11: Circuit representation of CU gate.

2.4.10 Controlled-NOT (CNOT) gate

Controlled-Not (CNOT, CX) gate is one of the most fundamental two-qubit gates. When the control qubit click, the CNOT gate applies the Pauli-X gate to the target qubit. Both Figure 2.12 and Figure 2.13 shows the circuit representation of CNOT gate. Figure 2.13 is a relatively common way to represent the CNOT gate.

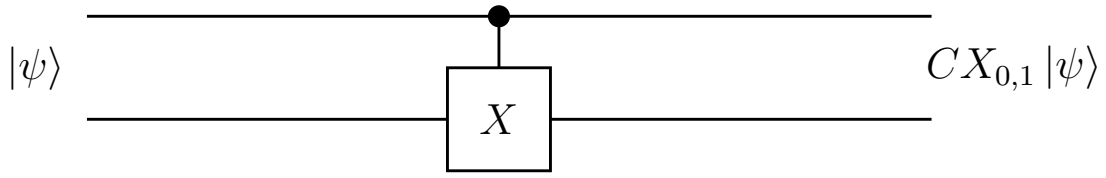


Figure 2.12: Circuit representation of CX gate.

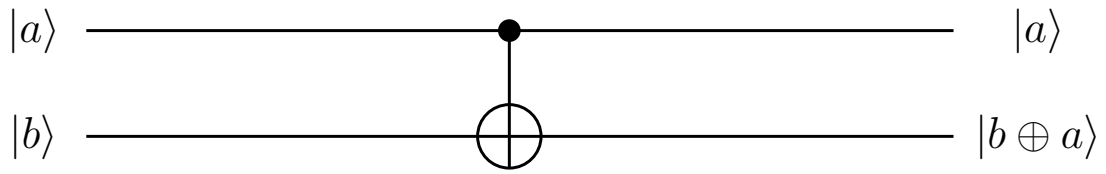


Figure 2.13: Another representation of CX gate. A notation $\otimes$ means XOR operation in classical circuit $(a + b \, mod \, 2)$.

The matrix representation of the CNOT gate is,

$$CNOT = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.4.10}$$

Many algorithms and applications include the CNOT gate.

2.4.11 CZ (C-Phase) gate

Controlled-Z (Controlled-Phase) gate applies the Pauli-Z gate when the control qubit is $|1\rangle$. This gate is used for creating a graph state that is a critical resource for

measurement-based quantum computing. The matrix representation of the CZ gate is,

$$CZ = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes Z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}. \tag{2.4.11}$$

.

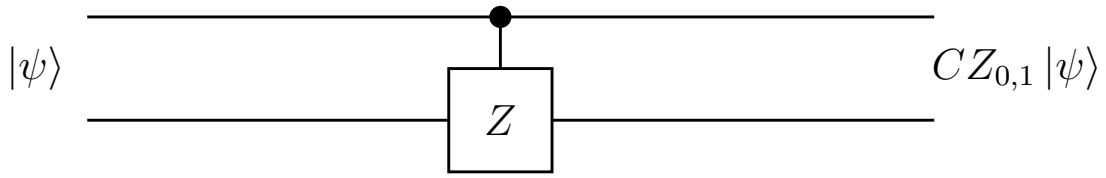


Figure 2.14: Circuit representation of CZ gate.

Clifford group

In general, the CZ gate, Hadamard gate, and S gate are called Clifford gate. Sometimes, the CNOT gate is also included because the CNOT gate is compatible with the CZ gate with the Hadamard gate, as shown in Figure.

Those gates have a particular name because they can be simulated in the classical computer efficiently. This theorem is called Gottesman-Knill's theorem.

2.5 Superposition

Superposition is an essential phenomenon that allows quantum computers to handle multiple data simultaneously. Superposition state includes several states at the same time with specific probability amplitudes. For example, in the single-qubit case, the easiest way to create superposition is to act the Hadamard gate on the $|0\rangle$.

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{2.5.1}$$

In this example, when we measure this single qubit, we can observe $|0\rangle$ with 50% chance ($\left|\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2}$) and $|1\rangle$ with 50% chance as well. As an another example, we are also able to assume a following quantum state.

$$|\psi\rangle = \frac{1}{2}|0\rangle + \sqrt{\frac{3}{4}}|1\rangle \tag{2.5.2}$$

In this case, we can observe $|0\rangle$ in 25% and $|1\rangle$ in 75%. This is also one possible superposition state. More generally, one qubit superposition state can be represented as,

$$|\psi\rangle = \alpha\,|0\rangle + \beta|1\rangle \tag{2.5.3}$$

where $|\alpha|^2 + |\beta|^2 = 1$ same as Eq. 2.2.2.

Note that when we measure a single superposition state, it converges a classical 0 or 1 state. To get a proportion of state before measurement, we have to prepare copies of the circuit, then measure it repeatedly. As statistical information, we could get the probability distribution of all outputs.

2.6 Entanglement

Entanglement plays a crucial role in Quantum information theory. An entangled quantum state can be generated by performing a quantum operation over the two different Hilbert spaces. Suppose we have a quantum state.

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{2.6.1}$$

When we measure this state, we cannot observe $|01\rangle$ and $|10\rangle$. Instead of those, the states we are only able to observe are $|00\rangle$ and $|11\rangle$. We also explain that phenomenon as "each qubit's state doe not change independently." If the measurement result of the first qubit is $|0\rangle$ ($|1\rangle$), then the state of the second qubit is determined as $|0\rangle$ ($|1\rangle$ at the same time, respectively. Important example usage of this entanglement is the quantum teleportation algorithm (See. Sec. 2.8)based on Bell state (See. Sec. 2.7).

For the historical background, the existence of Quantum Entanglement had been denied by physicists at that time. However, in later years, its existence was proven by the violation of Bell's inequality.

2.7 Bell State

As special states in entangled states, we have "Bell State," one of two qubits entangled states.

$$|\phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{2.7.1}$$

$$|\phi^-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}} \tag{2.7.2}$$

$$|\psi^+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}} \tag{2.7.3}$$

$$|\psi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}} \tag{2.7.4}$$

These are four types of bell states. The corresponding quantum circuit to generate the state 2.7.1 is Figure 2.15. Following equations explain the procedure of that circuit.

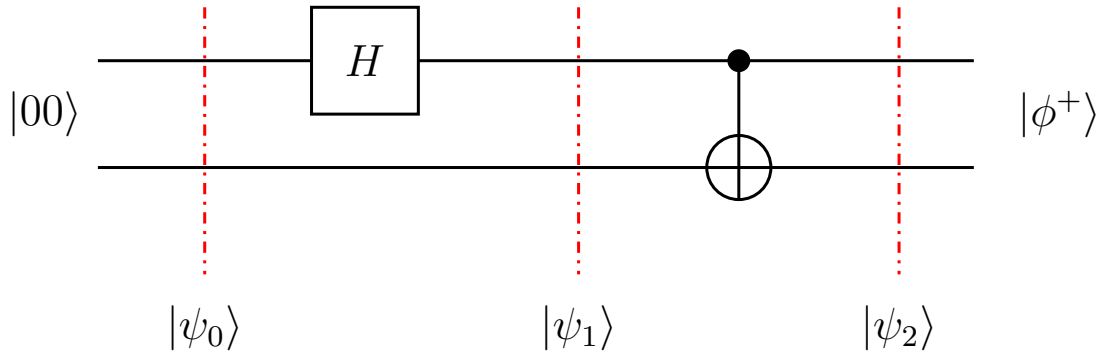


Figure 2.15: Quantum Circuit to create $|\phi^+\rangle$

First, prepare two qubits and initialize them.

$$|\psi_0\rangle = |00\rangle \tag{2.7.5}$$

Next, apply the Hadamard gate to the first qubit.

$$|\psi_1\rangle = H\,|\psi_0\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)|0\rangle = \left(\frac{|00\rangle + |10\rangle}{\sqrt{2}}\right) \tag{2.7.6}$$

Finally, apply the CX gate from the first qubit to the second qubit.

$$|\psi_2\rangle = CX\,|\psi_1\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{2.7.7}$$

2.8 Quantum Teleportation

Quantum Teleportation is one prominent application using entanglement, especially Bell pairs [11]. Ultimately, we can transfer information from one place to another, no matter how far away between them. This technology is used for quantum communication. The teleportation procedure is following. Figure 2.16 shows the quantum circuit for the quantum teleportation protocol to send one qubit information.

First, there are two people Alice and Bob, connected by a quantum link. Alice has two qubits, and Bob has one qubit. Initially, these qubits are all $|0\rangle$ state.

$$|\psi_0\rangle = |00\rangle_A\,|0\rangle_B \tag{2.8.1}$$

The subtext represents which person has which qubits (A is Alice and B is Bob).

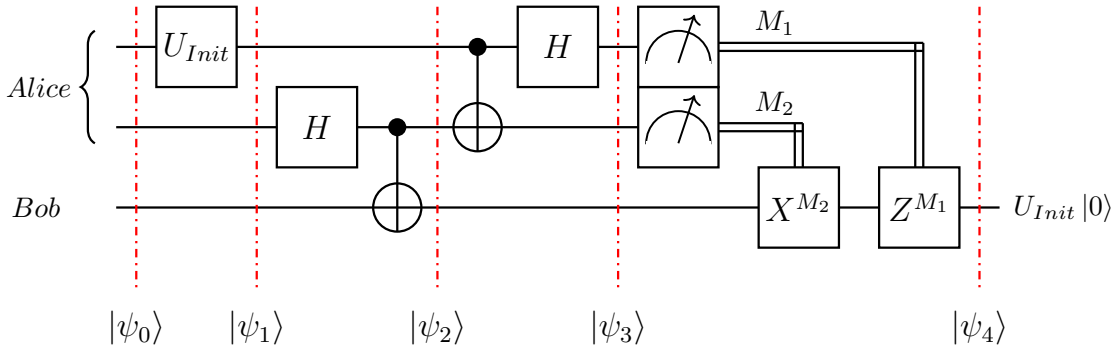


Figure 2.16: Procedure of Quantum Teleportation protocol. *Init* represents arbitrary one qubit gate to encode data into a quantum state.

Second, Alice encodes the information she wants to send into the first qubit. We can use any unitary quantum operation at this initialization. Here, we define that unitary operation as U_{Init}.

$$|\psi_1\rangle = U_{Init}\,|\psi_0\rangle \tag{2.8.2}$$

As a next step, Alice and Bob share an entangled pair of qubits. In this example, we used $|\phi^+\rangle$ state. They create an entanglement in the second and the third qubit in the circuit diagram. In practice, an intermediate node creates an entangled pair of qubits, then distributes one qubit to Alice and the other qubit to Bob.

$$|\psi_2\rangle = U_{Init}\,|0\rangle_{A_0} \otimes \left(\frac{|0\rangle_{A_1}|0\rangle_B + |1\rangle_{A_1}|1\rangle_B}{\sqrt{2}}\right) \tag{2.8.3}$$

After they successfully share an entangled pair of qubits, then Alice generates entanglement between her data qubit (the first qubit) and the shared qubit between Bob. Let $U_{Init}\,|0\rangle$ as $U_{Init}\,|0\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$.

$$\begin{aligned}
|\psi_3\rangle &= H_{A_0} CX_{A_0,A_1}(\alpha\,|0\rangle_{A_0} + \beta\,|1\rangle_{A_1}) \left(\frac{|0\rangle_{A_1}|0\rangle_B + |1\rangle_{A_1}|1\rangle_B}{\sqrt{2}}\right) \\
&= H_{A_0}\frac{1}{\sqrt{2}}\left[\alpha\,|0\rangle\,(|00\rangle + |11\rangle) + \beta\,|1\rangle\,(|10\rangle + |01\rangle)\right] \\
&= \frac{1}{2}\left[\alpha(|000\rangle + |011\rangle + |100\rangle + |111\rangle) + \beta(|010\rangle + |001\rangle - |110\rangle - |101\rangle)\right] \\
&= \frac{1}{2}\left[|00\rangle\,(\alpha\,|0\rangle + \beta\,|1\rangle) + |01\rangle\,(\alpha\,|1\rangle + \beta\,|0\rangle) + |10\rangle\,(\alpha\,|0\rangle - \beta\,|1\rangle) + |11\rangle\,(\alpha\,|1\rangle - \beta\,|0\rangle)\right]
\end{aligned} \tag{2.8.4}$$

As a final step, Alice and Bob measure their qubits and apply Pauli operations based on the measurement results. According to 2.8.4, we can estimate the third

qubit's measurement result from that of the first and the second qubit. There are four possible outcomes when we measure the first and second qubits in the Z basis.

$$|0\rangle_{A_0} |0\rangle_{A_1} \to \alpha |0\rangle_B + \beta |1\rangle_B = U_{Init} |0\rangle \tag{2.8.5}$$
$$|0\rangle_{A_0} |1\rangle_{A_1} \to \alpha |1\rangle_B + \beta |0\rangle_B = X \cdot U_{Init} |0\rangle \tag{2.8.6}$$
$$|1\rangle_{A_0} |0\rangle_{A_1} \to \alpha |0\rangle_B - \beta |1\rangle_B = Z \cdot U_{Init} |0\rangle \tag{2.8.7}$$
$$|1\rangle_{A_0} |1\rangle_{A_1} \to \alpha |1\rangle_B - \beta |0\rangle_B = Z \cdot X \cdot U_{Init} |0\rangle \tag{2.8.8}$$

From the above, we can see all possible final states include the state $U_{Init} |0\rangle$. Bob has to extract that information from the final state by applying proper Pauli operation(s). Note that in this process, Alice has to tell the measurement result. That means they can't communicate beyond the speed of light.

$$00 \to I \cdot U_{Init} |0\rangle = U_{Init} |0\rangle \tag{2.8.9}$$
$$01 \to X \cdot X \cdot U_{Init} |0\rangle = U_{Init} |0\rangle \tag{2.8.10}$$
$$10 \to Z \cdot Z \cdot U_{Init} |0\rangle = U_{Init} |0\rangle \tag{2.8.11}$$
$$11 \to X \cdot Z \cdot Z \cdot X \cdot U_{Init} |0\rangle = U_{Init} |0\rangle \tag{2.8.12}$$

Finally, Bob successfully extracts the information Alice sent. As I mentioned in the first part of this section, we can transfer one qubit data from one place to another as long as they share an entangled state.

2.9 Stabilizer Formalism

A stabilizer is one powerful tool to describe a specific type of quantum state. By using stabilizers, large quantum states can easily be handled.

Let $|\psi\rangle$ be a pure state. When a unitary operation U satisfies $U |\psi\rangle = |\psi\rangle$, U is a stabilizer of $|\psi\rangle$. In other words, when $|\psi\rangle$ is an eigenvector of U with eigen value 1, then U is a stabilizer of $|\psi\rangle$.

