New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG IOP Institute of Physics

Multi-agent blind quantum computation without universal cluster states

New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics
Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

OPEN ACCESS

RECEIVED
2 May 2023

REVISED
1 September 2023

ACCEPTED FOR PUBLICATION
18 September 2023

PUBLISHED
16 October 2023

PAPER

Multi-agent blind quantum computation without universal cluster states

Shuxiang Cao

Department of Physics, Clarendon Laboratory, University of Oxford, Oxford, OX1 3PU, United Kingdom

E-mail: shuxiang.cao@physics.ox.ac.uk

Keywords: blind quantum computing, ZX-calculus, measurement-based quantum computing

Abstract

Blind quantum computation (BQC) protocols enable quantum algorithms to be executed on third-party quantum agents while keeping the data and algorithm confidential. The previous proposals for measurement-based BQC require preparing a highly entangled cluster state. In this paper, we show that such a requirement is not necessary. Our protocol only requires pre-shared Bell pairs between delegated quantum agents, and there is no requirement for any classical or quantum information exchange between agents during the execution. Our proposal requires fewer quantum resources than previous proposals by eliminating the need for a universal cluster state.

1. Introduction

Quantum computers built from current technology are difficult to be miniaturized, and unlikely to become personal electronics such as a laptop or a cellphone [1–4]. Therefore, cloud-based services are considered the most applicable approach to offer the general public access to quantum computers. It is natural to ask whether the privacy of the quantum algorithm can be kept when one does not have complete control of the quantum hardware. Blind quantum computing (BQC) aims to solve this problem. Quantum algorithms can be executed with BQC protocols on third-party quantum agents while keeping the algorithm, data, and results confidential [5, 6].

Here we discuss two ways to implement universal quantum computation. One is gate-based quantum computing (GBQC) [7]. This method starts with a pure quantum state, usually by resetting all qubits to zero. Then it transforms the quantum state using a sequence of quantum gates. The final output state carries the processed information. The other method is called measurement-based quantum computing (MBQC) or one-way quantum computation [8–11]. This method prepares a highly entangled state of multiple qubits, often referred to as a *cluster state* [12], then performs a sequence of measurements and corrections to implement computation. Eventually it can give the same result as the GBQC.

The universal blind quantum computing (UBQC) protocol was proposed in [6] based on the MBQC framework. UBQC protocol utilizes a universal cluster state and can be implemented by a semi-classical client with a single agent or an entirely classical client with multiple agents. There are other proposals implementing BQC with a single agent and an entirely classical client are possible, however, these proposals require some computational assumptions [13–15].

In this paper, we make use of a quantum graphical reasoning method, ZX-Calculus, to derive a BQC protocol that can be implemented with multiple agents and an entirely classical client. The UBQC protocol utilizes a universal cluster state, forcing all the information describing the algorithm to be encoded in the measurement axis. It sacrifices the ability to encode information into the entanglement structure between qubits. Contrarily, our method does have information encoded in the entanglement structure, and does not require a universal cluster state. This makes our protocol more resource-efficient.

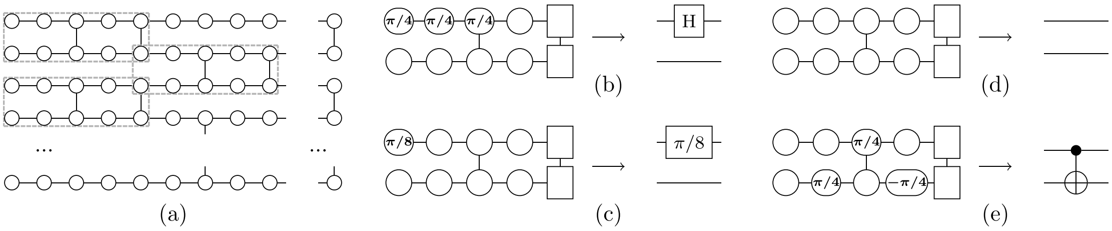
This paper is arranged as follows: section 2.2 describes ZX-calculus, a graphical quantum reasoning technique that we use to derive our result. Section 3 explains our BQC protocol. Section 4 gives proof of the correctness and secureness of our protocol. Section 6 discusses the compatibility with existing verification protocols, and quantifies the resource cost of our protocol and the UBQC protocol. Section 7 summarizes the paper.

2. Background

2.1. UBQC

The UBQC protocol employs the MBQC method to implement BQC [6]. Under the MBQC framework, the algorithm can be described with only the entanglement structure between qubits and each qubit's measurement axis. To make the algorithm blind to the agents, the information each agent possesses, the entanglement structure, the measurement axis, and the measurement output of the agents, must not reveal any information about the algorithm. A valid BQC protocol must make this information independent from the delegated task.

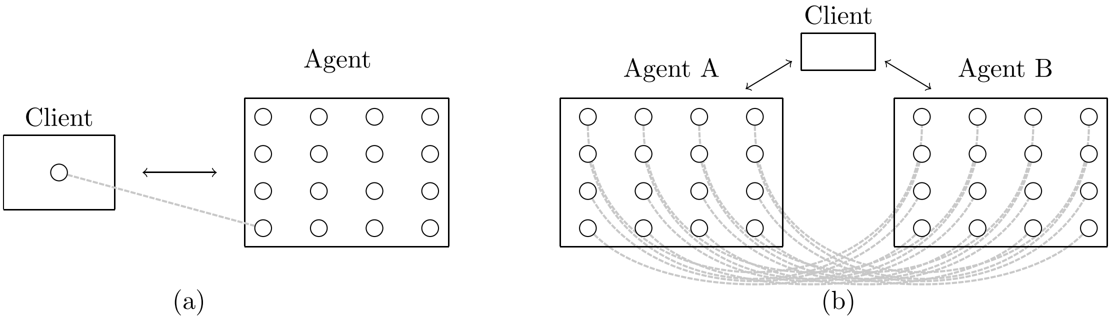
To make the entanglement structure of the delegated task independent from the quantum algorithm, UBQC utilizes a universal cluster state, which can implement arbitrary quantum algorithms with the same entanglement structure but a different measurement axis. Such a method concentrates the information describing the quantum algorithm on the measurement axis. UBQC protocol uses the brickwork cluster state to implement MBQC. Different quantum gates can be implemented by measuring the cluster state with different angles in sequence. For the brickwork state, the qubits are measured from left to right. Based on the measurement result, *corrections* is applied to the following qubits on each step. The calculated result is then stored in the qubit on the right end of the brickwork cluster state and can be further processed by piling up more elementary components, or more 'bricks'.



Example 2.1. Brickwork cluster state and MBQC with brickwork resource state. Each node denotes a qubit prepared in $|+\rangle$ state. Wires connecting two qubits denote an entanglement that is generated by applying a CZ gate between two qubits. The result is stored inside the very right qubits after measuring each qubit from left to right. The angle inside each node represents the angle of measurement that would be applied to the corresponding qubit. (a) The layout of a typical brickwork cluster state. The gray square shows a fundamental element of the brickwork cluster state. (b) Implement a Hadamard gate. The square on the left side denotes the qubits that hold the computation output. (c) Implement T ($\pi/8$) gate. (d) Implement identity gate. (e) Implement a CNOT gate.

Usually the measurement axis is defined by doing a single qubit rotation before the physically implementable measurement, which is usually the Pauli Z axis. To make the measurement axis independent to the quantum algorithm, a second agent is introduced to implement all or part of the single qubit rotation. When only one remote agent is available, UBQC protocol requires the client to be semi-classic; that is, the client can manipulate a minimum of a single qubit. It also requires the remote server to exchange quantum information by physically swapping qubits or establishing new entanglement. See example 2.2(a). In the original proposal, known as the 'prepare-and-measure' method, the semi-classical agent effectively prepares the measurement angle. The agent and the client share the entanglement of each qubit, and the client measures its qubit at a random angle. This random angle would be 'teleported' to the agent and affect the cluster state. The agent only needs to initialize the qubit into a superposition state and directly measure the qubit without rotating the qubit [6]. Alternatively, the agent can provide the cluster state and send the state back to the agent, known as the 'measurement-only' method. Only the agent has access to the measurement angle [16]. The UBQC protocols can also be implemented with multiple remote quantum agents and a purely classical client when the two agents' communication is restricted. See example 2.2(b). A uniformly distributed measurement axis for the delegated agent can be implemented on the first agent by simply requesting the second agent to measure their entangled qubits from a random axis. Then the computation can continue with the same method for a single agent UBQC.

The measurement outcome for the 'prepare-and-measure' approach is obfuscated by randomly flipping the outcome distribution during the measurement. Such obfuscation can be done by randomly choosing to measure at its original or with a π difference. The measurement outcome would flip when the measurement angle is chosen with π difference. Then the client classically restores the distribution after the measurement. Since the agent does not know if the distribution has been flipped or not, it can only observe a uniform distribution of 0 and 1 outcomes.



Example 2.2. Two protocols of UBQC. Both methods execute quantum algorithms with MBQC on the brickwork cluster state prepared on the remote agents. (a) Protocol with a semi-classical client and a single remote agent. The client can manipulate only one qubit and exchange qubits with the agent—the client prepares phase or measures the qubit at a random angle. The agent would know the actual rotation angle obfuscated by this random angle. (b) Protocol with a full classical client and multiple remote agents with shared entanglement. A second agent is introduced to replace the semi-classical client.

It is worth mentioning that a circuit-based BQC method proposed in [17] utilizes a similar philosophy as the UBQC protocol. A 'universal circuit' that can implement arbitrary operation by modifying the single-qubit gate rotation angle has been introduced in the proposal. The entanglement structure is then irrelevant to the circuit on the agent, and the rotation angles are obfuscated with quantum computing on encrypted data [18–20], which requires the agent to exchange quantum information with the client. The circuit-based protocol computes the cluster state in a circuit-based manner. However, it still requires exchanging the same amount of quantum information between the client and agent as the UBQC protocol to implement 'correction'.

2.2. ZX-calculus

In this section, we provide a brief review of the ZX-calculus [21]. The ZX-calculus is a diagrammatic method for reasoning the linear maps of quantum operations. With the gate representation of quantum computation, we decompose a unitary operation into a sequence of predefined gates; with ZX-calculus, we decompose the unitary into a network, the so-called *ZX-diagram*, consists of red and green spiders. In the following discussion, we ignore the scalars of the ZX-diagrams.