As the simplest example of a stabilizer, Z gate stabilize $|0\rangle$.

$$Z |0\rangle = |0\rangle , -Z |1\rangle = |1\rangle \tag{2.9.1}$$

This formalization can be extended to any number of qubits, and it is a useful tool for constructing error-correcting codes for a large number of qubits. A quantum circuit that is only composed of Clifford gates is called a stabilizer circuit. The stabilizer circuits can be simulated with stabilizer formalism in a classical computer within polynomial time complexity [12, 13, 14]. This theory is known as Gottesman and Knill's theorem proposed by Daniel Gottesman and Emanuel Knill.

2.10 Graph state

Graph state is a highly entangled state among several or more qubits. Graph state is also called "Resource state." This state plays a crucial role in the Measurement-Based Quantum Computing, one alternative quantum computational model rather than the circuit model.

It is also an essential resource for blind quantum computing [15, 16, 17] and secret sharing [18, 19].

2.10.1 Graph Theory

First, we define a graph $G = (V, E)$ with a series of vertices and edges. This graph has $|V|$ vertices. Each vertex has an index i and ith vertex can be represented by v_i. If v_i and v_j are connected by an edge, then we can say the conbination of them is included in the edges E ($\{v_i, v_j\} \in E$).

One more way to describe a graph is using the "Adjacency Matrix." The adjacency matrix is N by N matrix that only includes 0 or 1 as an element. Each column and row corresponds to the index of the graph. This is an adjacency matrix of a graph in Figure 2.17.

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}. \tag{2.10.1}$$

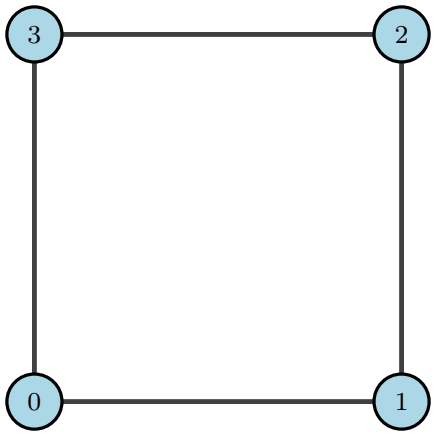


Figure 2.17: An example of graph with only four vertices.

In a general way, an adjacency matrix A is represented as,

$$A_{i,j} = \begin{cases} 1\,(\{v_i, v_j\} \in E) \\ 0\,(otherwise). \end{cases} \tag{2.10.2}$$

When the graph is undirected, the adjacency matrix is symmetric ($A = A^T$).

2.10.2 Representation of Graph State

Let $|G\rangle$ be a graph state. The graph state can be represented as follows.

$$|G\rangle = \prod_{\{i,j\} \in E} CZ_{i,j} \, |+\rangle^{|V|} \tag{2.10.3}$$

where $|+\rangle^{|V|}$ represents the number of qubits is $|V|$ and all qubits are superposition state. We need as much qubit as the number of vertices in the graph. Each vertex corresponds to each qubit. To generate entanglement between vertices, we apply CZ gates corresponding to edges in the graph.

More conveniently, we have another notation to describe a graph state. That is a stabilizer. Let K_a be a correlation operator at vertex $a \in V$.

$$K_a = \sigma_x^a \sigma_z^{N_a} := \sigma_x^a \prod_{b \in N_a} \sigma_z^b \tag{2.10.4}$$

Eigenvalues to the correlation operators are all $+1$ for all a. The subgroup S of the local Pauli-group P^V generated by the set of correlation operators is called the graph state's *stabilizer*.

2.11 Measurement-Based Quantum Computing

Measurement-Based Quantum Computing (MBQC) is an alternative way of quantum computing. By only measuring qubits one by one, an arbitrary quantum circuit can be executed [20, 21].

As the first step of MBQC, it requires a resource state (so-called graph state). In the MBQC scheme, we usually use a two-dimensional graph state. Figure 2.18 shows how the MBQC works. The measurement operations go from left to right. Z measurement destroys the coherence, and the qubit is removed from the graph state. To achieve two or more qubit operations, entanglements between several rows must remain.

Based on the previous measurement results, the observer decides which basis the next qubit is measured. After all of the qubits are measured, the observer corrects the final state with a series of operations.

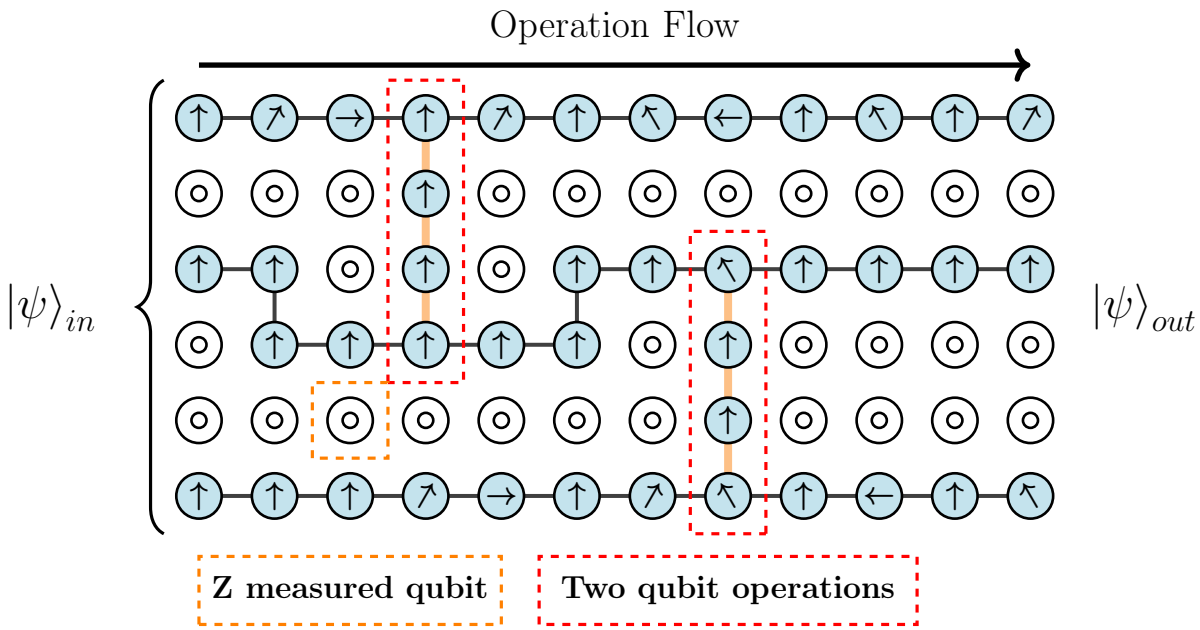


Figure 2.18: Process of Measurement-Based Quantum Computing

MBQC is known as a universal quantum computation model. That means any quantum circuits can be represented in the MBQC model.

An example of a quantum operation in MBQC is how the Hadamard gate works in the MBQC scheme. Suppose there is a quantum state $|\psi\rangle_1 = a|0\rangle + b|1\rangle$. Add one more qubit as a $|+\rangle$. By applying a CZ gate between them,

$$CZ(|\psi\rangle_1 |+\rangle) = a|0\rangle|+\rangle + b|1\rangle|-\rangle . \tag{2.11.1}$$

Projecting the first qubit to $|0\rangle + e^{i\theta}|1\rangle$, the second qubits become,

$$a|+\rangle_2 + be^{-i\theta}|-\rangle_2 = He^{i\theta Z/2}|\psi\rangle . \tag{2.11.2}$$

This results applying the Hadamard gate to the original state. If the first state was projected onto the other state, such as $|0\rangle - e^{i\theta}|1\rangle$, the second qubit becomes,

$$a|+\rangle_2 - be^{-i\theta}|1\rangle_2 = XHe^{i\theta Z/2}|\psi\rangle . \tag{2.11.3}$$

In this case, additional X operation is applied to the first state $|\psi\rangle$. This X must be removed at the final state.

2.12 Quantum State Verification

Quantum state verification is a statistical approach to verify the results of quantum computing and quantum networking. In the quantum networking scheme, multiple

quantum nodes are working together. Under that circumstance, each quantum computer has to verify incoming and outgoing states to trust other quantum computers. Many protocols have been proposed in the last decades.

Ultimately, quantum tomography can check if a quantum state is completely the same as the ideal state. The quantum tomography protocol can estimate an unknown quantum state with plenty of measurement. However, it requires substantial quantum resources that are performed measurement. Generally speaking, quantum tomography requires exponential quantum resources.

As alternative ways to verify a quantum state, there are several verification approaches with a reasonable amount of quantum resources [22, 23, 24, 25]. More specifically, state verification for a graph state can be done in efficient ways [26, 27].

2.13 Quantum Errors

"Error" or "Noise" in quantum computing is one dominant problem to solve for practical usage. A quantum state is fragile to interaction with the outer world. Quantum noise affects the result of quantum computation. In the worst case, there are only random output comes out as a result.

There are many types of errors on the quantum computer. One prominent noise on the NISQ device is a gate error, in which a quantum computer acts quantum gates on a qubit. Other than that, there are many types of errors such as measurement error, T1 and T2 error, photon loss error.

As an example of a quantum error in one qubit system, assume a mixed state ρ_2 of $|0\rangle$ and $|1\rangle$. The target state assumed to be a measurement output is $|0\rangle$.

$$\rho_2 = p\,|0\rangle\langle 0| + (1-p)\,|1\rangle\langle 1| = \begin{pmatrix} p & 0 \\ 0 & (1-p) \end{pmatrix} \tag{2.13.1}$$

where p is a proportion of $|0\rangle$. When $p = \frac{1}{2}$, that state is called "completely mixed" state. In general, N qubit mixed state is represented as,

$$\rho_N = \sum_i^N p_i\,|\psi_i\rangle\langle\psi_i| \tag{2.13.2}$$

where p_i is a proportion of i th state. When all $p_i = \frac{1}{N}$, it is N qubit mixed state.

Quantum Error Correction (QEC) allows us to overcome those noises and make quantum computers a practical computer. [28]

2.14 Quantum Internet and Networking

The quantum Internet [29, 30] is, along with the universal quantum computer [31], one of the “killer applications” of quantum technologies.

One of the primary challenges is robust and coherent communication between distant quantum computers in the network. Classical network nodes communicate via strong light signals that propagate through optical fibers and become attenuated in the process. These signal attenuation can be relatively easily mitigated by inspecting the classical signal and amplifying it.

Quantum nodes communicate via an exchange of much weaker signals, which are often at the single-photon level. Photon loss in optical fibers becomes an extremely pertinent problem as well. Furthermore, in order for the whole network to preserve its quantum functionality, the individual signal photons must retain their coherent properties. Propagating photons lose these coherent properties due to decoherence. Direct amplification of these weak signals becomes undesirable because of the low signal-to-noise ratio of the boosted signal [32]. Unlike in the classical case, it is impossible to copy and resend these photons due to the quantum no-cloning theorem [33, 34, 35, 36], which states that it is forbidden to clone an unknown quantum state deterministically.

Quantum repeaters are designed to overcome all of these problems [37, 38, 30, 39]. Rather than boosting the signal directly or attempting to copy it, they work on the principle of *entanglement swapping* [40, 41]. Consider a scenario where two nodes of a quantum network, node A, and node B, wish to establish an entangled pair of qubits. Sending one of the qubits of the entangled pair from A to B would result in a final state with low fidelity that would not be suitable for further information processing tasks. A better solution is to introduce an intermediate repeater node C that shares a maximally entangled state with node A, e.g. $|\Phi^+\rangle_{A,C1} = (|00\rangle+|11\rangle)/\sqrt{2}$, and another maximally entangled state with node B, $|\Phi^+\rangle_{C2,B}$. Repeater node C then measures its two qubits on a Bell basis, which transfers the entanglement to nodes A and B. Depending on the outcome of the measurement, which is communicated classically to A and B, the end nodes apply local correction operations in order to establish a maximally entangled state $|\Phi^+\rangle_{A,B}$. Traditionally, quantum repeaters have been designed to rely on quantum memories in order to store the qubits before they are measured in the Bell basis [42, 43]. Recent developments have shown that this is not a fundamental requirement and that all-photonic quantum repeaters [44, 45, 46, 47, 48] offer an attractive memory-less alternative.

2.15 Graph Algorithms

The graph has plenty of applications, such as community detections, the structure of proteins, navigations on the map. Generally speaking, graph data is large and difficult to handle. Thus there have been many algorithms developed to make graph processing efficient. In this project, three graph processing algorithms were applied to the initial graph state.

2.15.1 Stoer Wagner Algorithm

The Store Wagner algorithm was proposed by Mechthild Stoer and Frank Wagner in 1997 [49]. This algorithm aims to find the set of edges to cut a weighted graph with minimum weights. When the target graph is not a weighted graph, it finds the minimum number of edges to cut the graph into two parts.

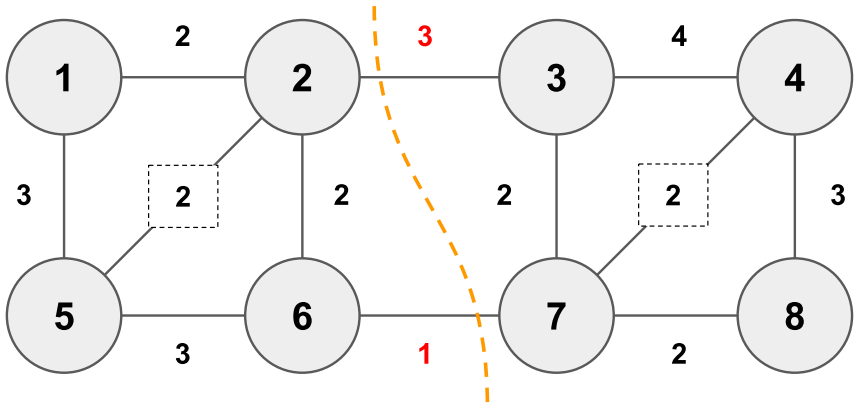


Figure 2.19: The stoer-wagner finds minimum weight cut in a graph.

Let $G = (V, E, w)$ be a weighted and undirected graph with a series of vertices. It has a series of edges with some weights. The procedure of this algorithm can be decomposed into two subroutines. One is a minimum cut phase subroutine shown in Algorithm 1. What this subroutine does here is finding a minimum cut for cutting

Algorithm 1 Minimum Cut Phase Subroutine

Input: G, w, a

a can be an any vertex in V

1: $A = \{a\}$, $(a \in V)$
2: **while** $A \neq V$ **do**
3: Add the most tightly connected vertex to A.
4: Store the cut-of-the-phase and shrink G by merging the two vertices added last
5: **end while**

out one vertex from an entire graph. "The most tightly connected vertex" is a vertex with more weight than the other vertices, which is one hop distance from the list of

vertices A, but not included A. An example of the most tightly connected vertex is shown in Figure 2.20.