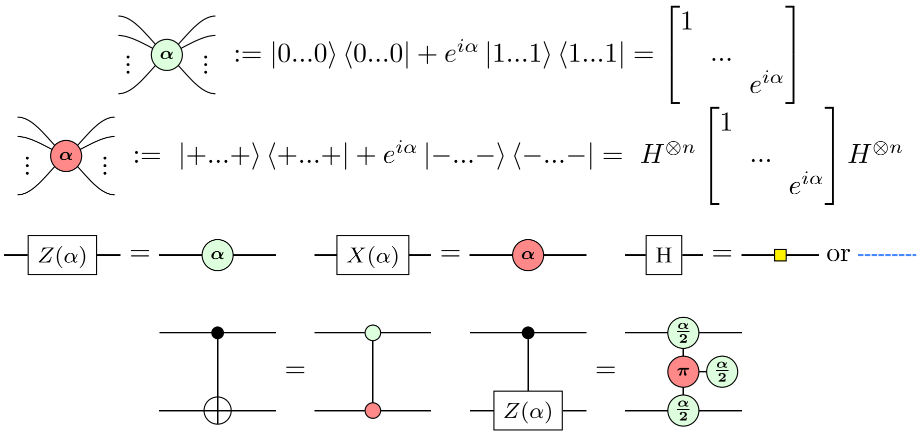
2.2.1. ZX-diagram

A ZX-diagram consists of *wires* and *spiders*, corresponding to legs and tensors in the tensor network language [22, 23]. There are two types of spiders: Z spiders and X spiders noted as green and red dots. The spiders are defined as a tensor parameterized by a single phase variable. The opened wire can be considered an input or output of the ZX-diagram. The summary of the definition of the basic building blocks of ZX-diagram is shown in example 2.3. A quantum circuit can be easily rewritten to a ZX-diagram with rules provided in example 2.3. The X gate can be replaced with a red spider, and the Z gate can be replaced with a green spider. The rotation angle is represented as the phase of each spider.

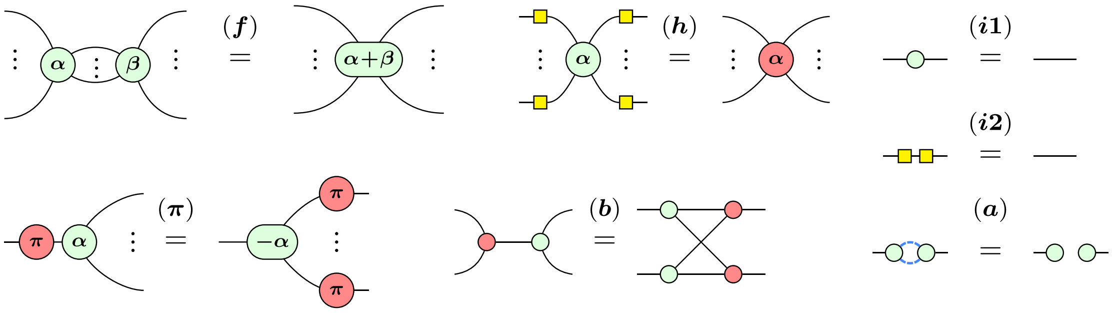
Instead of directly contracting the tensor network, ZX-calculus provided a set of rules to manipulate ZX-diagrams while keeping them equivalent. ZX-calculus is complete on Clifford+T language with the set of rules are specified [24, 26]. Some of these rules are shown in example 2.4.

Instead of representing the gates with both input wires and output wires, spiders can have just a single wire. With a single output wire, the spider represents a *bra* notation. Such *bra* notation denotes to a post-selection operation when extracting the diagram into a quantum circuit. While with a single input wire, the spider represents a *ket* notation. Such *ket* notation denotes preparation of the initial state of the quantum circuit.

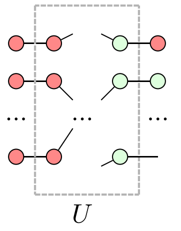
A ZX-diagram can also represent a density matrix. For a pure state $|\psi\rangle$, the density matrix is $\rho = |\psi\rangle\langle\psi|$, which is the tensor product of $|\psi\rangle$ and $\langle\psi|$. In a ZX-diagram, a tensor product can be represented by putting two disconnected diagrams together. By writing $|\psi\rangle$ and $\langle\psi|$ into the same diagram, we have the ZX-diagram of the density matrix ρ shown in example 2.6(b).


$:= |0...0\rangle\langle 0...0| + e^{i\alpha}|1...1\rangle\langle 1...1| = \begin{bmatrix}1 & & \\ & \cdots & \\ & & e^{i\alpha}\end{bmatrix}$
$:= |+...+\rangle\langle +...+| + e^{i\alpha}|-...-\rangle\langle -...-| = H^{\otimes n}\begin{bmatrix}1 & & \\ & \cdots & \\ & & e^{i\alpha}\end{bmatrix}H^{\otimes n}$
$Z(\alpha)$
$X(\alpha)$
H
or
α
π
$\frac{\alpha}{2}$

Example 2.3. Basic building blocks of ZX-diagram. ZX-diagram is a notation representing tensor networks. Any quantum circuit can be converted into a quantum tensor network and further represented by ZX-diagram [24, 25]. ZX-diagram consists of *spiders*, which is a tensor with constraints above. Standard quantum circuits can be converted into ZX-diagram with the rules listed above.



Example 2.4. The power of ZX-calculus is it derives a set of rules to transfer a ZX-diagram to another one while keeping them equivalent. Here we show several rules that we will use later. The (f) rule indicates that any two spiders with the same color can be merged. The (h) indicates that spider color can be changed by adding a Hadamard spider on every wire of the spider. (π) indicates that a Pi operation with a different color can be copied and moved to other wires while changing the sign of the spider's phase. Also, $(i1)$ and $(i2)$ can help generate or remove redundant spiders and Hadamard nodes. (b) the bialgebra rule. The Hopf law or antipode law (a) shows two parallel Hadamard wire results canceling each other.

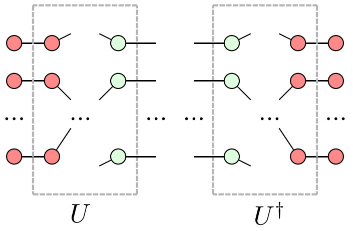


State $U|0_n\rangle$ with post-selection

Example 2.5. Post-selection in ZX-diagram. The post-selection is represented by attaching a spider with no output to the output wires of the ZX-diagram. Apply a red dot denoting post select the $|0\rangle$ state, and a green dot means $|+\rangle$ state.

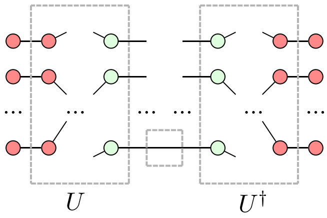
A mixed state can be generated by partially tracing away part of a pure system. The reduced density matrix of which some qubits are traced away can be represented by directly connecting the wire of the traced-away qubits between the $|\psi\rangle$ diagram and the $\langle\psi|$ diagram. This is shown in example 2.7.

Although converting an arbitrary quantum circuit into a ZX-diagram is easy, it is not always trivial to convert a ZX-diagram back into a quantum circuit. A ZX-diagram can always represent an arbitrary gate in quantum circuits; however, a ZX-diagram can also represent a non-unitary tensor. For example, the number of input and output wires can be different. The process of converting a ZX-diagram into a quantum circuit is often referred to as *circuit extraction* [27].



Density matrix of state $U\left|0_n\right\rangle$

Example 2.6. The representation of a pure state density matrix. Putting two isolated ZX-diagram together gives the tensor product between to ZX-diagram. Suppose density matrix is $\rho = |\psi\rangle \otimes \langle\psi|$, it can be represented by placing two ZX-diagram of $|\psi\rangle$ and $\langle\psi|$ together.



Reduced density matrix by tracing away a qubit

Example 2.7. The ZX-diagram representation of a reduced density matrix by tracing away one of the qubits. The reduced density matrix can be represented by directly connecting the open wires of the qubit. In this example, the two open wires of the last qubit are connected, marked with a rectangle.

2.2.2. Graph-like ZX-diagram

There is a special form of ZX-diagram that is particularly useful, called *graph-like ZX-diagram* [28].

Definition 2.8. A diagram is called graph-like if

1. All spiders are Z-spiders.
2. Spiders are only connected via Hadamard wires.
3. There are no parallel Hadamard wires or self-loops.
4. Every input or output is connected to a Z-spider.
5. Every Z-spider is connected to at most one input or output.

An example is shown in example 2.9.

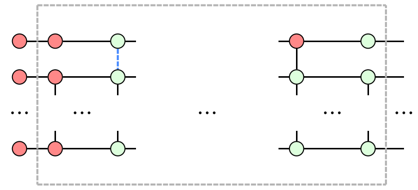
From the Gottesman–Knill theorem, quantum circuits containing gates only from the Clifford group can be simulated efficiently on a classical computer [29]. After a quantum circuit is written into a ZX-diagram, it is possible to simplify the diagram and remove Clifford operations before further modifications. This technique has been developed for circuit simplification [28].

2.3. Flow and determinism in MBQC

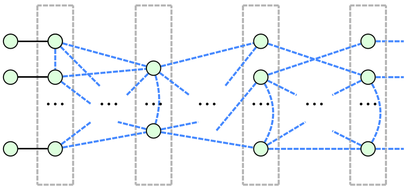
Suppose a circuit is converted into a ZX-diagram and then transformed into a graph-like ZX-diagram. Then we modify the ZX-diagram and give every spider an extra regular wire. See example 2.10(a). The new diagram equals the original diagram when all these wires terminate with a zero-phase Z spider.

Now let us consider each spider corresponds to a qubit, and measure the qubit closes the added open wire with either a zero-phase spider or a π-phase spider. When we get an unwanted π-phase spider, see example 2.10(b), we restore the state by applying extra single-qubit operations on the related qubit which we have not been measured. This is equivalent to the *correction* operation for MBQC. See example 2.10(c). In this way, we can always obtain the same distribution as the original quantum circuit.

Pushing the π phase into unmeasured spiders is a simple correction strategy; however, it does not work for arbitrary graphs. For example, if a non-output qubit is being measured with an unexpected result, and all its neighboring qubit has already been measured, then there is no qubit the π phase can be pushed to. For a diagram that can utilize this single qubit correction strategy, the diagram must admit a *causal flow* [30].

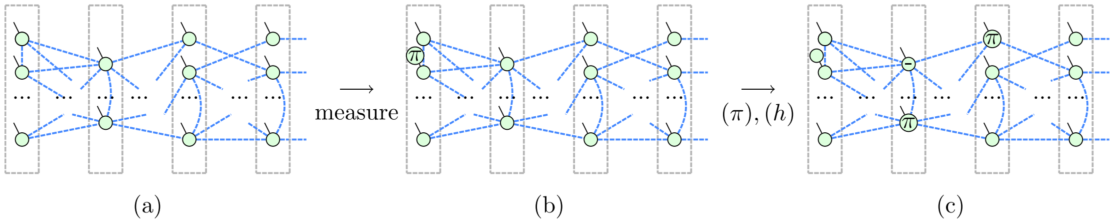


(a)Original circuit in ZX-diagram



(b)Circuit represented in graph-like ZX-diagram

Example 2.9. A graph-like ZX-diagram. A graph-like ZX-diagram must have only Z-spiders (green), and the internal connections are only Hadamard wires (dashed-blue line). There are no self-loops or parallel wires, and each spider is connected to at most one input or output. (a) demonstrates a schematic of an original circuit written into ZX-diagram. It can be created by substituting each quantum gate in the circuit with its corresponding ZX-diagram component. This diagram contains both X and Z spiders, with each row representing a qubit and each column denoting a layer of the circuit. (b) Demonstrate schematics of a graph-like ZX-diagram, comprised solely of Z spiders and Hadamard edges. A graph-like ZX-diagram that is equivalent to a diagram like (a) can always be found [28]. In a graph-like ZX-diagram, the columns and rows no longer correspond directly to a gate layer or a qubit. The open Hadamard edges with the ellipsis denote some arbitrary configuration that is abbreviated.



Example 2.10. Measurement sequence of implementing correction. (a) The spiders have been grouped based on their distance from the output spiders. Shown in gray squares. (b) The measurement has been performed on the group with the largest distance; some unexpected outcome has been measured. (c) Apply the $(\boldsymbol{\pi})$ and $(\boldsymbol{h})$ to recover the state. The phase can always get propagated into groups with a lower distance to the output spiders.

Definition 2.11 (causal flow [30, 31]). A *causal flow* is a pair $(f,\prec)$ with $\prec$ a partial order and f a function $f: O_c \to I_c$ on open graph state (G,I,O) which associates with every non-output vertices a set of non-input vertices such that

- $u \in N(f(u))$.
- $u \prec f(u)$
- $u \prec v$ for all $v \neq u$, $v \in N(f(u))$.

where $N(K)$ denote the neighbor vertices of K.

Consider we are looking for a strategy to execute the graph with the MBQC method, which consists of measuring the qubits and modifying one qubit $f(u)$ after each measurement. The partial order of the causal flow describes a possible order to execute the measurement. $f(u)$ qubits must be measured after u, which gives the second rule. Also, applying the correction would affect not only u, but all the neighbors of $f(u)$. Therefore these neighbors must be measured after u.

However, admitting a causal flow is unnecessary for a graph to be executed with MBQC [30–33]. Recall that modifying one qubit can correct the unexpected measurement outcome from a graph with the causal flow. There are at least two improvements to the flow mechanism that can be applied.

First, instead of correcting the state by modifying one qubit, the idea of graph stabilizers can be used to obtain a set of qubits and corresponding operations to correct the state. A stabilizer of a graph is a set of operations that can be applied to the state while keeping it identical. To correct the state, we could consider the unexpected gate as part of a stabilizer, which would keep the state unchanged if we complete it. Such stabilizers can be found intuitively with ZX-calculus by pushing the unexpected π phase around the phase-free graph [24, 34]. We then could relax $f(u)$ to be multiple qubits called the *correction set*.

Second, for the qubits measured on the Pauli basis, some correction does not need to be applied to the qubit physically [31, 35]. For example, pushing a π spider through another π phase spider with a different color does not need to apply physical corrections because $-\pi$ and π phase are equivalent.

The above two modifications lead to the definition of *Pauli flow*.

Definition 2.12 (Pauli flow [35]). An open graph state (G, I, O) has Pauli flow if there exists a map $f\colon O^c \to F(I^c)$ and a partial order $\prec$ over V such that for all $u \in O^c$

1. if $v \in f(u)$, and $\lambda(v) \notin X, Y$ then $u \prec v$,
2. if $v \neq u$, and $\lambda(v) \notin Y, Z$ then $v \notin \mathrm{Odd}(f(u))$,
3. if $v \preceq u, v \in f(u)$ and $\lambda(v) = Y$ then $v \in \mathrm{Odd}(f(u))$,
4. if $\lambda(u) = XY$ then $u \notin f(u)$ and $u \in \mathrm{Odd}(f(u))$,
5. if $\lambda(u) = XZ$ then $u \in f(u)$ and $u \in \mathrm{Odd}(f(u))$,
6. if $\lambda(u) = YZ$ then $u \in f(u)$ and $u \notin \mathrm{Odd}(f(u))$,
7. if $\lambda(u) = X$ then $u \in \mathrm{Odd}(f(u))$,
8. if $\lambda(u) = Z$ then $u \in f(u)$,
9. if $\lambda(u) = Y$ then either: $u \notin f(u)$ and $u \in \mathrm{Odd}(f(u))$ or $u \in f(u)$ and $u \notin \mathrm{Odd}(f(u))$.

Where $\mathrm{Odd}(K) = \{u, |N(u) \cap K| = 1 \bmod 2\}$ is the odd neighbor of K, i.e. the set of vertices which have an odd number of neighbors in K. $N(K)$ denote the neighbor vertices of K. $|K|$ denote the number of vertices in K. $\lambda(u)$ denote the measurement plane of u, for green spiders with 0 or π phase, the measurement plane is X. The measurement plane is XY for other arbitrary phases.

In the following sections, we show that all the rewrite rules used to implement the protocol would at least preserve the Pauli flow of the graph. This guarantees the transformed ZX-diagrams can be executed on MBQC hardware.

3. BQC from ZX-calculus

The previous section shows that a graph-like ZX-diagram can fully describe the information needed to execute a quantum algorithm. This information includes each spider's phase, the connectivity of the ZX-Diagram which represent the entanglement structure required by the the algorithm, and the number of spiders used in the graph. The outcome of each measurement may also contain information about the result of the algorithm.

The UBQC protocol splits the initial phase into two parts to obfuscate this information. Each is independent of the initial phase; however, the execution would yield the same result when combined. The measurement results are obfuscated by randomly flipping the qubits before measurement and classically restoring them by the client after measurement. For obfuscation of the connectivity, the UBQC protocol utilizes a universal cluster state; therefore, any algorithm would have an identical entanglement structure. Using the universal cluster state forces all the information of the algorithm to be stored in the phases. Limiting the ability to represent information with the layout of the cluster state requires extra resources. An overview of our protocol is presented in Figure 1. Our proposal obfuscates the connectivity by making those connectivities carry information about the algorithm entanglement structure and become connectivities between two different agents. Because each agent only possesses the fragment of the diagram that executes on itself, it loses track of the entanglement structure of the algorithm. Because our protocol does not require the universal cluster state, and encoding a considerable portion of the algorithm into the connectivity between the spiders, it requires fewer resources than the UBQC protocol.

In this section, we show how to implement our protocol with ZX-calculus. The phases and measurement results are obfuscated with similar approaches to the UBQC protocol. With proper manipulation of the ZX-diagram, the connectivity information can all be hidden by ensuring that each agent only possesses one end of the entanglement that holds the information about the entanglement structure.

3.1. Defining blocks

Here we introduce the concept of *blocks*. Blocks B_i are a set of spiders that are hosted by the same agent. $B(V) = B_k$ denote the block spider V belongs to block B_k. For simplicity, we define $B_i - 1 = B_{i-1}$.

Definition 3.1 (spider depth). For a given quantum algorithm represented in ZX-diagram $G(E_{\mathrm{N}}, E_{\mathrm{H}}, V)$, let $d_{\mathrm{G}}(V_1, V_2)$ denote the distance of V_1 and V_2 in graph G, V_{o} denotes all output spiders. Define depth of the spider V_i in the graph G as

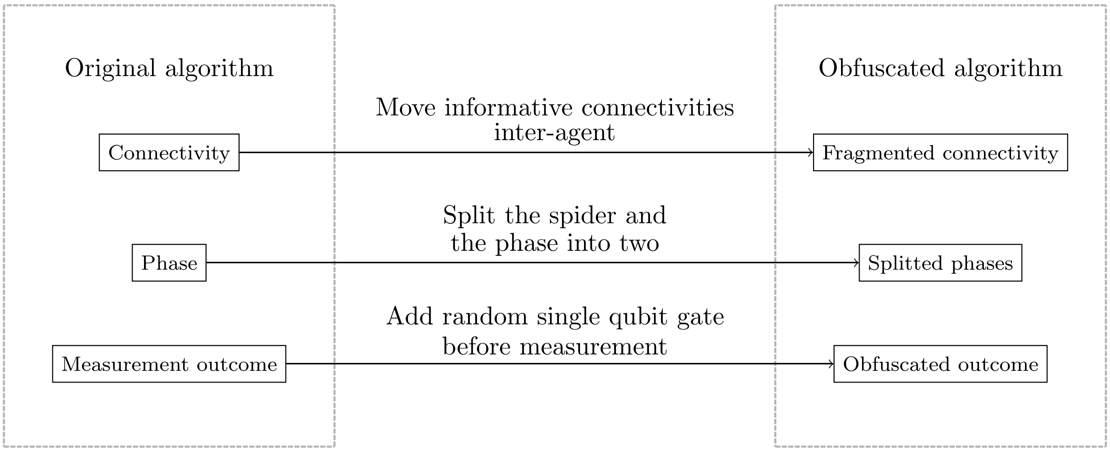


Figure 1. An overview of the obfuscation strategy of our proposed protocol.

$$D\left(V_i\right) = \min\left(d\left(V_i, V_j\right)\right), \forall V_j \in V_{\mathrm{O}}. \tag{1}$$

Definition 3.2 (blocks initialization). Define the block as a set of spiders, and spider V_i belongs to block $B(V_i)$, given by

$$B\left(V_i\right) = B_k \tag{2}$$

where

$$k = D\left(V_i\right). \tag{3}$$

To illustrate this partition, consider a quantum circuit directly transformed into a ZX-diagram. This partition simply categorizes each layer of the quantum circuit into an individual block.