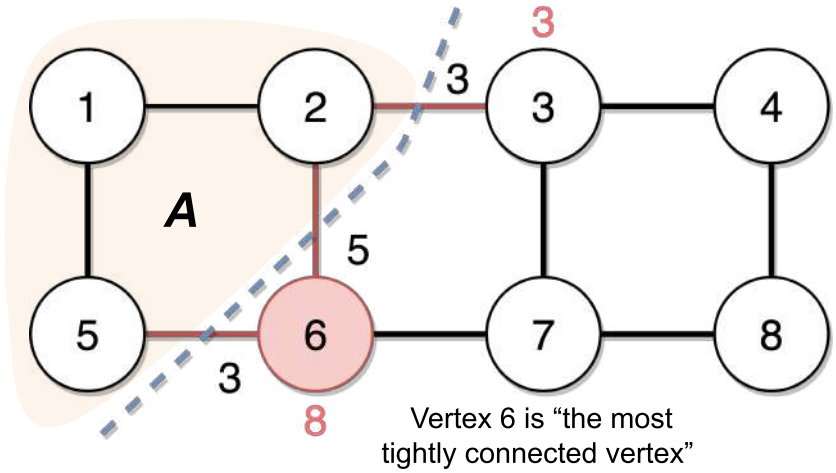


Figure 2.20: An example of "the most tightly connected vertex". Suppose $w(\{v_i, v_j\})$ is the weight on the edge $\{v_i, v_j\}$. Vertices $1, 2, 5$ are already stored in searched list A. The weights on the edges $w(\{2, 3\}) = 3$, $w(\{5, 6\}) = 3$, $w(\{2, 6\}) = 5$. The node 6 has a weight $5 + 3 = 8$ and node 3 has a weight 3 in total. In this case, the node 6 is the most connected vertex in this graph.

The second subroutine finds a global minimum cut of a graph G with the subroutine Algorithm 1.

Algorithm 2 Minimum Cut Subroutine

Input: G, w, a

```
1: while |V| > 1 do
2:    MinimumCutPhase(G, w, a) Shown in Algorithm 1
3:    if the cut-of-the-phase is lighter than the current minimum cut then
4:       store the cut-of-the-phase as the current minimum cut
5:    end if
6: end while
```

These pseudo-code are quoted from [49, 50]. The time complexity of this algorithm is $O(|V|\,|E| + |V|^2 \log |V|)$ where $|V|$ denotes the number of vertices and $|E|$ denotes the number of edges in a graph.

2.15.2 Kernighan Lin Algorithm

The Kernighan Lin algorithm was proposed by B. W. Kernighan and S. Lin in 1970 [51]. This algorithm is used for designing classical circuits and components in VLSI[52]. This algorithm allows us to partition a graph into two parts with a balanced number of vertices. This proportion can be extended to an arbitrary ratio. When this algorithm partitions a graph, it tries to find the minimum edge cut under that proportion.

The time complexity of this algorithm is $O(|V|^2 \log |V|)$.

Algorithm 3 Kernighan Lin Algorithm from [53]

Input: $G(V, E)$

1: Divide G into two subgraphs A, B of equal size arbitrarily.
2: **while** There are vertices **do**
3: Select $a_i \in A$, $b_i \in B$ such that reducing the cost (the sum of weights of partitions).
4: Swap a_i, b_i
5: Let C_i be the cost of partition after swapping a_i, b_i
6: **end while**
7: Return (A', B') corresponding to the smallest C'.

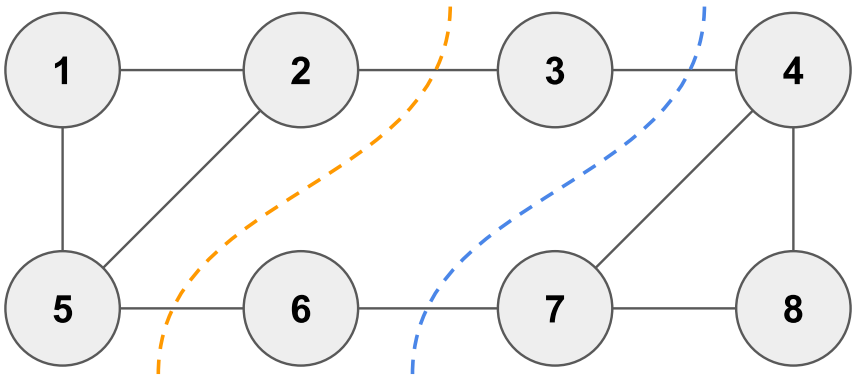


Figure 2.21: The stoer-wagner finds minimum weight cut in a graph.

2.15.3 Densest-k subgraph search Algorithm

The densest-k subgraph problem is the problem that finds the densest-k vertices subgraph induced by a graph. This problem is known as an NP-hard problem. There are several algorithms to solve this problem with approximations. However, no approximation exists within the multiplicative error. The approach this paper use is [54].

Chapter 3

Problem Definition and Proposal

Distributed Quantum Computing (DQC) is one promising platform in quantum computing. In the DQC scheme, quantum processors are connected with quantum internet and working together. This scheme allows quantum computers to enhance their computational power [55, 56, 57]. Several architectures for distributed quantum computing have been proposed in the different quantum systems [56, 58, 59, 60, 61, 62].

Measurement-based quantum computing with graph states is one candidate for distributed quantum computing -sharing a large graph state over the distributed systems and measuring qubits one by one to run a quantum circuit [63]. Generally speaking, the size of the graph state corresponds to the size of the quantum circuit to be embedded.

However, there are many technical difficulties to be solved. Unlike classical distributed systems, there are some constraints derived from the No-cloning theorem and Entanglement. In the classical system, data and programs can be copied without any considerations. However, no one can copy an unknown quantum state because of the no-cloning theorem. In addition to that, a quantum state is so fragile that generating remote entanglements is one of the challenging tasks in the quantum networking field.

There are several approaches to create a graph state efficiently. [64, 65] show efficient ways of preparing graph states on the local devices. In these papers, the main aim is to make a graph state with fewer operations on the quantum circuit. Things that need caring in that scheme are the quality of qubits in a local device. However, in the DQC protocol, quantum link properties have to be taken into account.

To make the DQC protocol possible, controlling resources over the distributed systems is an important task. There are several related works for distributing graph states. In [66, 67], authors showed efficient distribution way of graph states in the ideal situation. Their approaches are based on local complementations that transform one graph state into the other LC equivalent graph state with Pauli measurements.

As a problem definition, the problem to solve in this project is how to allocate resources over noisy quantum networks efficiently. This project focuses on a graph state that is one particular quantum state used in Measurement-Based Quantum

Computing. Figure 3.1 is a simple expression of this problem definition. We have several quantum computers in the same network or even outside of the network. However, each quantum computer and quantum link have different properties such as error rate, qubit capacity. This project aims to establish a scalable protocol that deals with those processes taking device properties into considerations.

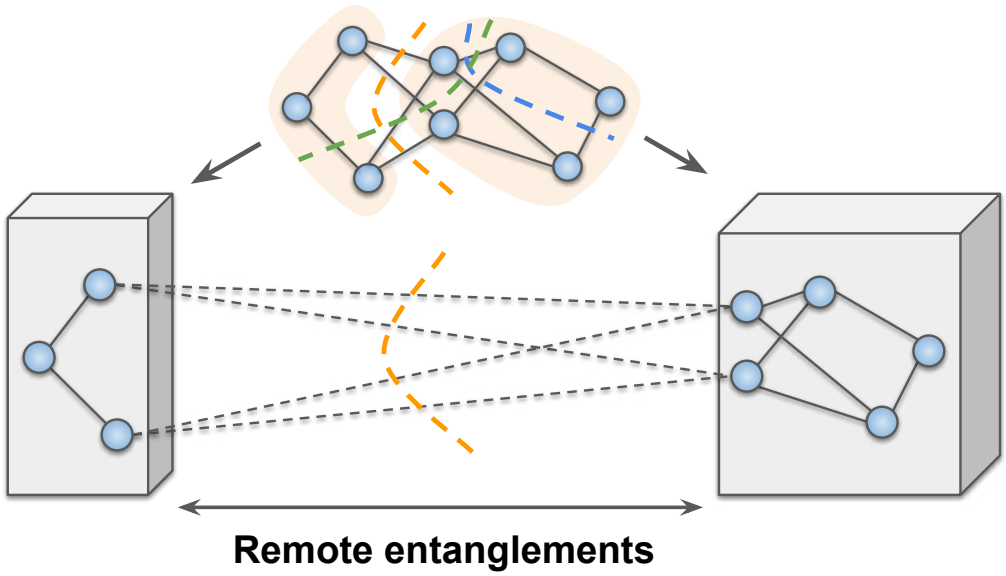


Figure 3.1: This diagram is a simple explanation of the problem definition. A graph at the top of this picture represents the final graph state we want to create. There are two devices connected with a noisy quantum link. In general, different quantum computers and links have different error rates. Considering those error properties, we have to assign qubits properly to maximize the entire performance.

The proposal is composed of two steps. The first step is the classical communication step to share the device information. The second step is the quantum computation step that operates quantum gates on the device and over the quantum links.

Figure 3.2 shows an example quantum network topology. There are two types of quantum computers. One is called initiator that starts the entire process of this algorithm. The other is called responder that responds to the initiator's request. They are connected with both quantum and classical links.

First of all, the initiator decides the size and shape of the graph state. It depends on what application it needs to run on that. For example, measurement-based quantum computing requires a two-dimensional lattice graph state. Here, let $G = (V, E)$ be the target state's graph with a series of vertices V and edges E. The total number of qubits required is $N = |V|$.

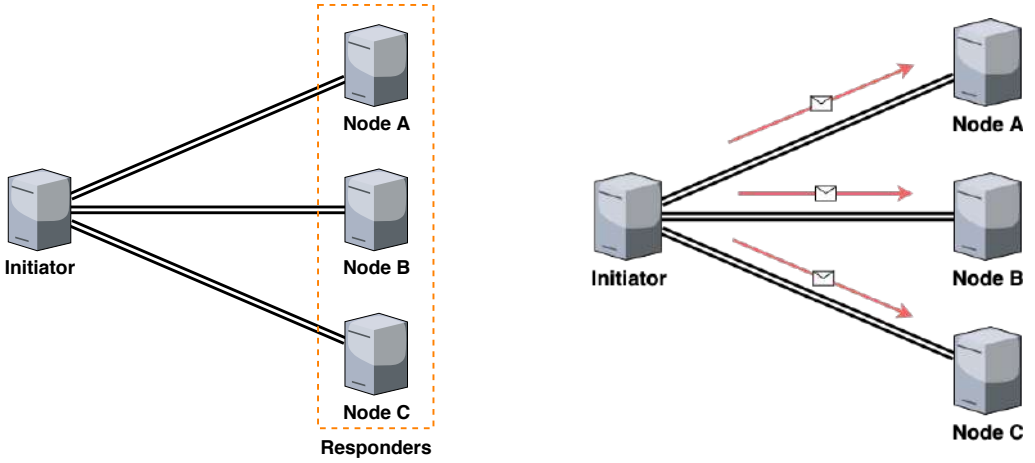


Figure 3.2: This is an example of network topology. The left device is *initiator* that sends requests to responders on the right-hand side.

First, the initiator sends requests to one-hop neighbor nodes to get neighbor information, as shown in Figure 3.2. In this process, the initiator requests the number of qubits available for this protocol, the average error rate on those qubits' error rates, and average errors on the quantum link between them using benchmarking methods. Note that each responder has to know link errors beforehand at this protocol. As types of errors, this protocol is only assuming bit-flip error and phase-flip error. However, other types of error can be used as long as it's quantifiable.

Second, the responders return responses to the initiator with attaching device information. At this point, if a responder is busy with other tasks or not working, it returns status as busy, and it's eliminated from the following processes. If a responder device is available for this protocol, it returns how many qubits it can provide and an average error rate of qubits and links. Note that, in practice, the initiator and responders must trust each other because responders can give the initiator wrong information. It causes spoiling the entire process.

Finally, the initiator divides workloads based on those responses. As the first step of this process, the initiator sums up the available number of qubits in total. If this value is not enough for the required number of qubits, the initiator tries to send the request again after some time durations. In this process, the initiator uses a decision tree shown in Figure 3.8. There are three policies to follow when the initiator divide graph states into several parts 3.8.

Policy 1: Use better devices with lower link error rates This policy prioritizes responders. A responder that has lower error qubits is assigned a relatively large graph state. Let $\boldsymbol{s}$ be the possible combination for dividing workloads. The priority for each setting is calculated with the cost $C_{i\in\boldsymbol{s}}$,

$$C_i = \alpha \frac{\sum_t^{N_i} \varepsilon_t}{N_i} + \beta \varepsilon_{i,j} \tag{3.0.1}$$

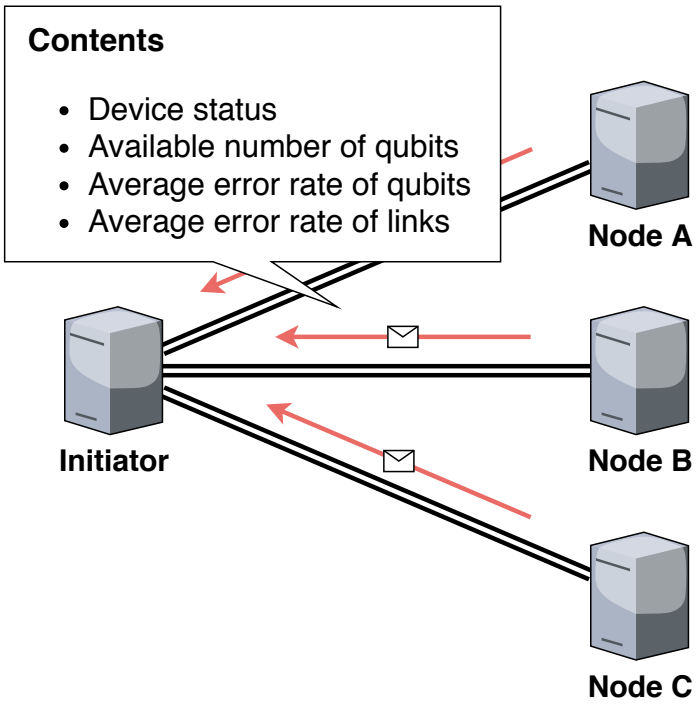


Figure 3.3: Send Response

where $\varepsilon_{i,j}$ is the average link error given by link tomography and α, β are constant value for device error and link error. Note that the initiator can choose what kind of error should be cared for in the allocation process. That means arbitrary error properties can be included as long as they are measurable.

The initiator generates local topologies based on available devices and picks up the best topology from possible local topologies as shown in 3.4. Note that this is just an estimation of the quality of the final state. This cost function doesn't include any aspects of the number of qubits. That means the allocation could be biased, and redundant remote entanglements could appear. To avoid this, the initiator needs to do post-evaluation to verify if this allocation is the best in the possible allocations.