3.2. Phase obfuscation

Before we move into the method, let us start with a few definitions.

Definition 3.3 (semi-graph-like diagram). A diagram is called semi-graph-like if

1. All spiders are Z-spiders.
2. There are no parallel Hadamard wires or self-loops.
3. Every input or output is connected to a Z-spider.
4. Every Z-spider is connected to at most one input or output.

The difference between a semi-graph-like diagram and a graph-like diagram is that it allows regular edges to be present in the graph.

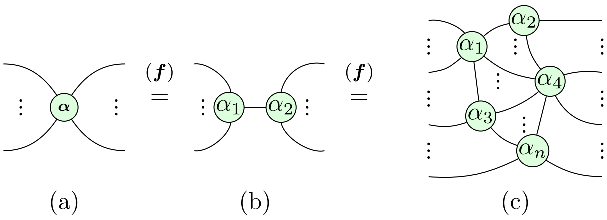
Definition 3.4 (reduced graph-like diagram). A graph-like diagram G_{g} is the *reduced graph-like diagram* of a semi-graph-like diagram G_{sg} if G_{sg} can be transformed into G_{g} with only rule (f).

Now we consider the obfuscation process of the phase of a spider. The goal of this obfuscation step is to rewrite the graph so that the individual phase value in the new graph is independent of the phase values in the original graph. Such rewrite can be implemented by applying the (f) to make a single spider become multiple spiders connected with regular edges, see example 3.5. Suppose the original spider has phase α. The new phases for new spiders are α_i. Rule (f) shows that the rewrite graph is equivalent to the original graph if phase α_i is chosen to satisfy

$$\alpha = \sum_i \alpha_i. \tag{4}$$

When the operation to implement phase α is split into multiple operations across different agents, each single agent would not be able to find the original phase α.

We have shown the obfuscated diagram is equivalent to the original diagram, and next, we show the obfuscated diagram can also be executed on the physical hardware in an MBQC manner.



Example 3.5. Phase obfuscation with rule $(\boldsymbol{f})$. (a) is the original spider with multiple inputs and outputs. (b) is equivalent to (a), while the phase has been split into two spiders connected with a regular wire. α_1 and α_2 can be chosen randomly with the restriction $\alpha = \alpha_1 + \alpha_2$. Adding one extra spider gives minimum protection to hide the rotation phase from the agent. (c) depict a more general form where the spider can be split into n spiders.

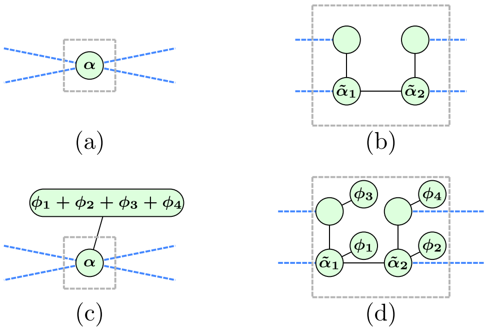
Theorem 3.6 (spider split flow preservation). *Given a graph state (G, I, O), where G is a graph-like diagram $G(E_\mathrm{N}, E_\mathrm{H}, V)$. Split spider $V_i \in V$ into N spiders, $\tilde{\mathbf{V}}_i = \{\tilde{V}_i^{(0)} \dots \tilde{V}_i^{(N)}\}$, that is construct a new graph $\tilde{G}(\tilde{E}_\mathrm{N}, \tilde{E}_\mathrm{H}), \tilde{V}_i$, where $\tilde{V} = V$ except V_i is replaced with a set of spiders V_i^k. V_i^k are connected through regular edges. If (G, I, O) admit a Pauli flow $(f, \prec)$, the new graph $\tilde{G}$ also admit a Pauli flow.*

Proof. By measuring all the splitted spider $\tilde{V}_i^{(n)}$, we could obtain all $\phi_{\tilde{\phi}^i}$. Using $(\boldsymbol{f})$ to merge the split spider back to one spider, we can obtain the effective measurement outcome for V_i as $\phi_{V_i} = \sum_N \phi_{\tilde{V}^i}$. Therefore the split spiders $\boldsymbol{\forall}_\mathbf{i}$ has the same predecessors and successors as V_i in partial order $\prec$. The partial order $\tilde{\precsim}$ for $\tilde{G}$ can also be constructed as follows.

$$\begin{aligned}
&\tilde{\precsim} = \bigcup\{(V_m, V_n)\} \cup \bigcup\{(V_i', V_k)\} \cup \bigcup\left\{\left(V_j, V_i'\right)\right\}, \\
&\quad \forall\,(V_m, V_n) \in \prec, V_m \neq V_i, V_n \neq V_i, \\
&\quad \forall\left(V_j, V_i\right) \in \prec \text{ and } \forall\,(V_i, V_k) \in \prec .
\end{aligned} \tag{5}$$

□

Since V_i satisfies all the requirements from definition of Pauli flow 2.12, each node $\{\tilde{V}_i^{(0)} \dots \tilde{V}_i^{(N)}\}$ also satisfies all the requirements. Therefore new graph $\tilde{G}$ also admits a Pauli flow.



Example 3.7. Execution strategy on a ZX-diagram with regular edges. For a single spider in the original ZX-diagram (a), it is split into multiple spiders (b) by rule $(\boldsymbol{f})$. All split spiders were measured when executing a measurement step in the original diagram (d). Inversely apply rule $(\boldsymbol{f})$ gives an equivalent effect of measuring the single spider in the original diagram (c).

From the theorem 3.6, we define the Pauli flow for a semi-graph-like diagram.

Definition 3.8 (Pauli flow on semi-graph-like diagram). A semi-graph-like diagram admits a Pauli flow if its reduced graph-like diagram admits a Pauli flow.

Lemma 3.9 (spider rule flow preservation). *Rewrite rule $(\boldsymbol{f})$ preserves Pauli flow on semi-graph-like diagrams.*

Proof. Because applying $(\boldsymbol{f})$ to a semi-graph-like diagram will not change its reduced graph-like diagram. From definition 3.8, $(\boldsymbol{f})$ will not affect the flow property of the diagram. □

Theorem 3.10 (phase obfuscation). *Given a quantum algorithm represented in graph-like ZX diagram $G = (\emptyset, E_{\mathrm{H}}, V)$ with n spiders, where $\emptyset$ denote the null set and E_{H} denote the Hadamard edges. Each spider V_i has phase α_i. A graph $\tilde{G} = (\tilde{E}_{\mathrm{N}}, \tilde{E}_{\mathrm{H}}, \tilde{V})$ equivalent to G can always be found, preserves the Pauli flow of G, and each individual phase $\tilde{\alpha}_i$ is independent to G.*

Proof. Construct $\tilde{G} = (\tilde{E}_{\mathrm{N}}, \tilde{E}_{\mathrm{H}}, \tilde{V})$, with $2n$ spiders. The Hadamard edge $\tilde{E}_{\mathrm{H}}$ and regular edge $\tilde{E}_{\mathrm{N}}$ is given by

$$\tilde{E}_{\mathrm{H}} = \bigcup \left\{ \left(\tilde{V}_{2i+1}, \tilde{V}_{2j}\right) \right\}, \forall \left(V_i, V_j\right) \in E \tag{6}$$

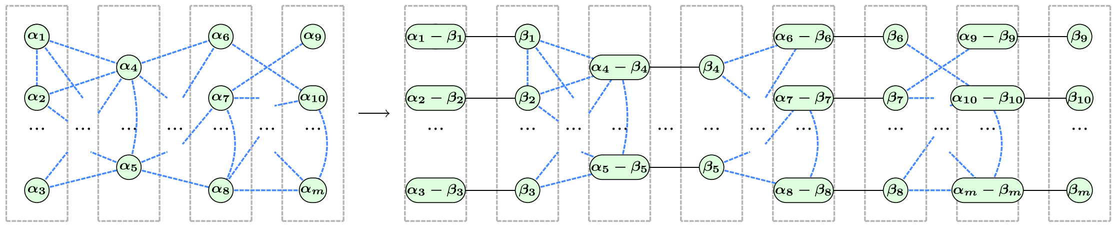
$$\tilde{E}_{\mathrm{N}} = \bigcup \left\{ \tilde{V}_{2i}, \tilde{V}_{2i+1} \right\}, \forall V_i \in V. \tag{7}$$

And the new phase $\tilde{\alpha}_i$ is given by:

$$\tilde{\alpha}_{2i} = \alpha_i - \beta_i \tag{8}$$

$$\tilde{\alpha}_{2i+1} = \beta_i \tag{9}$$

where β_i is a random phase value. $\tilde{G}$ can be rewrite to G by applying the $(\boldsymbol{f})$ to merge spider $\tilde{V}_{2i}$ and $\tilde{V}_{2i+1}$. Therefore $\tilde{G}$ and G are equivalent. Since $\tilde{\alpha}_{2i} = \alpha_i - \beta_i + b_{2i}\pi$, $\tilde{\alpha}_{2i+1} = \beta_i + b_{2i+1}\pi$, when β_i is chosen uniformly random, $\tilde{\alpha}_{2i}$ or $\tilde{\alpha}_{2i+1}$ is independent from α_i. Since $\tilde{\alpha}_{2i}$ and $\tilde{\alpha}_{2i+1}$ only dependent to α_i and β_i, each single of them is independent to G. This rewrite only uses $(\boldsymbol{f})$, from lemma 3.9, it preserves the Pauli flow. □



(b)Original diagram $\mathcal{G}$

(b)After phase obfuscation

Example 3.11. To illustrate the phase obfuscation strategy, consider (a) the original graph-like ZX diagram describing the original algorithm. Each spider V_i with phase α_i is split into two spiders connected with a regular wire. A random phase value β_i is generated, and the phase for two new spiders $\tilde{\alpha}_{2i} = \alpha_i - \beta_i$, $\tilde{\alpha}_{2i+1} = \beta_i$, results in (b).

After this construction, we could show that $D(\tilde{V}_{2i+1}) < D(\tilde{V}_{2i})$.

3.3. Connectivity obfuscation

Recall that in the phase obfuscation step, we have turned each spider into a pair of spiders connected with a regular wire, and two spiders in the pair now belong to different blocks. The wires within each block can be rewritten into a wire between the adjacent block. Such wire can be rewritten by disconnecting it from one spider and connecting it to the spider in an adjacent block which has a regular wire connected to the just disconnected from.

Theorem 3.12 (internal connectivity to external connectivity). *Given graph $G = (E_{\mathrm{N}}, E_{\mathrm{H}}, V)$ where $E_{Hi,j} = \{\tilde{V}_{2i}, \tilde{V}_{2j}\}$ is an wire connect two spiders $\tilde{V}_{2i}$ and $\tilde{V}_{2j}$ in the same block. $E_{Hi,j}$ can be replaced with $\{\tilde{V}_{2i+1}, \tilde{V}_{2j}\}$, where $\tilde{V}_{2i}$ and $\tilde{V}_{2i+1}$ are connected with an regular wire. The rewrite rule preserves the Pauli flow of G.*

Proof. The graph can be constructed with the following rewrite presented in figure 2. The rewrite uses only $(\boldsymbol{f})$, from lemma 3.9 it preserves Pauli flow. □

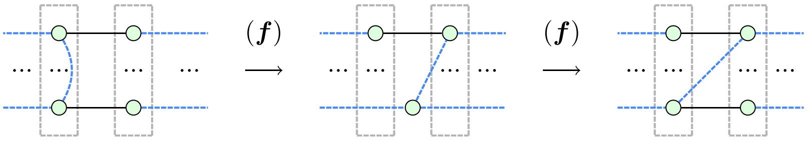


Figure 2. Proof of Theorem 3.12.

Theorem 3.13 ([36, 37]). *Let $G = (V, E)$ be a graph with vertices V and edges E. Suppose the labeled open graph (G, I, O), and $\lambda(u) \in \{XY, X\}$ for all $u \in O^c$, has Pauli flow. Pick an edge $v, w \in E$ and subdivide it twice (see figure 3), i.e. let $G' := (V', E')$ where $V' := V \cup v', w'$ contains two new vertices v', w', and*

$$E' = (E / \{\{v, w\}\}) \cup \{\{v, w'\}, \{v', w'\}, \{v', w\}\}. \tag{10}$$

Then (G', I, O, λ') has Pauli flow, where

$$\lambda'(u) := \begin{cases} \lambda(u), & \text{if } u \in V/O \\ X, & \text{if } u \in \{v', w'\}. \end{cases} \tag{11}$$

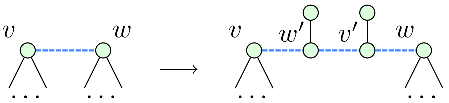


Figure 3. The Rewrite rule for Theorem 3.13.

Theorem 3.14 (obfuscate the connectivity between blocks). *Given a semi-graph ZX diagram $G(E_\mathrm{N}, E_\mathrm{H}, V)$ generated from phase obfuscation, an equivalent, Pauli flow preserving semi-graph ZX-diagram $\tilde{G}(\tilde{E}_\mathrm{N}, \tilde{E}_\mathrm{H}, \tilde{V})$ and a block partition B can be found, such that for edges $\tilde{E} = (\tilde{V}_i, \tilde{V}_j)$ within the same partition $\tilde{V}_i \in B_k, \tilde{V}_j \in B_k$, all edges depends only on* $\deg(V_i)$, $V_i \in V$.

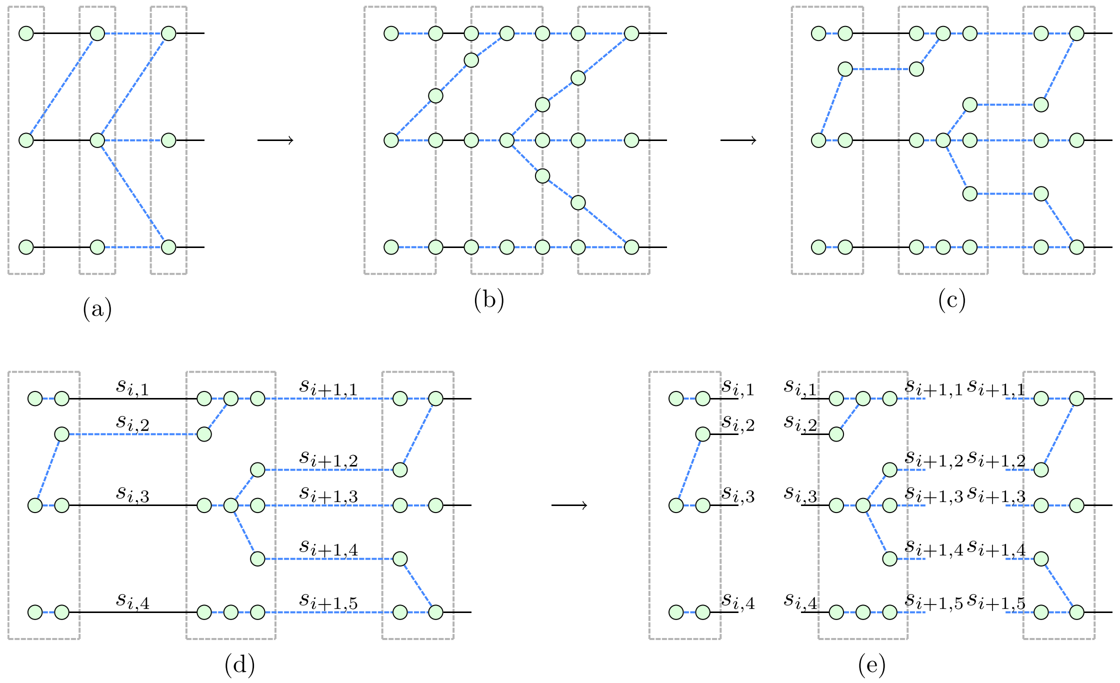
Proof. To find $\tilde{G}$, we first apply the rewrite rule from theorem 3.12 and remove wires within the blocks. Then for all $E_{i,j} = \{V_i, V_j\} \in E_\mathrm{H}$, we apply rewrite rules from 3.13. Denote the newly added spider for each edge $E_{i,j}$ as $W_{i,j}$ and $W'_{i,j}$. Assign $W_{i,j}$ to block $B(V_i)$ and $W'_{i,j}$ to $B(V_j)$.

After this rewrite, all the edges within the blocks connect to an extra spider $W_{i,j}$ and $W'_{i,j}$. For any two spiders W_A and W_B connect to V_i, they can be only distinguished by the external connection. Therefore all the edges $(V_i, W_{i,j})$ and $(V_j, W'_{i,j})$ within the same block does not depends on E_H. However since multiple W spiders may still connect to a single V spider, the internal wires depend on the degree of spider V_i and V_j.

Because the rewrite rule used in obfuscation is from theorems 3.12 and 3.13 and they both preserve Pauli flow, the rewrite for obfuscation also preserves Pauli flow. □

The connectivity obfuscation restricts each agent to have only the label of edges $s_{i,j}$ instead of the actual qubit connected in the adjacent agents. Such obfuscation prevents the connectivity configuration of the ZX-diagram from being reconstructed. After the obfuscation, the leg connects to hub spiders are all in an equivalent position; therefore agent cannot distinguish the direction of information flow during the execution. In practice, the $s_{i,j}$ can be used to identify the pre-shared Bell pairs between the agents. Each agent would not be able to know the entanglement structure of other agents.

The connectivity obfuscation step hides the agent's other end of the entanglement. However, the actual required entanglement can be estimated by the agent by counting the number of entanglements within its block, connected to some *hub spiders*. To further obfuscate the resource requirement of the quantum algorithm, we need to modify the number of connectivity of these hub spiders. We could add dummy qubit resources to obfuscate the exact resource requirement of the quantum algorithm. This can be done by attaching two phase-free spiders to the existing graph and connecting them with Hadamard edges.



Example 3.15. The process of connectivity obfuscation between blocks can be illustrated as follows. Suppose a fragment of the quantum algorithm looks like (a). Here all the phases in the spiders are not included in the diagram. First extra spiders are created with rule **(*f*)** to ensure no two inter-block wires are connected to the same spider, see (b). Then extra dummy spiders are created to obfuscate the spiders' degree (wire connected to the same spider). Each dummy spider has only two wires and would connect another dummy spider or a hub spider. See (c). Now for each inter-block wire, a random id $s_{i,k}$ is generated for its identification, see (d). The spiders' order can be randomly shuffled in each block, as long as the connectivity remains the same. See (e). Two adjacent blocks are assigned to a different quantum agent. Each agent only needs the wire identity $s_{i,k}$ to establish the correct entanglement.

Theorem 3.16 ([36, 37]). *Let $G = (V,E)$ be a graph with vertices V and edges E. Suppose the labeled open graph (G,I,O), and $\lambda(u) \in \{XY, X\}$ for all $u \in O^c$, has Pauli flow. Pick a node $u \in E$ and append two new vertices connected by a Hadamad edge (See figure 4), i.e. let $G' := (V',E')$ where $V' := V \cup v, w$ contains two new vertices v', w', and*

$$E' = E \cup \{\{v,w\},\{w,w'\}\}. \tag{12}$$

Then (G',I,O,λ') has Pauli flow, where

$$\lambda'(u) := \begin{cases} \lambda(u), & \text{if } u \in V/O \\ X, & \text{if } u \in \{w,w'\}. \end{cases} \tag{13}$$

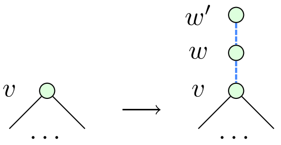


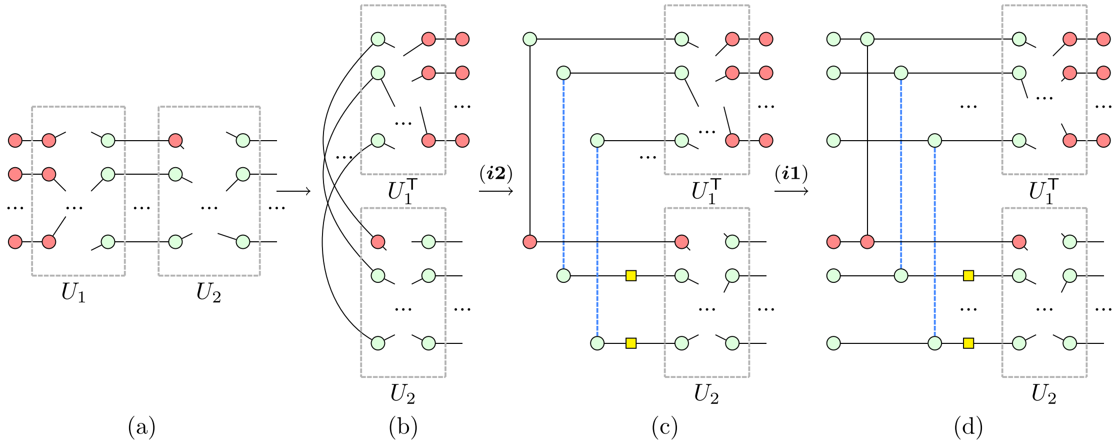
Figure 4. The rewrite rule for Theorem 3.16.