Policy 2: Fewer remote entanglements This policy aims to reduce the number of remote entanglements over a quantum link. Generally speaking, maintaining entanglements over the remote devices is much harder than that of local entanglements. That is because the error rate of the quantum link is much higher, and qubits could be lost while transferring. To reduce the number of remote entanglements, this allocator takes two graph algorithms. One is the Stoer-Wagner algorithm, and the other is the Kernighan-Lin algorithm.

The Stoer-Wagner algorithm finds the minimum weight of edges that divide a graph into two separated subgraphs. When the target graph is not a weighted graph and undirected, the minimum weight directly corresponds to the minimum number of edges. The allocation process with the Stoer-Wagner algorithm is shown in Al-

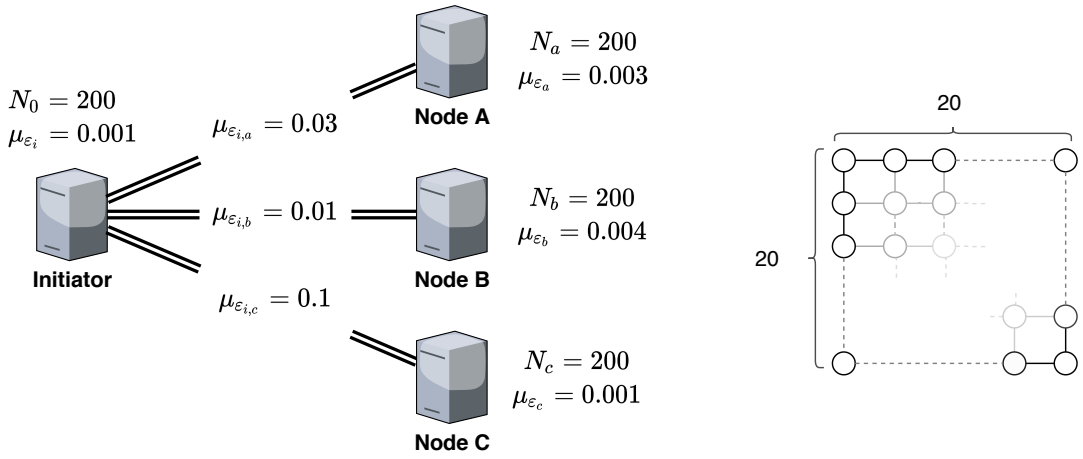


Figure 3.4: This is an example cost calculation. Suppose the initiator tries to generate a 20 by 20 lattice graph state. The initiator requires 400 qubits in total. At this point, no device can handle 400 qubits only in their local processors. Let $\{i\}$ be the initiator device and $\{a, b, c\}$ be the responders. All possible device combinations are $\{\{i\}, \{a\}, \{b\}, \{c\}, \{i, a\}, \{i, b\}, \{i, c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{i, a, b\}, \{i, a, c\}, \{i, b, c\}, \{a, b, c\}, \{i, a, b, c\}\}$. However, several situations can be eliminated because of the number of qubits and connectivity of devices. In this case, the possible situations that needs considering are $\{\{i, a\}, \{i, b\}, \{i, c\}, \{i, a, b\}, \{i, a, c\}, \{i, b, c\}, \{i, a, b, c\}\}$. Costs for these settings with $\alpha = 1, \beta = 1$ are $\{0.034, 0.015, 0.102, 0.048, 0.135, 0.116, 0.149\}$. From this, a combination $\{i, b\}$ is picked up first and $\{i, a\}$ second.

gorithm 4. Let $G = (V, E)$ be a target graph and $\mathbf{L} = (d_0, d_1, ..., d_{n_l})$ be quantum devices in a local topology chosen by the allocator. n_l is the number of devices in the chosen local topology. The output of this procedure is a series of subgraphs for each device $\mathbf{S} = (G'_1, G'_2, ..., G'_{n_l})$, and the cost corresponding to that setting. For convenience, Let G' be a sub-graph for the next round.

First, the allocator cuts the target graph G into two induced subgraphs G'_1, G'_2 and allocates the first half of the subgraph G'_1 to the first quantum node.

Second, the allocator cuts the left half of the subgraph again and assigns the first half. The allocator repeats this process in a greedy manner shown in Figure 3.5.

By repeating this process for all possible local topology, the allocator prepares all possible settings. After subgraphs are prepared, the allocator calculates the costs for those subgraphs again. This time, the allocator calculates the cost based on the subgraph's density and the number of links between them. Let j be the neighbor quantum nodes of i th device. The cost function to calculate the final cost (C_f) is,

$$C_f(\mathbf{S}, \mathbf{L}) = \gamma \sum_{l \in L} D(S_l) \cdot \varepsilon_l + \delta \sum_{\{l,m\} \in L, m \in neig(l)} x_{\{l,m\}} \cdot \varepsilon_{\{l,m\}} \tag{3.0.2}$$

where $D(S_l)$ is the density of subgraph S_l generated in device l and $x_{\{l,m\}}$ is the

the number of links between l th and m th subgraphs. $\varepsilon_l, \varepsilon_{\{l,m\}}$ are average error rates of device and link respectively, and γ and δ are constants. This algorithm can find a minimum cut in polynomial time, but this minimum cut is sometimes not a good choice. That is because the size of the subgraph can exceed the capacity of the device. For these reasons, the second option is the Kernighan-Lin graph partitioning algorithm.

Algorithm 4 Graph allocation based on Stoer-Wagner algorithm

Input: G, $\mathbf{L}$ *Target graph* G *and local topology* $\mathbf{L}$
Output: $\mathbf{S}, C$ *List of subgraphs* $\mathbf{S}$ *and cost* C

Initialisation :
$G' = G$, $\mathbf{S} = \{\}$
Loop for devices in local topology L :
for $i = 1$ to n_l **do**
  $G'_1, G'_2 =$ Stoer-Wagner(G')
  Add as the i-th subgraph
  $\mathbf{S}[i] = G'_1$
  $G' = G'_2$
end for
Cost function shown in Eq.3.0.2
C = Cost($\mathbf{S}$, $\mathbf{L}$)
return $\mathbf{S}$, C

Kernighan-Lin graph partition algorithm aims to partition a graph into two subgraphs in specific proportion. Initially, it decomposes a graph into balanced two subgraphs in terms of the number of vertices. However, it can be extended to an arbitrary ratio. This algorithm takes an initial partition that tells the allocator the number of qubits in output subgraphs in this protocol.

The process of this allocation is shown in 5. Input arguments $G, \mathbf{L}$ are the same as the previous allocation. In this allocation, the allocator compares the cases with different partitions. Let N_i be the number of qubits in the device i, and P_i be the partition that tells the allocator the number of vertices in one subgraph. The allocator searches for several possible cases within the number of qubits in the device by incrementing the number of qubits assigned for the first device.

First, the allocator starts searching for the best allocation with a different number of qubits. As an example, Figure 3.6 shows the procedure of Kernighan Lin allocation. The allocator searches for several possible cases within the number of qubits in the device by incrementing the number of qubits assigned for the first device.

Policy 3: Proper proportion of densities The final policy is about the density of the subgraphs. When the subgraph is dense, the number of operations gets large. This policy can be effective when the link's error rate is reasonably low, and the device error is dominant. As an example of this situation, suppose a high-quality device and

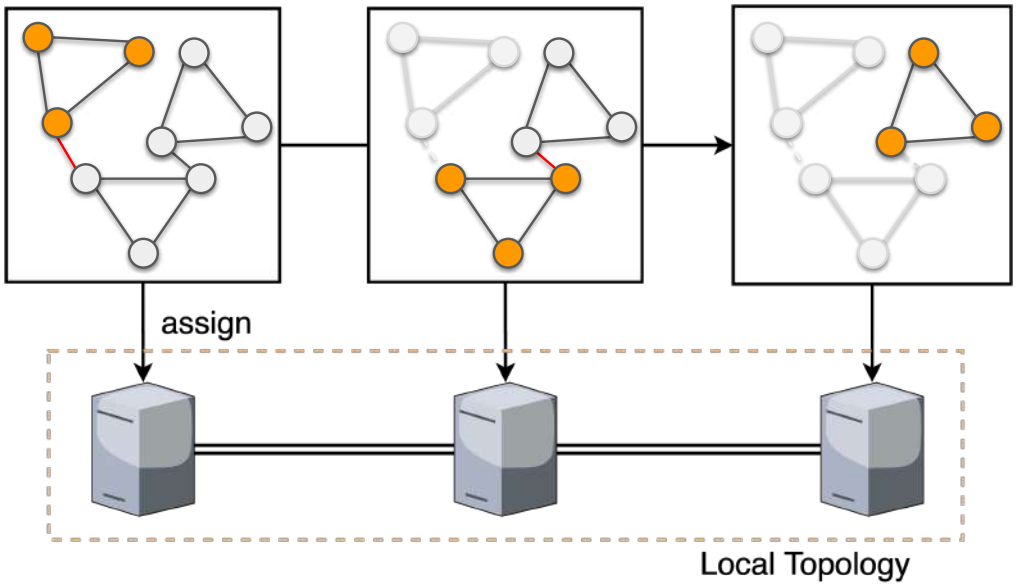


Figure 3.5: Resource allocation based on Stoer-Wagner algorithm. The allocator chooses one local topology and applies the Stoer-Wagner algorithm to it. The allocator assigns the first subgraph to the first node in the local topology and applies the Stoer-Wagner again. The allocator repeats this process until all nodes are assigned subgraphs.

a low-quality device connected by a high-quality link. Under this circumstance, if the low-quality device undertakes the dense subgraph, the final state would be noisy. From this point, the allocator partitions a target graph based on the density. Note that there are several algorithms to look up the densest subgraphs. However, no algorithm can approximate it with the multiplicative error so far.

The method to divide a graph based on subgraph's density, this allocator employed a method in [54].

Algorithm 5 Graph allocation based on Kernighan Lin algorithm

Input: G, $\mathbf{L}$ *Target graph* G *and local topology* $\mathbf{L}$

Output: $\mathbf{S}, C$ *List of subgraphs* $\mathbf{S}$ *and cost* C

Initialisation :

$G' = G$, $\mathbf{S} = \{\}$

Loop for devices in local topology

for $i = 1$ to n_l **do**

  $\mathbf{C}_T = \{inf\}$, $S_T = \{\}$ *Temporary cost list and temporary subgraphs*

  Loop for different partitions

  for $j = 1$ to N_i **do**

    $P_i = \{\{0, ..., j\}, \{j, ..., N_i\}\}$ *Two different paritions*

    $G'_1, G'_2 =$ Kernighan-Lin(G', P_i)

    $C_j =$ Cost($\{G'_1, G'_2\}$)

    if $C_j \leq \min(C_T)$ **then**

      $\mathbf{S}_T[j] = \{G'_1, G'_2\}$

    end if

    $\mathbf{C}_T[j] = C_j$

  end for

  Take the final (best) subgraph

  $G'_1, G'_2 = last(S_T)$ *Last element must be best in* S_T

  $\mathbf{S}[i] = G'_1$

  Update subgraph with left half of subgraph G'_2

  $G' = G'_2$

end for

C = Cost($\mathbf{S}, \mathbf{L}$)

return $\mathbf{S}$, C

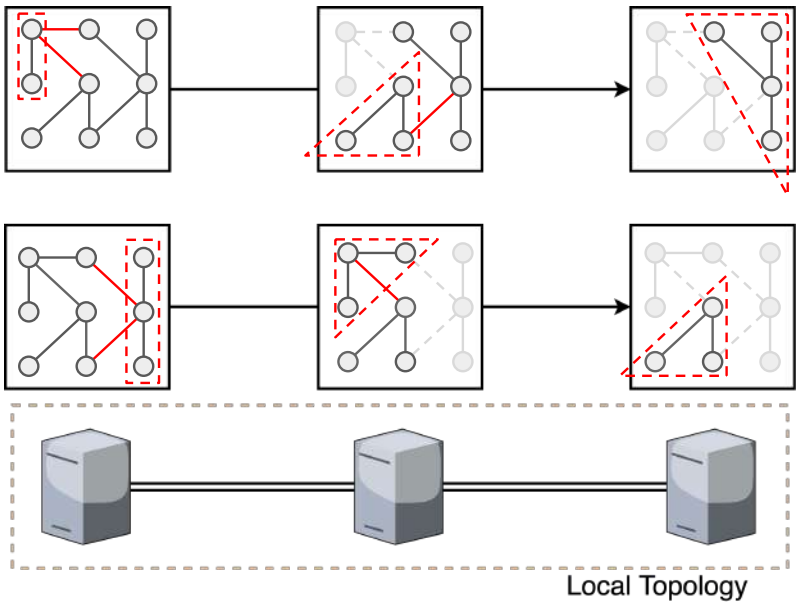


Figure 3.6: The procedure of Kernighan-Lin based allocation. Several partitions can be considered as many as the number of vertices. By changing the partitions, the allocator finds the best partition. The allocator repeats this process for all possible partitions.

Algorithm 6 Graph allocation based on densest-k subgraph search based algorithm

Input: G, $\mathbf{L}$ *Target graph* G *and local topology* $\mathbf{L}$

Output: $\mathbf{S}, C$ *List of subgraphs* $\mathbf{S}$ *and cost* C

Initialize a temporary subgraph G' *with target graph and* $\mathbf{S}$ *as empty list.*

$G' = G$, $\mathbf{S} = \{\}$

Loop for devices in the local topology $\mathbf{L}$

for $i = 0$ to n_l **do**

  $\mathbf{C}_T = \{inf\}$, $\mathbf{S}_T = \{\}$ *Temporary cost list and list for subgraph.*

  Loop for possible subgraphs with different number of vertices.

  for $j = 0$ to N_i **do**

    $G'_1, G'_2 = \mathrm{DkS}(G')$ *Densest-k subgraph search*

    $C_j = \mathrm{Cost}(\{G'_1, G'_2\})$

    if $C_j \leq \min(C_T)$ **then**

      $\mathbf{S}_T[j] = \{G'_1, G'_2\}$

    end if

    $\mathbf{C}_T[j] = C_j$

  end for

  Take the final (best) subgraph

  $G'_1, G'_2 = last(S_T)$

  $\mathbf{S}[i] = G'_1$

  Update subgraph with left half of subgraph G'_2

  $G' = G'_2$

end for

$\mathrm{C} = \mathrm{Cost}(\mathbf{S}, \mathbf{L})$

return $\mathbf{S}$, C

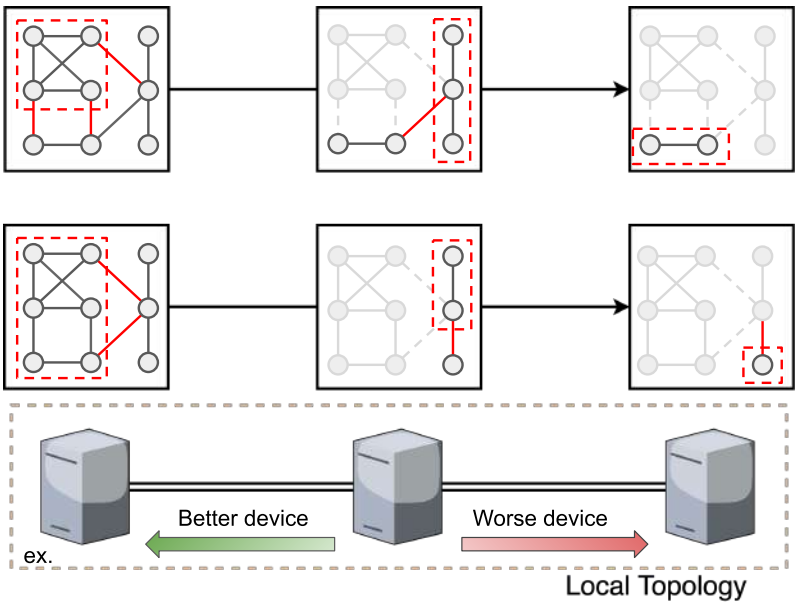


Figure 3.7: The procedure of the densest-k subgraph density based allocation. The allocator finds several candidates for the first subgraph. The allocator repeats this process for possible subgraphs.