Theorem 3.17 (dummy resources). *Given a ZX-diagram $G(E_\mathrm{N},E_\mathrm{H},V)$ and partition B generated from connectivity obfuscation. An equivalent, Pauli flow preserved graph $\tilde{G}(\tilde{E}_\mathrm{N},\tilde{E}_\mathrm{H},\tilde{V})$ and a partition $\tilde{B}$ can be found, such that the hub spider has a larger degree.*

Proof. Suppose we want to increase the degree of spider $V_i \in B_j$. Apply the rewrite rule from theorem 3.16 to the spider V_i, and we have two added spider w and $w^{'}$. Assign w to B_{j-1} and $w^{'}$ to B_{j-2}. The rewrite graph is equivalent to the original graph, preserves the Pauli flow, and increases the degree of V_i. □

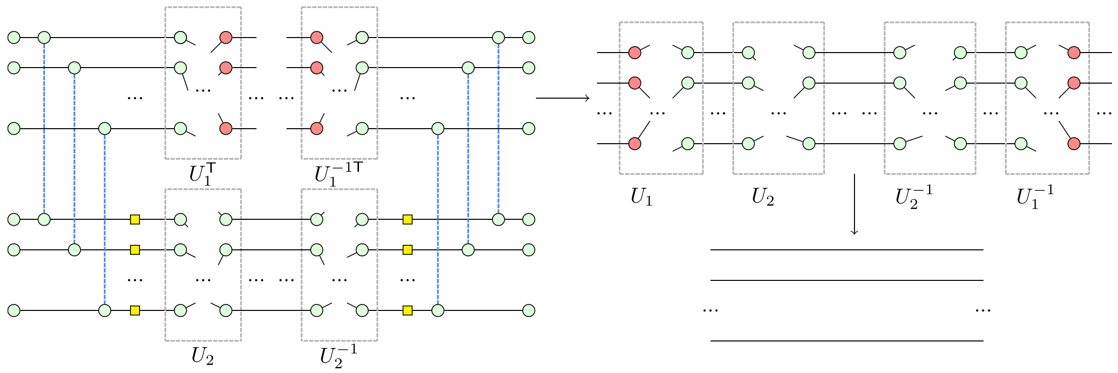
3.4. Measurement result obfuscation

So far, we have made input information to each agent independent of the algorithm. However, each agent may obtain information from their measurement outcome. Here we show some information may leak out from the measurement distribution of intermediate blocks, if the measurement outcome is not further obfuscated. To illustrate it, first consider a circuit is folded into two piece and each part of the ZX-diagram is executed on a different agent. See example 3.18.



Example 3.18. Fold a quantum circuit. On (a) we rewrite our quantum circuit into a ZX-diagram and then divide them into two quantum circuits noted as U_1 and U_2. Then we fold the ZX-diagram in (b) and rewrite the folded connection between U_1 and U_2 in (c). Eventually, we add the initial state of the quantum circuit in (d). The ZX-diagram in (d) is ready to be extracted into a quantum circuit with shallower depth but used twice as the original quantum circuit.

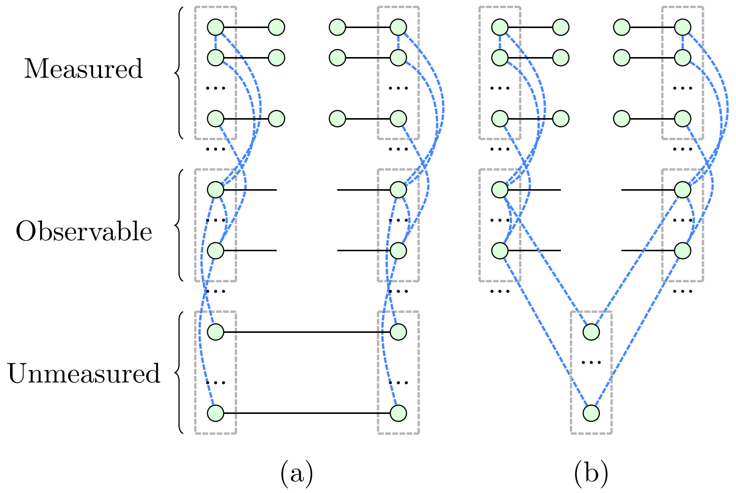
Now we would like to understand the measurement result distribution of the agent executing the upper half of the diagram. See example 3.19.



Example 3.19. The reduced density matrix of a folded circuit. The reduced density matrix can be expressed by adding a dual of the existing quantum circuit and connecting the qubits that need to be traced away. Here we can show that after tracing away the qubits containing the computation result, the ZX-diagram of the reduced density matrix is an identity. This identity indicates that the measurement distribution of these qubits is uniform.

The graph-like ZX-diagram can be considered folding the circuit until it has only one operation before it gets measured. We can apply the same analysis to our protocol.

The observed distribution of each agent is characterized by the reduced density matrix. A non-uniform distribution indicates a potential information leakage. Now we introduce the measurement result obfuscation approach to resolve the information leakage from the non-uniform measurement outcome distribution.

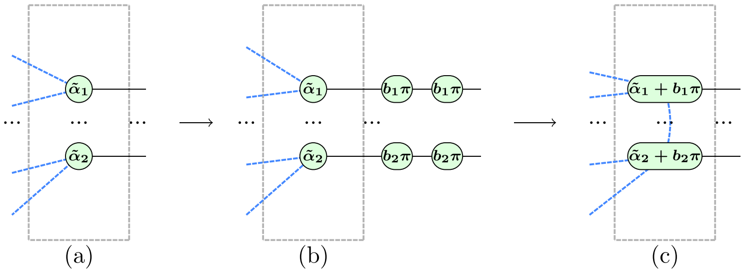


(a) (b)

Example 3.20. Information leakage from correction. Here we show that there could be information leakage from the agent's observation when correction is applied. To understand each agent's measurement distribution, we first arrange the spiders into the same column. Then we generate its conjugate diagram next to the existing diagram. The observed distribution described by the reduced density matrix can be obtained by tracing away the unmeasured qubits. However, we always apply the correction to restore the quantum state for measured qubits. Therefore it is equivalent to measuring the zero-phase spider on these qubits. The reduced density matrix is shown in (a). Now we try to simplify the quantum circuit; most of the unmeasured qubits can be traced away. However, it still leaves a graph that is not necessarily identity—shown in (b). The non-identity graph indicates the agent can observe a non-uniformed distribution, which may carry useful information.

Theorem 3.21 (measurement result obfuscation). *Given ZX-diagram $G(E_\mathrm{N}, E_\mathrm{H}, V)$ executes with MBQC methods. For each spider $V_i \in V$ with phase α_i, generate a random bit b_i and define $\tilde{\alpha}_i = \alpha_i + b_i\pi$, the measured distribution is independent of the diagram and the distribution for the calculation result can be reconstructed classically by the client.*

Proof. Suppose for each shot a new diagram $\tilde{G}(\tilde{E}_\mathrm{N}, \tilde{E}_\mathrm{H}, \tilde{V})$ is constructed at the execution time. Consider the measurement outcome for measuring spider $\tilde{V}_i$ is $\tilde{r}_i$. The result of executing $\tilde{G}$ is equivalent to G when we consider $r_i = \tilde{r}_i \oplus b_i$. Since b_i is chosen randomly, the distribution of $\tilde{r}_i$ is random and independent to the diagram G. □



(a) (b) (c)

Example 3.22. To illustrate the obfuscation of the readout distribution for each qubit, we introduce a bit string b_i. Suppose a fragment of the ZX diagram is shown as (a). For each V_i, we add two connected spiders with phase $b_i\pi$. This is equivalent to adding $2\pi b_i$ to each V_i as (b). Then one of the spiders is removed and converted into a classical flip operation. Merge the other spiders, and we have a new diagram as (c). For each $\tilde{V}_i$, we add a π phase if b_i is 1, otherwise, keep it the same.

To understand this more easily, consider a quantum circuit that generates a binary distribution. Such distribution can be hidden by randomly applying a π rotation to the qubit, swapping the probability of $|0\rangle$ and $|1\rangle$ just before the measurement. Then the original distribution can be restored by classically swapping them back.

Suppose the distribution without measurement obfuscation is $\tilde{r}_j$, and the distribution after measurement obfuscation is r_j. The extra π phase swaps the distribution of measuring the qubit with π phase or zero phases. Therefore $r_j = \tilde{r}_j \oplus b_j$, where $\oplus$ denote the bit-wise exclusive or operation. If b_i is chosen uniformly random, the measured result would be uniform. The correction process would be intuitive: $\tilde{r}_j = r_j \oplus b_j$. If we have measured a π phase and have already added a π phase to the spider, it cancels out if we have measured a zero phase and have added a π phase to the spider, it is equivalent to measuring a π phase without modifying the phase of the spider.