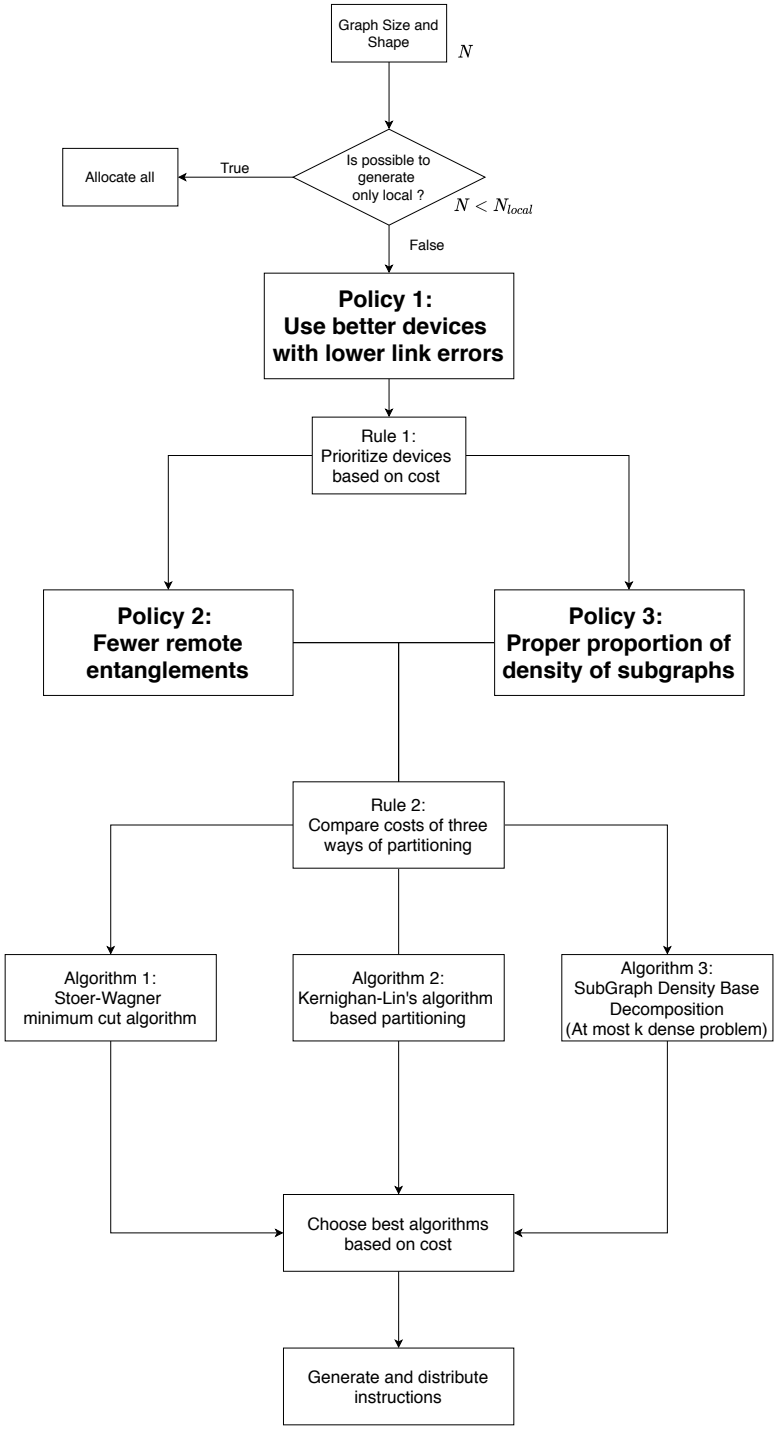


Figure 3.8: Policy stack to allocate resources

Chapter 4

Evaluation and Experiments

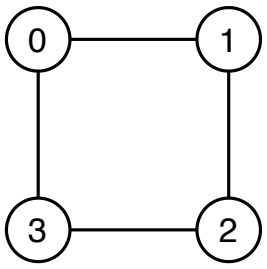
4.1 Evaluation

To evaluate if this allocation method is good, setting three criteria.

Quality: The quality of the final state based on the graph state verification method

To handle more than hundreds of qubits, a straightforward way to evaluate quantum state such as quantum tomography is not realistic in classical simulation. Instead of using such direct methods, using the graph state verification method as an evaluation method shown in 2.12. The procedure to evaluate a graph state is as follows.

First, prepare stabilizer generators according to the shape of the target state. As an example, suppose G is a graph shown in 4.1.



$$
\begin{aligned}
S &= \langle g_0, g_1, g_2, g_3 \rangle \\
g_0 &= X_0 \otimes Z_1 \otimes I_2 \otimes Z_3 \\
g_1 &= Z_0 \otimes X_1 \otimes Z_2 \otimes I_3 \\
g_2 &= I_0 \otimes Z_1 \otimes X_2 \otimes Z_3 \\
g_3 &= Z_0 \otimes I_1 \otimes Z_2 \otimes X_3
\end{aligned}
$$

Figure 4.1: Example of Stabilizer generators

Second, generate the graph state in local and global according to the subgraphs generated by the allocator. Note that, for simplicity, the simulator only cares about bit-flip and phase-flip errors.

Finally, measure the qubits based on the stabilizer generators prepared in the previous step. According to the measurement results, the allocator decides if the final

state is acceptable as the graph state. The simulator gets $m_i = 0$ or $m_i = 1$ as the measurement result when it measures i-th qubit. Let N be the number of qubits in the final state and $S_g = \langle g_0, g_1, ..., g_N \rangle$ be the stabilizer generators for that graph state. When the measurement results satisfy,

$$\prod_{s \in S_g} (-1)^{m_i} = 1 \tag{4.1.1}$$

the simulation accepts this as a graph state. Otherwise, the simulation rejects this graph state. The acceptance rate in one trial R_{acc} is,

$$R_{acc} = \frac{N_s}{N} \tag{4.1.2}$$

where N_s is the number of measurements that satisfies 4.1.1. The total acceptance rate R_t can be calculated by taking the average of hundreds of t trials.

$$R_t = \frac{\sum_i R^i_{acc}}{t} \tag{4.1.3}$$

When the final state is noisy, the value goes 0.5 (random output). Note that this is not a fidelity of the final state, so that the state can't be qualified with only that value. However, the final states can be compared in terms of that value.

Scalability: How fast the entire complexity grows. The scalability of the system is one dominant aspect of practical use. One way to evaluate the scalability of the system is the time complexity that tells us how hard the entire problem is.

Applicability: Constraints in practical use Finally, for practical use, some constraints can be considered.

4.2 Experiments

Several experiments can be set for investigating the performance of this protocol. First, by using different network topologies and shapes of graph state, investigate the quality of the final state.

Experiments	Topology	Target Graph
Experiment 1	Two node topology (Figure 4.3)	10×10 lattice (100 qubits)
Experiment 2	Two node topology (Figure 4.3)	30×30 lattice (900 qubits)
Experiment 3	Two node topology (Figure 4.3)	60 qubits random graph state [68]
Experiment 4	Four node topology (Figure 4.10)	10×10 lattice (100 qubits)
Experiment 5	Four node topology (Figure 4.10)	4×26 (104) brick
Experiment 6	Six node topology (Figure 4.15)	6×39 (234) brick

Table 4.1: Table of experiments with different network topology and different graph states. Figure 4.2 shows example of target state called brick state.

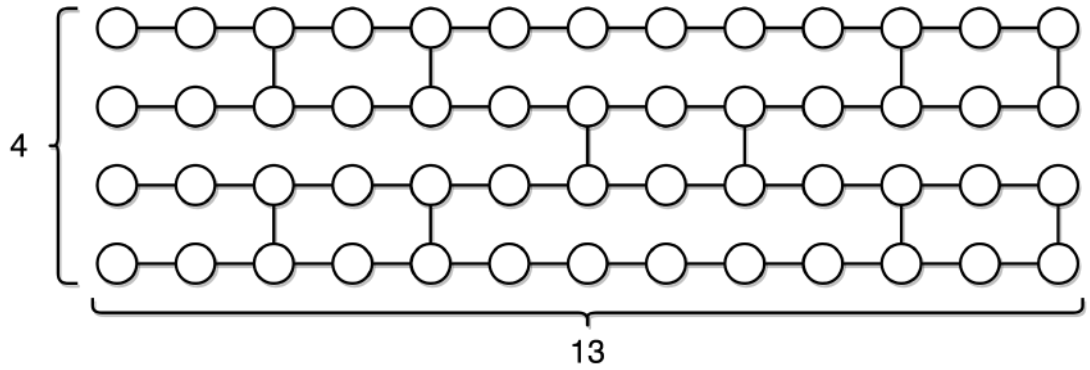


Figure 4.2: An example of 4×13 (52 qubit) brick state. This state can be used for blind quantum computation. The simplest graph state for universal measurement based quantum computing.

4.3 Results

4.3.1 Experiment 1: 100 qubits lattice state with two quantum processors

The first experiment is a toy experiment with two quantum devices that try to generate 100 qubits lattice state over the noisy quantum link and processors shown in Figure 4.3.

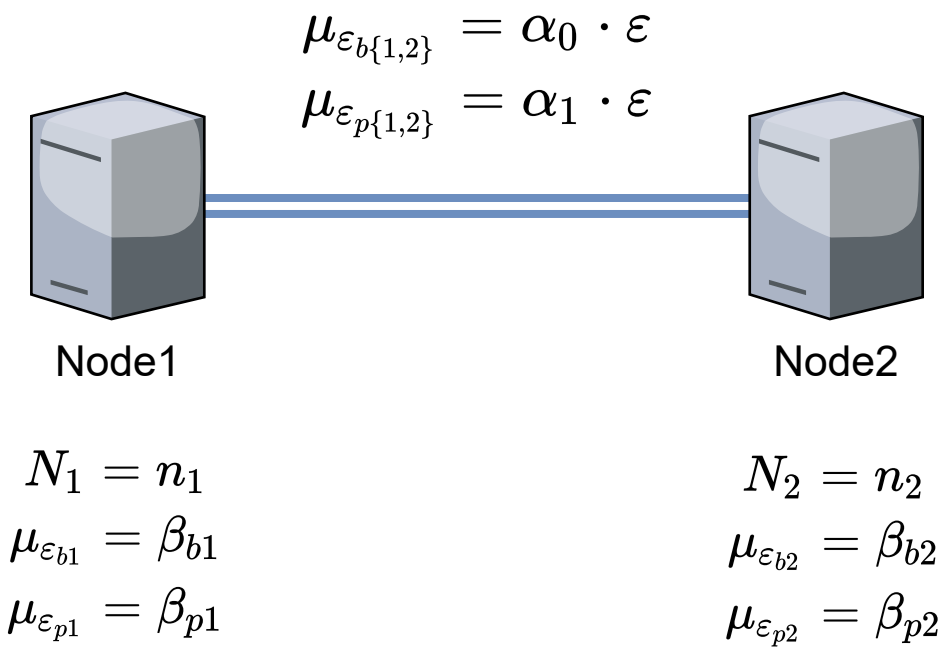


Figure 4.3: The simplest topology that has two quantum computers with one link. For the toy model of this simulation, both quantum computer have specific amount of qubits.

Figure 4.4 shows the quality transition over the different link errors (bit-flip and phase-flip) from 0 to 0.1 (X-axis). One data plot is the average of 100 trials, and error bars are generated by the standard deviations ($\pm 1\sigma$ range). As the error rate is getting large, the difference in qualities becomes large (The difference between the two values is almost 0.1 when the error rates are $\alpha = 0.1$).

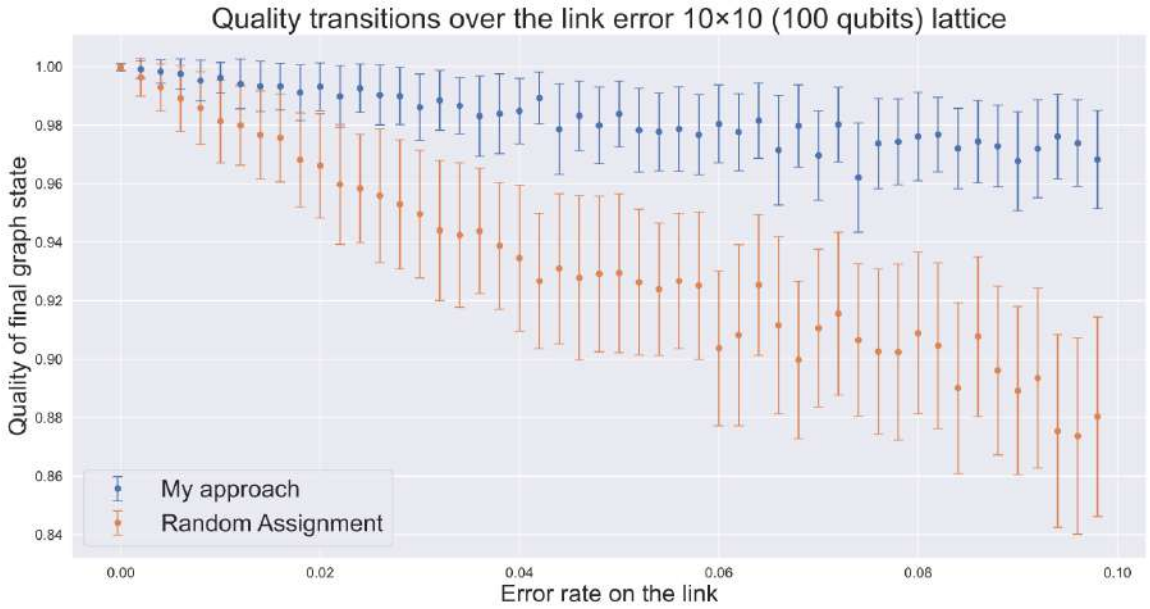


Figure 4.4: 10×10 (100 qubit) lattice graph state over the two quantum computers shown in 4.3 with 80 qubits each $n_1 = 80, n_2 = 80$. The blue dots represents the transition of the quality of graph state with this allocation method. The orange dots represents random allocation. By varying the error rate ε of 4.3, transition of quality can be seen in this plot.