Theorem 3.23 (measurement result independence). *The distribution of measurement results r_i is independent of the executed quantum algorithm.*

Proof. The measurement result without measurement obfuscation $\tilde{r}_j$ can be non-uniform. The measurement result observed by each agent is $r_j = \tilde{r}_j \oplus b_j$. With the b_j chosen uniformly random, the measurement distribution of r_j would be uniform. □

3.5. Circuit extraction

So far, we have generated a ZX-diagram, which needs to be extracted into physical quantum operations. Note that the graph-like ZX-diagram contains only Hadamard wires. The regular wires between blocks come from the phase obfuscation step when each spider is split into two and connected with regular wires. So each spider is connected to a maximum of one regular wire to spiders at other agents.

The extraction can be implemented with the following method:

Theorem 3.24. *With a given obfuscated graph $\tilde{G}(\tilde{E}_{\mathrm{N}}, \tilde{E}_{\mathrm{H}}, \tilde{V})$ where $\mathrm{size}(\tilde{E}_{\mathrm{N}}) = 1$, i.e. each node would connect to multiple Hadamard edges and maximum one regular edge. $\tilde{G}$ can be extracted into physical quantum operations and implemented on a quantum device. The extracted physical operation to implement an edge is independent of the edge type.*

Proof. Each edge in graph $\tilde{G}$ can be extracted into two equivalent quantum operations, with different appraoch of placing the Hadamard gate (See figure 5). By randomly choosing the which operation that the diagram being extracted to, whether the Hadamard gate exists is independent of the type of wire. □

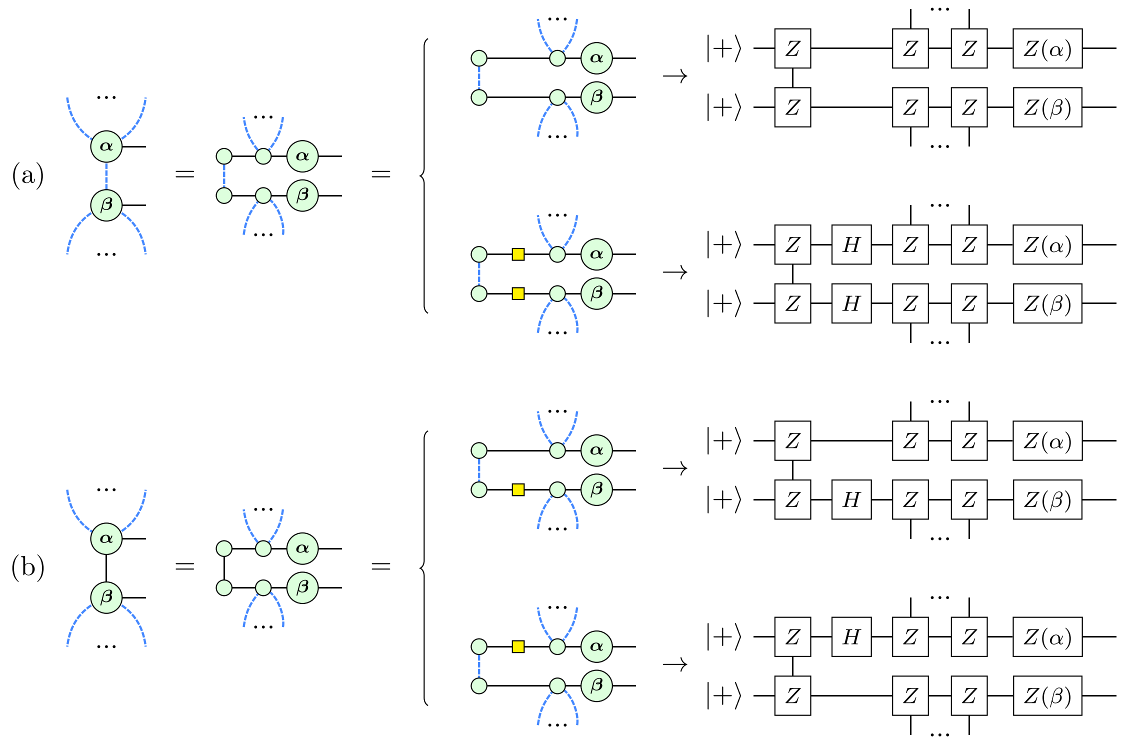


Figure 5. Proof of Theorem 3.24. (a) Hadamard wires can be extracted into a CZ operation, or applying CZ operation first, then applying Hadamard gate on both qubits. (b) Regular wires can be extracted into a CZ operation and then apply a Hadamard gate on only one side.

3.6. Summarize the protocol

The information describing the quantum algorithm consists of the phases of each spider and the connectivity between spiders under the perspective of ZX-diagram. Also, the measurement outcome would cause information leakage when the correction process is applied. Our protocol provides a complete solution to obfuscate information from these three aspects. First, our protocol utilizes the same strategy as the UBQC protocol to obfuscate the phase information and the measurement outcome. The phase rotation operation is split into two and performed by different agents. Then the measurement outcome is obfuscated by randomly flipping the quantum distribution with a phase difference of π. The rotation phase evaluation happens during the execution process to update the correction into the phase in real time. Finally, for the connectivity, our proposal moves all the connectivity information that reveals the algorithm as a wire between two different agents to hide the connectivity of the diagram. Since each agent cannot access the information from its neighboring agent, it loses track of the information on the other side of the wire. Here we present the formal description of our protocol. It contains two major components: The client's preparation step, which obfuscates and generates proper ZX-diagram blocks for each agent. Then, in the execution step, the client interacts with each agent to implement calculations and retrieve results.

Our protocol does not assume the input state is fully classical. For the case that the input state contains quantum data, it can be prepared by teleporting the quantum data to the agents, then make the teleport data into a segment of the ZX-diagram for computations. The client can remain fully classical to handle the quantum data by relying on a trusted third party to supply the quantum data.

4. Proof of correctness and blindness

In this section, we go through the techniques used to protect the information and give proof of the correctness and blindness of our protocol. First, we provide the definition of blindness.

Definition 4.1 (blindness). Let P be a quantum delegated computation on input X and let $L(X)$ be any function of the input. We say that a quantum delegated computation protocol is blind while leaking at most $L(X)$ if, on client's input X, for any fixed $Y = L(X)$, the following two hold when given Y

1. The distribution of the classical information obtained by an agent in P is independent of X.
2. Given the distribution of classical information described in 1, the state of the quantum system obtained by an agent in P is fixed and independent of X.

Definition 4.1 is proposed in [6] as a formal description to characterize blindness. Here X denotes information that the agent can obtain, and $L(X)$ is any information that can be inferred from given X. Now that $Y = L(X)$ is given, the agent cannot infer any algorithm information if the protocol is blind. The agent has two sources of information: the instructions it receives and the measurement outcome it gets. The first source suggests that the classical instructions obtained by the agent must be independent of the algorithm, and the second source suggests that the quantum information or measurement outcome must be independent of the algorithm.

We show the blindness of our protocol by proving the independence between the quantum algorithm being executed and the information each agent has access to.

Theorem 4.2 (inter-block connectivity independence). *Distribution of $s_{i,k}$ is independent of the executed quantum algorithm.*

Proof. $s_{i,k}$ is only used to identify the preshared entanglement pairs; its choice is independent of the quantum algorithm. □

Theorem 4.3 (inner-block connectivity leakage). *Distribution of $E(\tilde{V_i}, V_{i'}), V_i, V_{i'} \in B_m$ can leak at most* $\max(\deg(V_i))$.

Proof. For $E(\tilde{V}_i, \tilde{V}_{i'})$, the agent A_m can recover the $\deg(\tilde{V}_i)$ by reversely apply rule ***(f)***. With the extra dummy connectivity introduced, the degree recovered here is not necessarily the exact degree from the original algorithm, however, it is always greater or equal to $\deg(\tilde{V}_i)$.

□

Theorem 4.4 (safety of the correction process). *The correction process does not leak information.*

Proof. The correction process requires the client to modify α_i based on the measurement result of previous spiders. The information may leak out from the connectivity of the ZX-diagram, the phase information of each spider, and the measurement outcome distribution. We now discuss each aspect separately.

Protocol 1. MBQC BQC without universal cluster state.

Inputs.

1. $\mathcal{G}(E,V)$: a graph-like ZX-diagram describes the quantum algorithm Λ. E, V denote the connections and spiders in the graph. For classical data described input state, the preparation circuit is included in $\mathcal{G}$.
2. ρ_0: the input state of the quantum algorithm.
3. $A=\{A_1\ldots A_m\}, m\geqslant 2$: the available quantum agents. The number of agents m can be chosen arbitrarily, provided it is greater than or equal to 2.

Definitions.

1. $\tilde{\mathcal{G}}(\tilde{E},\tilde{V})$. the processed ZX-diagram for execution.
2. V_i: the ith spider in $\mathcal{G}$.
3. α_i: the phase of ith spider in $\mathcal{G}$.
4. $\tilde{V}_j$: the jth spider in $\tilde{\mathcal{G}}$.
5. $\tilde{\alpha}_j$: the phase of jth spider in $\tilde{\mathcal{G}}$.
6. β_j: the random value generated for phase obfuscation for V_i.
7. b_i: a random bit for measurement obfuscation of ith qubits.
8. $B_k=\{\tilde{V}_j\}$: the kth block of $\tilde{\mathcal{G}}$.
9. $d_j=d(\tilde{V}_j)$: the distance of V_j to its nearest output spider in $\tilde{\mathcal{G}}$.
10. $n=\max(d_j)$: the number of fragmented blocks in $\tilde{\mathcal{G}}$.
11. r_j: The measurement result of $\tilde{V}_j$.
12. $\tilde{r}_j$: The corresponding measurement result of $\tilde{V}_j$ without measurement obfuscation.

Goal. Retrieve the measurement distribution of $\Lambda(\rho_0)$.

a. Preparation.

1. The client split each $V_i\in V$ spider into two spiders $\tilde{V}_{2i}$ and $\tilde{V}_{2i+1}$ with rule $(\boldsymbol{f})$, each spider has phase $\tilde{\alpha}_{2i}$ and $\tilde{\alpha}_{2i+1}$. Note that $\tilde{\alpha}_{2i}$ and $\tilde{\alpha}_{2i+1}$ are symbols for placeholder, the actual value of will be evaluated in the later steps.
2. The client rearrange the ZX-diagram and group spiders into n blocks $B_k=\tilde{V}_j$ where the distance to output spider $d_j=k$.
3. For each connectivity within the same block, the client uses rule $(\boldsymbol{f})$ as example 3.12 to move it to the adjacent block.
4. The client split each Hadamard edge between blocks into two empty spiders and three edges, as shown in example 3.14. The two spiders are assigned to the block that their neighbor spider belongs to.
5. The client analyze $\tilde{G}$ and find a flow.

b. Execution.