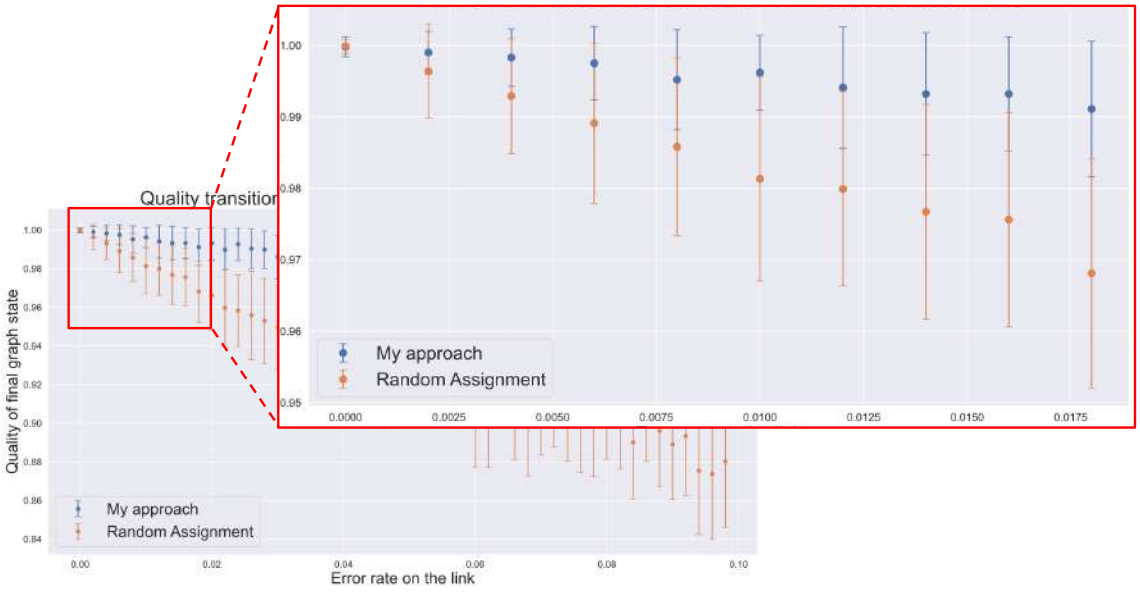


Figure 4.5: Quality transitions in small errors (0 to 0.02). The difference of quality between two allocation is remarkable even in the small error cases.

4.3.2 Experiment 2: 900 qubits lattice state with two quantum processors

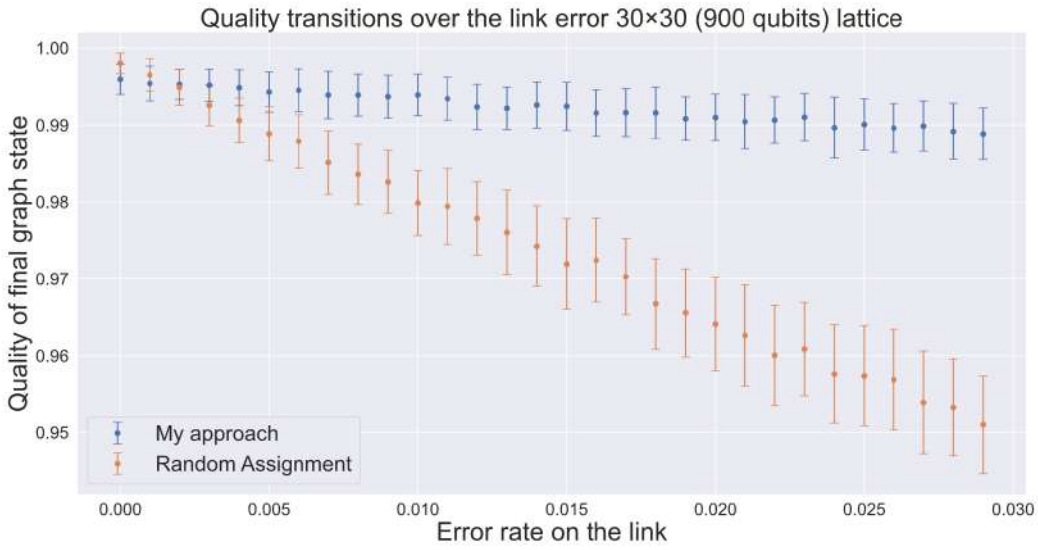


Figure 4.6: 30×30 (900 qubit) lattice graph state over the two quantum computers shown in 4.3.

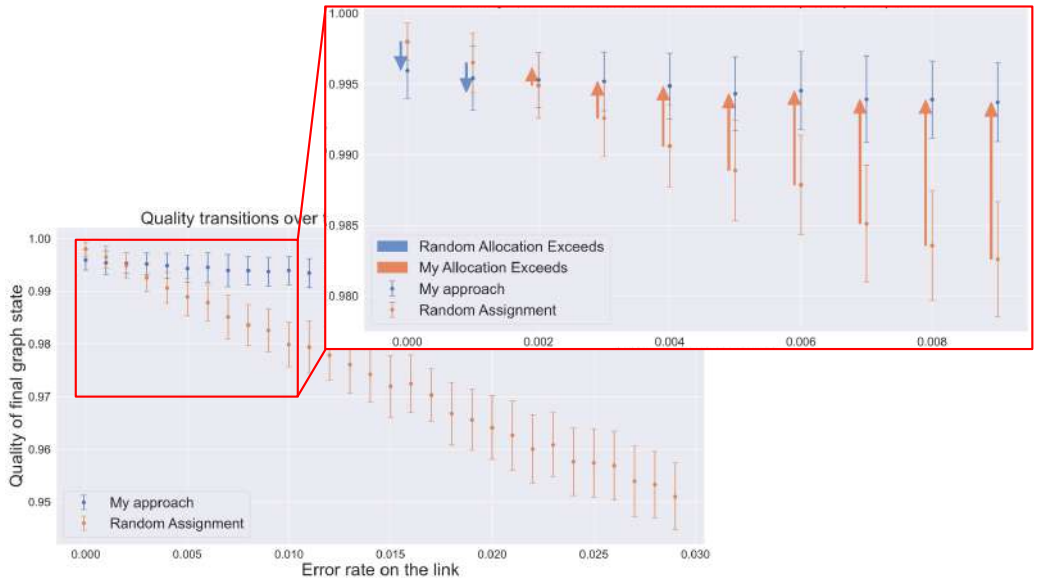


Figure 4.7: A magnified plot in small error rate cases. The orange arrows represents how much better the quality of this allocation is comparing to random allocation.

The second experiment is generating a 900 qubit lattice state with a two-node topology shown in Figure 4.3. The results are shown in Figure 4.6 and Figure 4.7. In

this experiment, the CZ gate's error rate varies from 0 to 0.03 (3%), and each node has small bit and phase errors on a single-qubit gate. This allocation method keeps quality between 0.99 to 1.0. However, in the small error case (0 to 0.002), random allocation is good enough to allocate qubits.

4.3.3 Experiment 3: 60 qubits random graph state with two quantum processors

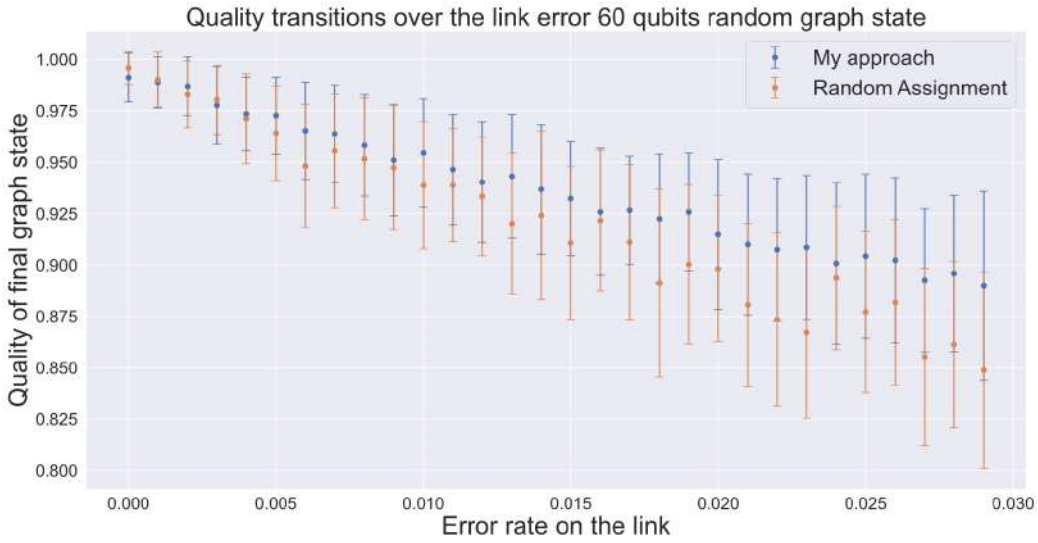


Figure 4.8: 60 qubit random sparse graph state over the two quantum computers shown in 4.3. As well as the previous result, the blue (red) dots represents this approach (random approach) respectively.

For the next experiment, the target state is a random graph state with 60 qubits. The type of random graph is called Erdős–Rényi model shown in [68] This result is derived from an experiment with a 60 vertices sparse random graph (the density of the target graph is 0.1). Figure 4.8 shows the difference of quality between this allocation method and the random allocation. Figure 4.9 highlights how much different they are. Comparing to the previous lattice experiments, the quality of this allocation gets lower rapidly. However, this allocation still outperforms the random allocation as the error rate on the link gets large.

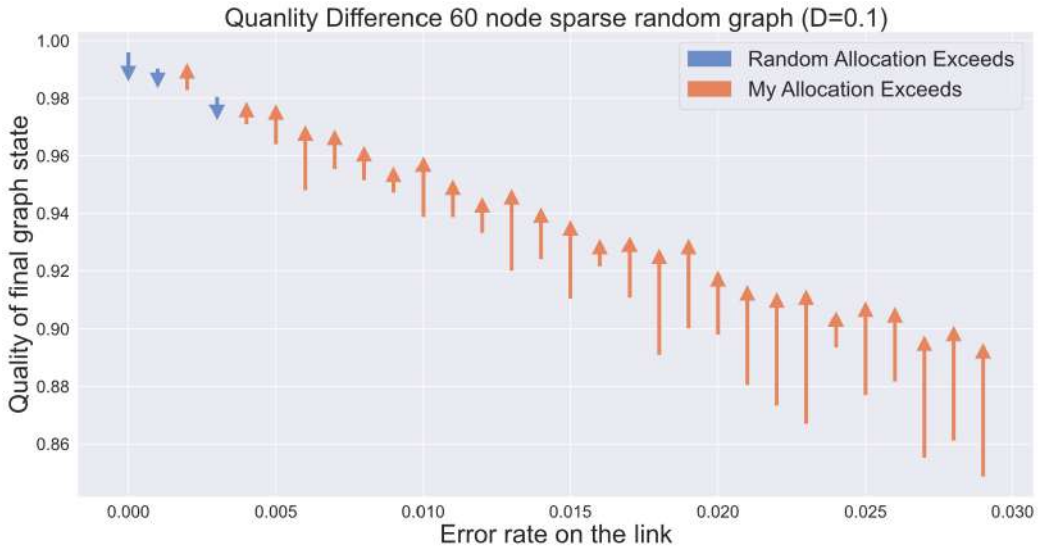


Figure 4.9: This plot shows how different the quality of the final state between this allocation and random allocation. The blue arrow shows this allocation method can not reach the quality of the random allocation's. On the other hand, the orange arrow means this allocation method exceed random allocation.

4.3.4 Experiment 4: 100 qubits lattice state with four quantum processors

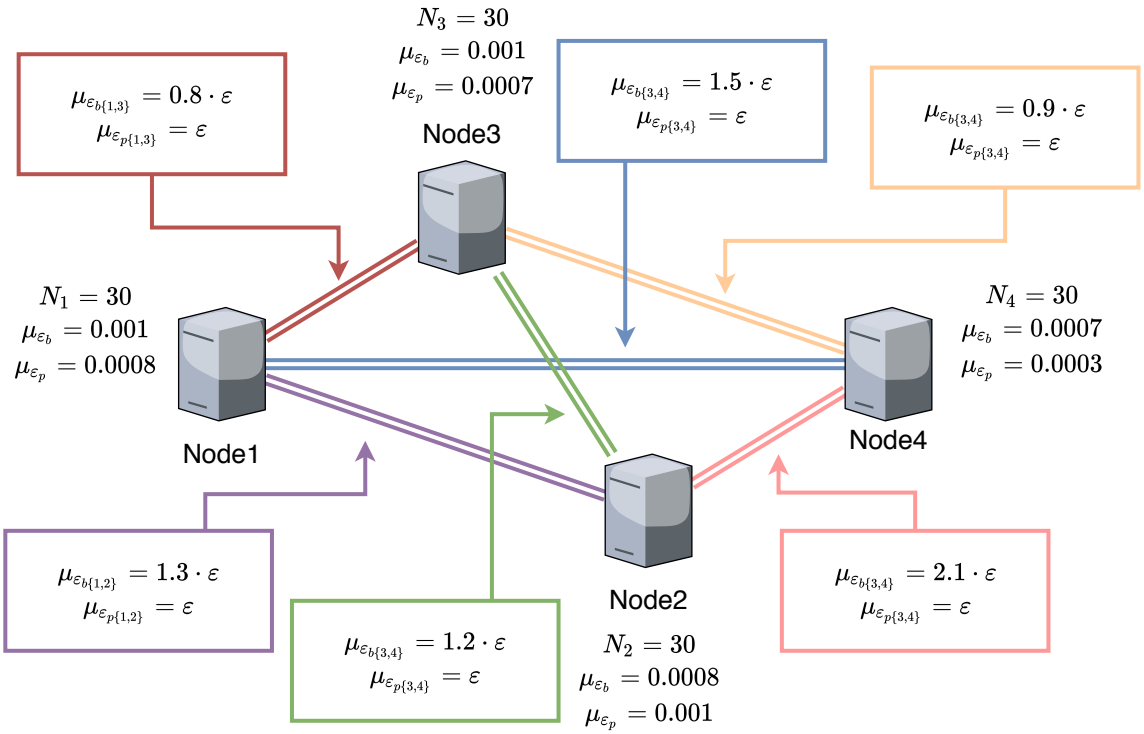


Figure 4.10: An example network topology with four small quantum computers with fixed error rate. μ_{ε_b} and μ_{ε_p} are the average bit-flip and phase-flip error rates respectively over the all qubits in a device and link.

The next experiment is on the four quantum processors shown in Figure 4.10 with 30 qubits each (120 qubits in total). Those four quantum processors try to generate a 100 qubits lattice state. The result is shown in Figure 4.11. In this case, the quality drops rapidly in both allocation method, but the random allocation outperforms this allocation. One possible reason that the random allocation exceeds this allocation method would be that this allocation method tries to allocate qubits greedily and make quantum computers generate much more remote entanglements.

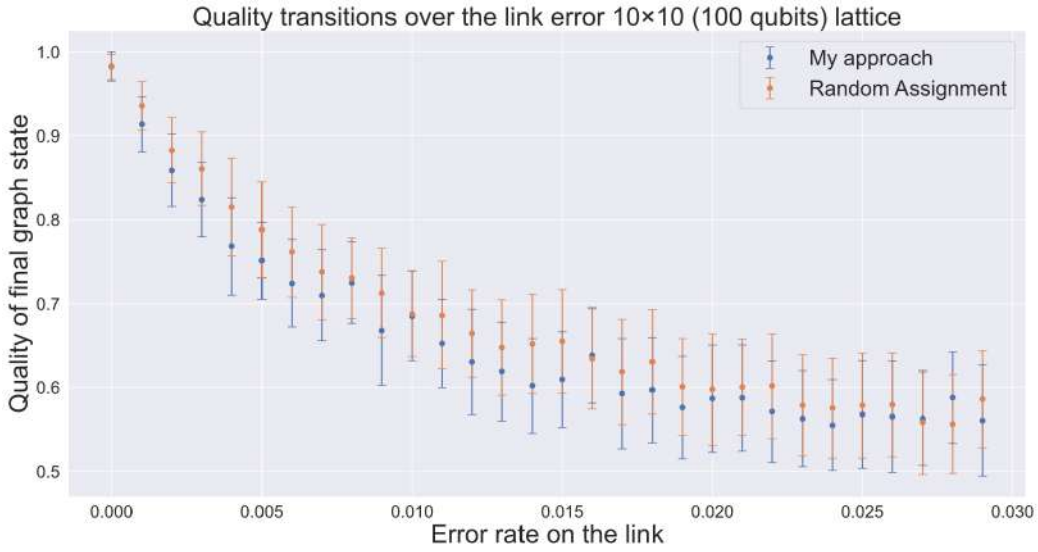


Figure 4.11: The result of experiment 4. The transitions of quality over the different bit and phase errors on the quantum link. Unlike the previous experiments, the quality of random allocation exceeds that of the random allocation in most cases.

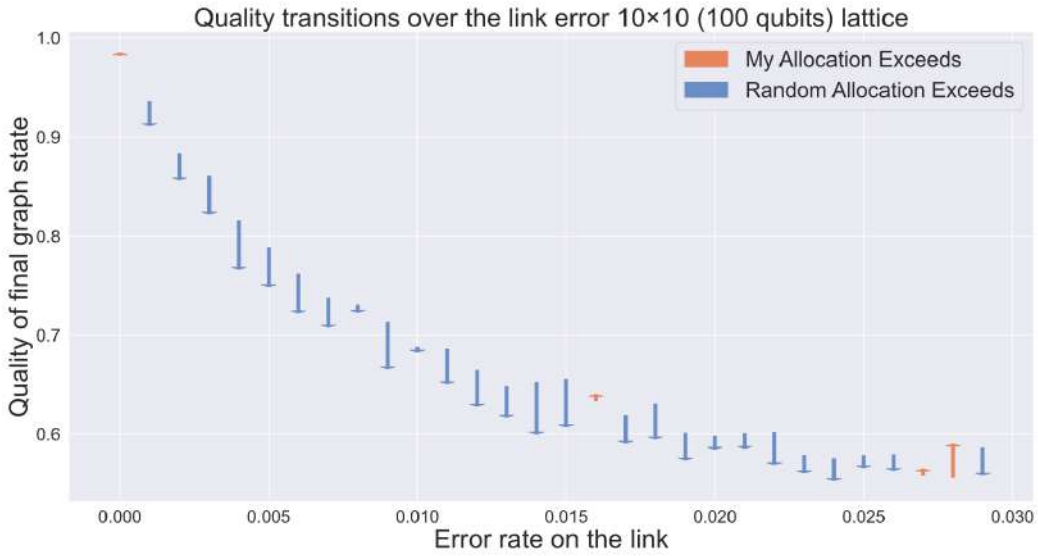


Figure 4.12: Highlighted visualization of the difference between two allocation methods. In this case, the random allocation exceeds in terms of quality.

4.3.5 Experiment 5: 104 qubits brick state with four quantum processors

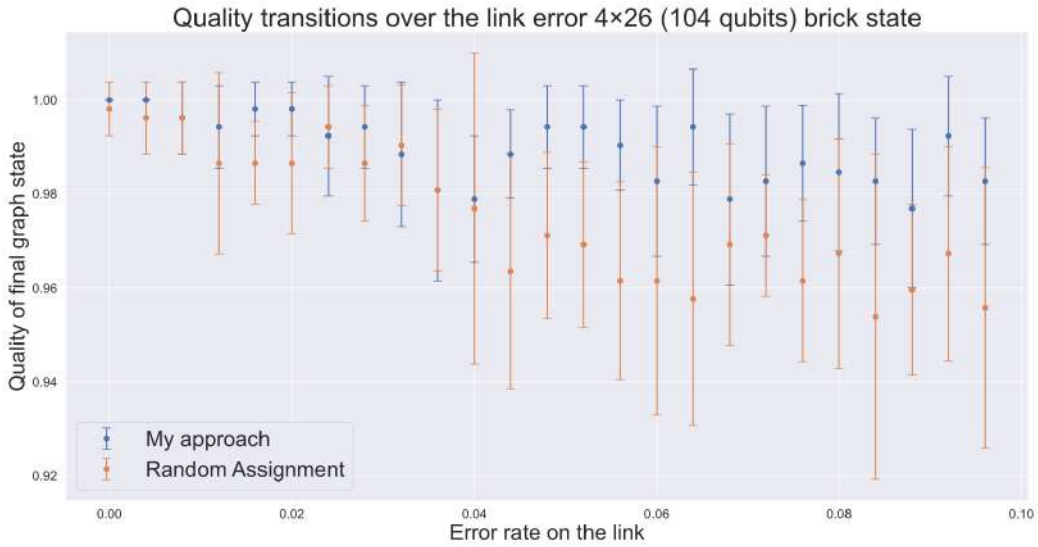


Figure 4.13: 4×26 (104 qubits) brick state on the network topology 4.10. The error bars are generated by standard deviations in 100 trials.

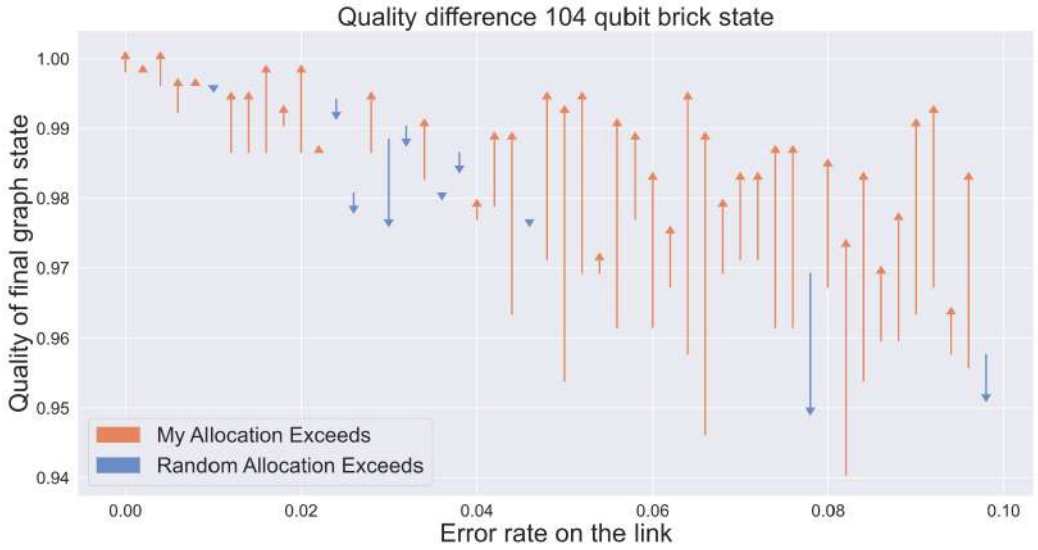


Figure 4.14: This plot shows quality differences of 4.13. When the error rate on the quantum link gets large, this allocation performs better than random allocation.

As a next experiment, the target state is different from previous experiments. The state is known as the "Brick state" shown in Figure 4.2. The corresponding

results are shown in Figure 4.13 and Figure 4.14. The size of the target brick state is $4 \times 26 = 104$ qubits. The dispersion of each data point is larger than that of previous results'.

4.3.6 Experiment 6: Six node topology and 234 qubits brick state.

For the final experiment, the number of quantum processors is larger than that of previous experiments shown in Figure 4.15. In this experiment, each quantum computer has small local gate errors (randomly generated from 0 to 0.001). The results are shown in Figure 4.16 and Figure 4.17

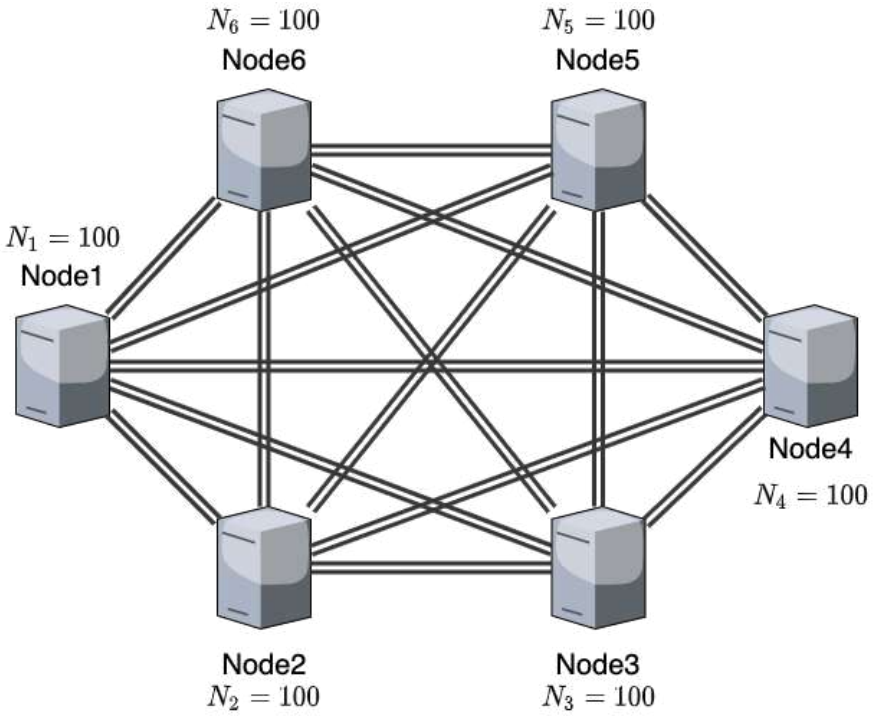


Figure 4.15: A network topology that contains six quantum computers. The same as the previous network topologies, all of them are connected by noisy links.

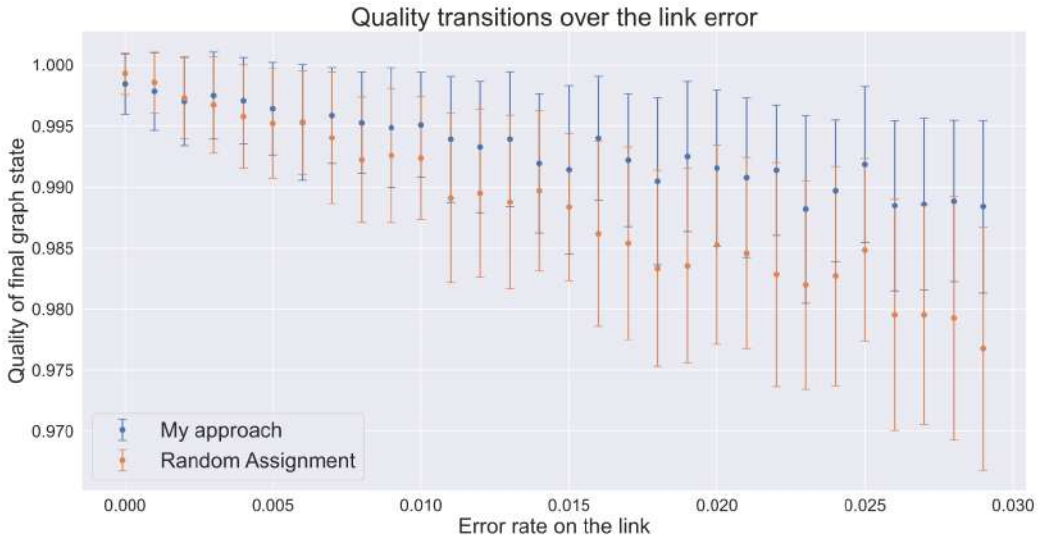


Figure 4.16: The result of 6 × 39 (234) brick state on Figure 4.15 with six quantum processors.

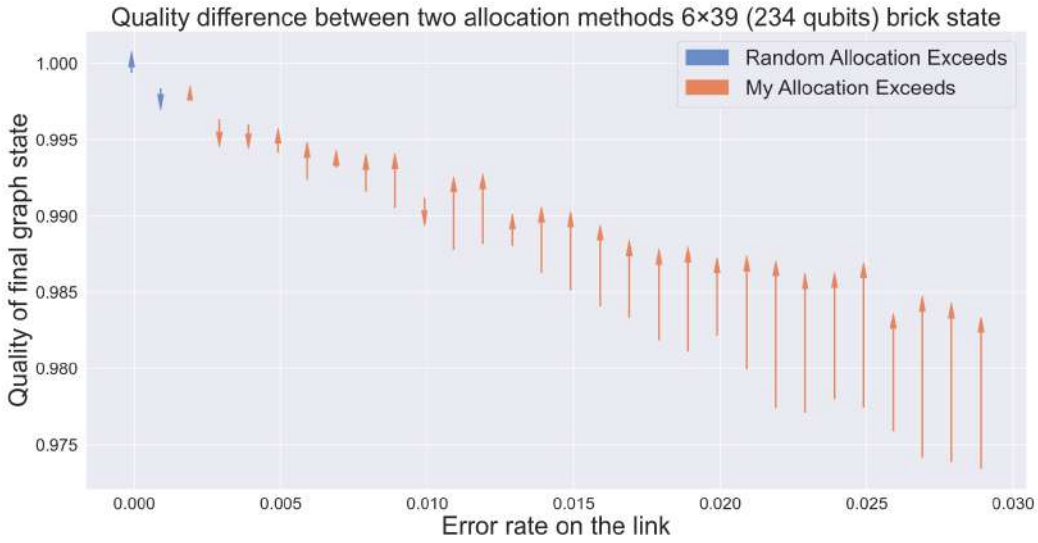


Figure 4.17: The highlighted plot of Figure 4.16. In this Figure, the difference between the two allocation methods is clearer than the previous results. When the error rate gets large, the difference gets large as well.

4.3.7 Scalability

The scalability can be measured by the time complexity of the entire process. Let $G = (V, E)$ be the target graph, and $T = (D, C)$ be the network topology. V and E represents the number of vertices and edges in the target graph, respectively. The same as the target graph, D represents the devices in the network topology, and C indicates the connections between devices. The total number of qubits N is $N = |V|$. The allocator can parallelize three allocations in the actual allocation phase. The bottleneck of the entire process can be the most complex allocation method, the densest-k subgraph search allocation. In this process, the densest-k subgraph search algorithm is the most complex allocation.

The time complexity can be decomposed into three parts. The first part is the number of local topologies candidates $\mathcal{L}$, and the second part is the subgraph generation $\mathcal{S}$. The complexity of the entire process is $O(\mathcal{LS})$. A set of subgraphs and cost can be considered for the number of possible local topologies.

There are two options for the local topology generation $\mathcal{L}$. The first option is searching for all possible candidates that satisfy the number of qubits. The other option is cutting off with pre-calculated cost and regulate the number of candidates. The worst-case can be $\mathcal{L} = O(n_l!)$ where n_l is the number of devices in a local topology ($n_l = |D|$). This is not practical in a large network topology. However, by setting a threshold for pre-calculated cost and the number of candidates, this could be $mathtcal L = O(1)$ where 1 a constant value. The strategy depends on how large the target graph and network topology are. As long as their size is reasonably small, all possible combinations can be calculated easily. The second part is the subgraph generation part. $\mathcal{S}$ represents the complexity of the subroutines of graph decomposition methods. If the allocator takes the Stoer-Wagner algorithm or Kernighan-Lin algorithm, S is $O(|V||E| + |V|^2 \log |V|)$ and $O(|V|^2 \log |V|)$. However, in the case that the allocator chooses the densest-k subgraph algorithm, this part is exponentially hard in multiplicative errors. Thus the allocator would have to avoid applying the densest-k subgraph search in terms of the overhead of the calculation in complex cases.

Chapter 5

Conclusion

This chapter concludes this paper and gives the future direction of this project. The aim of this paper was to establish an efficient resource allocation method for distributed quantum computing (DQC). This paper focused on generating graph state over distributed quantum computers for measurement-based distributed quantum computing. The key ideas for approaching that problem are three different graph algorithms, which allow us to make the allocation process flexible for any input graphs.

For the evaluation of this approach, several experiments were set, and qualities can be compared between the different allocation methods. In cases with small network topology and relatively large graph state, the allocation methods outperformed naive random resource allocation in terms of the quality of the final state. When the number of nodes in the network is large, the time complexity of the entire process would be large enough not to be able to handle. That because the allocator tries to investigate every possible local topology induced by the whole network topology. The solution to this problem is setting a threshold for the initial costs used to prioritize local topologies. This threshold can reduce the search space and prevents exploding time complexity.

As the future direction of this project, more detailed investigations can be considered. The simulator in this paper only applies bit-flip and phase-flip errors. More complex errors can be considered in a practical situation. As the next step of the protocol design is taking communication delay into account. Communication delay is also one dominant property that affects the quality of the quantum state.

This paper only showed the allocation of resources for graph state. However, in actual measurement-based quantum computing, the operations must be embedded in a graph state with measurements. This should be done by a quantum compiler in the distributed systems. This is still an open question of how to compile distributed quantum circuits efficiently in realistic situations.

Acknowledgement

I want to thank all of the people who involve in this thesis, especially Professor Rodney Van Meter, Project Assistant Professor Michal Hajdušek, Project Lecturer Takahiko Satoh, who guide me to a right direction with advice and discussions as leaders of AQUA. I also appreciate all my seniors and friends in AQUA, who encourage me to keep going. I finally would like to thank my family for their dedicated support.

Bibliography

[1] Richard P Feynman. Simulating physics with computers. *Int. J. Theor. Phys*, 21(6/7), 1982.

[2] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[3] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[4] Y. Nakamura, Yu. A. Pashkin, and J. S. Tsai. Coherent control of macroscopic quantum states in a single-Cooper-pair box. *Nature*, 398:786–788, April 1999.

[5] David P DiVincenzo. The physical implementation of quantum computation. *Fortschritte der Physik: Progress of Physics*, 48(9-11):771–783, 2000.

[6] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[7] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[8] William K Wootters and Wojciech H Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, 1982.

[9] Paul Adrien Maurice Dirac. A new notation for quantum mechanics. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 35, pages 416–418. Cambridge University Press, 1939.

[10] The Qiskit Team. Representing qubit states, Nov 2020.

[11] Akira Furusawa, Jens Lykke Sørensen, Samuel L Braunstein, Christopher A Fuchs, H Jeff Kimble, and Eugene S Polzik. Unconditional quantum teleportation. *science*, 282(5389):706–709, 1998.

[12] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004.

[13] Simon Anders and Hans J Briegel. Fast simulation of stabilizer circuits using a graph-state representation. *Physical Review A*, 73(2):022334, 2006.

[14] Sergey Bravyi, Dan Browne, Padraic Calpin, Earl Campbell, David Gosset, and Mark Howard. Simulation of quantum circuits by low-rank stabilizer decompositions. *Quantum*, 3:181, 2019.

[15] Chia-Hung Chien, Rodney Van Meter, and Sy-Yen Kuo. Fault-tolerant operations for universal blind quantum computation. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 12(1):1–26, 2015.

[16] Joseph F Fitzsimons. Private quantum computation: an introduction to blind quantum computing and related protocols. *npj Quantum Information*, 3(1):1–11, 2017.

[17] Xiaoqian Zhang, Weiqi Luo, Xiaoqing Tan, and Tingting Song. Measurement-based universal blind quantum computation with minor resources. *arXiv preprint arXiv:1801.03090*, 2018.

[18] Mark Hillery, Vladimír Bužek, and André Berthiaume. Quantum secret sharing. *Physical Review A*, 59(3):1829, 1999.

[19] Damian Markham and Barry C Sanders. Graph states for quantum secret sharing. *Physical Review A*, 78(4):042309, 2008.

[20] Robert Raussendorf and Hans J Briegel. A one-way quantum computer. *Physical Review Letters*, 86(22):5188, 2001.

[21] Hans J Briegel, David E Browne, Wolfgang Dür, Robert Raussendorf, and Maarten Van den Nest. Measurement-based quantum computation. *Nature Physics*, 5(1):19–26, 2009.

[22] Steven T Flammia and Yi-Kai Liu. Direct fidelity estimation from few pauli measurements. *Physical review letters*, 106(23):230501, 2011.

[23] Yuki Takeuchi and Tomoyuki Morimae. Verification of many-qubit states. *Physical Review X*, 8(2):021060, 2018.

[24] Alexandru Gheorghiu, Theodoros Kapourniotis, and Elham Kashefi. Verification of quantum computation: An overview of existing approaches. *Theory of computing systems*, 63(4):715–808, 2019.

[25] Yuki Takeuchi, Atul Mantri, Tomoyuki Morimae, Akihiro Mizutani, and Joseph F Fitzsimons. Resource-efficient verification of quantum computing using serfling's bound. *npj Quantum Information*, 5(1):1–8, 2019.

[26] Masahito Hayashi and Yuki Takeuchi. Verifying commuting quantum computations via fidelity estimation of weighted graph states. *New Journal of Physics*, 21(9):093060, 2019.

[27] Anupama Unnikrishnan and Damian Markham. Verification of graph states in an untrusted network. *arXiv preprint arXiv:2007.13126*, 2020.

[28] Simon J Devitt, William J Munro, and Kae Nemoto. Quantum error correction for beginners. *Reports on Progress in Physics*, 76(7):076001, 2013.

[29] H Jeff Kimble. The quantum Internet. *Nature*, 453(7198):1023–1030, 2008.

[30] Stephanie Wehner, David Elkouss, and Ronald Hanson. Quantum internet: A vision for the road ahead. *Science*, 362(6412), 2018.

[31] Jonathan P Dowling. *Schrödinger's killer app: race to build the world's first quantum computer*. CRC Press, 2013.

[32] Andy Chia, Michal Hajdušek, Rosario Fazio, Leong-Chuan Kwek, and Vlatko Vedral. Phase diffusion and the small-noise approximation in linear amplifiers: Limitations and beyond. *Quantum*, 3:200, 2019.

[33] W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299:802–803, October 1982.

[34] Juan Ortigoso. Twelve years before the quantum no-cloning theorem. *American Journal of Physics*, 86(3):201–205, 2018.

[35] James L. Park. The concept of transition in quantum mechanics. *Foundations of Physics*, 1(1):23–33, Mar 1970.

[36] D. Dieks. Communication by EPR devices. *Physics Letters A*, 92(6):271 – 272, 1982.

[37] Rodney Van Meter. *Quantum Networking*. Wiley-ISTE, April 2014.

[38] Sreraman Muralidharan, Linshu Li, Jungsang Kim, Norbert Lütkenhaus, Mikhail D Lukin, and Liang Jiang. Optimal architectures for long distance quantum communication. *Scientific Reports*, 6:20463, 2016.

[39] Filip Rozpędek, Raja Yehia, Kenneth Goodenough, Maximilian Ruf, Peter C Humphreys, Ronald Hanson, Stephanie Wehner, and David Elkouss. Near-term quantum-repeater experiments with nitrogen-vacancy centers: Overcoming the limitations of direct transmission. *Physical Review A*, 99(5):052330, 2019.

[40] M. Żukowski, A. Zeilinger, M. A. Horne, and A. K. Ekert. "event-ready-detectors" bell experiment via entanglement swapping. *Phys. Rev. Lett.*, 71:4287–4290, Dec 1993.

[41] Jian-Wei Pan, Dirk Bouwmeester, Harald Weinfurter, and Anton Zeilinger. Experimental entanglement swapping: Entangling photons that never interacted. *Phys. Rev. Lett.*, 80:3891–3894, May 1998.

[42] H.-J. Briegel, W. Dür, J. I. Cirac, and P. Zoller. Quantum repeaters: The role of imperfect local operations in quantum communication. *Phys. Rev. Lett.*, 81:5932–5935, Dec 1998.

[43] W. Dür, H.-J. Briegel, J. I. Cirac, and P. Zoller. Quantum repeaters based on entanglement purification. *Phys. Rev. A*, 59:169–181, Jan 1999.

[44] Koji Azuma, Kiyoshi Tamaki, and Hoi-Kwong Lo. All-photonic quantum repeaters. *Nature Communications*, 6:6787, April 2015.

[45] Donovan Buterakos, Edwin Barnes, and Sophia E. Economou. Deterministic generation of all-photonic quantum repeaters from solid-state emitters. *Phys. Rev. X*, 7:041023, Oct 2017.

[46] Yasushi Hasegawa, Rikizo Ikuta, Nobuyuki Matsuda, Kiyoshi Tamaki, Hoi-Kwong Lo, Takashi Yamamoto, Koji Azuma, and Nobuyuki Imoto. Experimental time-reversed adaptive Bell measurement towards all-photonic quantum repeaters. *Nature communications*, 10(1):1–9, 2019.

[47] Zheng-Da Li, Rui Zhang, Xu-Fei Yin, Li-Zheng Liu, Yi Hu, Yu-Qiang Fang, Yue-Yang Fei, Xiao Jiang, Jun Zhang, Li Li, et al. Experimental quantum repeater without quantum memory. *Nature Photonics*, 13(9):644–648, 2019.

[48] Paul Hilaire, Edwin Barnes, and Sophia E Economou. Resource requirements for efficient quantum communication using all-photonic graph states generated from a few matter qubits. *arXiv preprint arXiv:2005.07198*, 2020.

[49] Mechthild Stoer and Frank Wagner. A simple min-cut algorithm. *Journal of the ACM (JACM)*, 44(4):585–591, 1997.

[50] Dec 2016.

[51] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970.

[52] Si Pi Ravikumār and CP Ravikumar. *Parallel methods for VLSI layout design*. Greenwood Publishing Group, 1996.

[53] Carl Kingsford. Kernighan-lin, graph distance metrics.

[54] Juan Miguel Arrazola and Thomas R Bromley. Using gaussian boson sampling to find dense subgraphs. *Physical review letters*, 121(3):030503, 2018.

[55] Vasil S Denchev and Gopal Pandurangan. Distributed quantum computing: A new frontier in distributed systems or science fiction? *ACM SIGACT News*, 39(3):77–95, 2008.

[56] Daniele Cuomo, Marcello Caleffi, and Angela Sara Cacciapuoti. Towards a distributed quantum computing ecosystem. *arXiv preprint arXiv:2002.11808*, 2020.

[57] Stephen DiAdamo, Marco Ghibaudi, and James Cruise. Distributed quantum computing and network control for accelerated vqe. *arXiv preprint arXiv:2101.02504*, 2021.

[58] Yuan Liang Lim, Almut Beige, and Leong Chuan Kwek. Repeat-until-success linear optics distributed quantum computing. *Physical review letters*, 95(3):030505, 2005.

[59] Rodney Van Meter, Thaddeus D Ladd, Austin G Fowler, and Yoshihisa Yamamoto. Distributed quantum computation architecture using semiconductor nanophotonics. *International Journal of Quantum Information*, 8(01n02):295–323, 2010.

[60] Robert Beals, Stephen Brierley, Oliver Gray, Aram W Harrow, Samuel Kutin, Noah Linden, Dan Shepherd, and Mark Stather. Efficient distributed quantum computing. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 469(2153):20120686, 2013.

[61] Rodney Van Meter and Simon J Devitt. The path to scalable distributed quantum computing. *Computer*, 49(9):31–42, 2016.

[62] Santiago Rodrigo, Sergi Abadal, Eduard Alarcón, and Carmen G Almudever. Exploring a double full-stack communications-enabled architecture for multi-core quantum computers. *arXiv preprint arXiv:2009.08186*, 2020.

[63] Vincent Danos, Ellie d'Hondt, Elham Kashefi, and Prakash Panangaden. Distributed measurement-based quantum computation. *Electronic Notes in Theoretical Computer Science*, 170:73–94, 2007.

[64] SR Clark, C Moura Alves, and D Jaksch. Efficient generation of graph states for quantum computation. *New Journal of Physics*, 7(1):124, 2005.

[65] Adán Cabello, Lars Eirik Danielsen, Antonio J López-Tarrida, and José R Portillo. Optimal preparation of graph states. *Physical Review A*, 83(4):042314, 2011.

[66] Clément Meignant, Damian Markham, and Frédéric Grosshans. Distributing graph states over arbitrary quantum networks. *Physical Review A*, 100(5):052333, 2019.

[67] Alex Fischer and Don Towsley. Distributing graph states across quantum networks. *arXiv preprint arXiv:2009.10888*, 2020.

[68] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.