1. The client assign block B_j to agent A_i when j mod $m=i$. Fragment block B_j are found by the rules from section 3.1.
2. For each sample
   - 2.1. The client generate random phase values $\{\beta_i\}$ and random bit $\{b_j\}$.
   - 2.2. The client assign $\tilde{\alpha}_{2i}=\alpha_i-\beta_i+b_{2i}\pi$ and $\tilde{\alpha}_{2i+1}=\beta_i+b_{2i+1}\pi$.
   - 2.3. The client randomly assigns spiders to qubits and allocates resources from each agent.
   - 2.4. Agents reset all qubits in all the blocks into $|+\rangle$*state*. For quantum data, teleport the input state into the input qubits and set all the other qubits into $|+\rangle$ state.
   - 2.5. The client extracts the ZX-diagram into quantum operations with example 3.15. Then request agents to establish shared entanglement between agents based on $\{s_i,j\}$.
   - 2.6. Follows the flow of G to execute the diagram. Handle V_i in ascending order of the partial order of the flow. For all spiders $\tilde{V}_j$ that splits from V_i
     - i. The client sends $\tilde{\alpha}_j$ to the corresponding agent, requesting the agent to measure qubits in XY plane with angle of $-\tilde{\alpha}_j$.
     - ii. The client get results r_j, calculate the $\tilde{r}_j=b_j\oplus r_j$.
     - iii. The client calculates the effectively measured phase $r=\sum r_j$ on V_i,
     - iv. The client make changes to α_j for correction based on r and the ZX-diagram with example 2.10.
3. The client returns the sampled distribution of $\tilde{r}_o$ where $\tilde{V}_o$ is a output spider.

1. Correction does not change the connectivity between spiders; therefore it does not invalidate theorems 4.2 or 4.3.
2. Note that the value of α_i updates with the measurement outcome from previous steps for correction, and from theorem 3.10, $\tilde{\alpha}_{2i}$ and $\tilde{\alpha}_{2i+1}$ is independent after the correction process updates α_i. Therefore, the correction process will not invalidate the independence between the phase and the actual algorithm.
3. The correction process would change the distribution of the measurement outcome. However, from theorem 3.23, each agent could not obtain any information from its measurement result.

Therefore, the correction process does not leak information. □

Theorem 4.5 (extraction universality). *ZX-diagram generated from the proposed protocol can always be extracted into practical quantum operations for real-world devices.*

Proof. The original graph-like ZX diagram was converted by a quantum circuit. Therefore, it must admit a focused Pauli flow [28] and can be executed with MBQC [10]. All the rewrite rules used in our protocol preserve the Pauli flow; therefore the obfuscated diagram must also admit a Pauli flow. Hadamard wires can be implemented into a CZ gate to extract the ZX-diagram into quantum operations. The regular wire only comes from splitting the spiders. So, each spider can have at most one regular wire. Such diagrams can be extracted with method form example 3.24. These rules included all possible diagrams that can be generated from our protocol. □

Theorem 4.6 (universality and correctness). *The modified ZX-diagram $\tilde{\mathcal{G}}$ is universal for quantum computation, can always be implemented on a quantum device, and yields the same distribution as $\mathcal{G}$.*

Proof. Any arbitrary ZX-diagram $\mathcal{G}$ with flow can be converted to $\tilde{\mathcal{G}}$ following the protocol and $\mathcal{G}$ is universal for quantum computation. Therefore $\tilde{\mathcal{G}}$ is universal and yields the same result as $\mathcal{G}$. Then from theorem 4.5, any $\tilde{\mathcal{G}}$ generated from $\mathcal{G}$ with our protocol preserves its flow and can be extracted into quantum operations can be executed on a quantum device. □

Definition 4.7 (ϵ-private [20]). A delegated quantum computation protocol requires the implementation of a linear map Φ_i on agent $A_i \in \mathbf{A}$ given classical information q_i. A simulator $\mathcal{S}_i$ has the same input and output space as $\mathbf{A} - A_i$, which can simulate the interaction between A_i and $\mathbf{A} - A_i$. The agent A_i interacts with $\mathcal{S}_i$, producing a linear map Ψ_i. The protocol is ϵ-private if for every agent A_i there exists such simulator $\mathcal{S}_i$ that $||\Phi_i - \Psi_i||_\diamond < \epsilon$, where $||\Phi_i - \Psi_i||_\diamond$ denote the diamond distance between Φ_i and Ψ_i.

Theorem 4.8 (private). *Our protocol is 0-private.*

Proof. The graph of each agent constructed gives the Choi–Jamiołkowski state $J(\Phi_i)$ of the linear map Φ_i [38], see figure 6. The information obtained by an agent is $q_i = \{G_i, \{s_i\}\}$, where $G_i(E_\mathrm{N}, E_\mathrm{H}, V)$ is the graph fragment assigned to agent A_i. From theorems 3.10, 3.23 and 4.2, q_i is randomly distributed and independent to the quantum algorithm for execution. Therefore $J(\Phi_i)$ is a mixed state with some layout restrictions from constructing connectivity obfuscation in theorems 3.23 and 4.2. Consider a simulator $\mathcal{S}_i$ that keeps the pre-shared entanglement pairs but does nothing on them. See the figure below. The layout of the graph representing the corresponding Choi–Jamiołkowski state $J(\Psi_i)$ (the graph in the right solid square) is in fact, identical to $J(\Phi_i)$ (in the left solid square), therefore $||J(\Psi_i) - J(\Phi_i)|| = 0$.

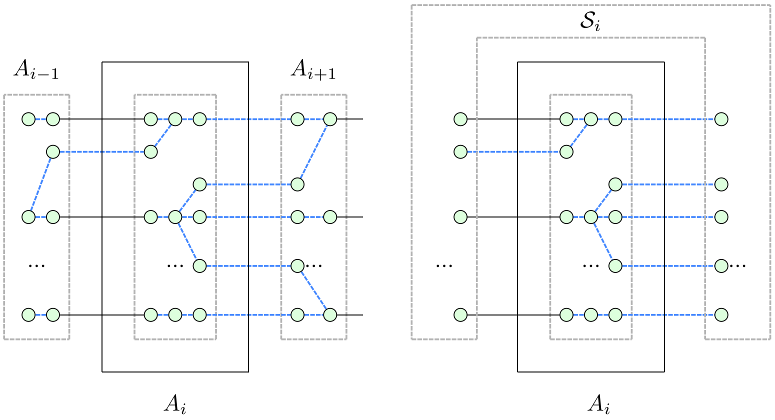


Figure 6. Proof of Theorem 4.8.

From relation $\frac{1}{n}||\Phi_i - \Psi_i||_\diamond < ||J(\Psi_i) - J(\Phi_i)||$ [39], where n is the size of the system, we conclude for our protocol is 0-private. □

Theorem 4.9 (blindness). *Our protocol is 0-private, and the information leakage would be at most* (max(deg($\tilde{V}_i$)),$N(B_k)$,n) *where* deg($\tilde{V}_i$) *is the degree (number of wires connected to a spider) of* $\tilde{V}_i$, max(deg($\tilde{V}_i$)) *is the maximum possible degree that* V_i *could have.* $N(B_k)$ *is the qubit number of block* B_k, n *is the of fragmented blocks in* $\tilde{G}$.

Proof. Client's input for each agent A_m consists of $N(B_k)$,n, $\tilde{\alpha}_i$, $s_{i,j}$ for all $V_i \in B_m$ and $V_j \in B_l$, where B_l is all adjacent blocks of B_m, $E(\tilde{V}_i, \tilde{V}_{i'})$ for $V_i, V_{i'} \in B_m$.

1. From theorem 3.10, $\tilde{\alpha}_i$ is independent from the algorithm.
2. From theorem 4.2, $s_{i,j}$ is independent from the algorithm.
3. From theorem 4.3, at most $\max(\deg(\tilde{V}_i))$ can be inferred by agent from the distribution of $E(\tilde{V}_i, \tilde{V}_{i'})$.
4. From theorem 3.23, the measurement distribution of each qubit is independent of the algorithm.
5. From theorem 4.4, the correction process does not leak information.
6. Each agent may know the total number of agents m, and infer the total block number n.
7. From example 3.24, each node can be extracted into quantum operation with or without a Hadamard gate. The existence of the Hadamard gate is independent of the algorithm.
8. From theorem 4.8, our protocol is 0-private.

Therefore, A_m can get only $(\max(\deg(\tilde{V}_i)), N(B_k), n)$ from the classical information it gets. □

The same as the UBQC protocol would inevitably disclose the size of the brickwork cluster state, our protocol also discloses some information about the resources required of the algorithm. UBQC uses a universal cluster state, which provides some surpluses of entanglement; therefore, UBQC protocol does not need to worry about the leakage of $\max(\deg(\tilde{V}_i))$. Our proposal optimized the resource requirement, which discloses more information about required resources. However, such information can be hidden by allocating more resources and doing random operations on extra resources as long as it will not affect the computation result.

The secureness of our protocol requires that communication between different agents is limited. Except for the shared entanglement generated in advance, agents should not exchange any information during the execution. Such an assumption is difficult to be fulfilled indefinitely since two agents need to share entanglement. When there are collusive agents, blindness may be compromised. Here, we show that the blindness of our protocol would be compromised only when adjacent blocks are executed on two collusive agents.

Theorem 4.10 (blindness compromise from collusive agents). *Information may leak out only when two adjacent blocks are executed on collusive agents.*

Proof. When the attacker obtains information on two adjacent blocks, the attacker can apply the $(\boldsymbol{f})$ rule to reverse the spider splitting and find the rotation angle or find a portion of connectivity in the original ZX-diagram. When attackers obtain information from non-adjacent blocks, it is equivalently to assign those non-adjacent blocks to the same agent. The attacker obtained the information from that single agent. Therefore from theorem 4.9 the information can be recovered is still $(\max(\deg(\tilde{V}_i)), N(B_k), n)$. □

Although the proposed protocol only requires pre-shared Bell pairs between agents, no information needs to be exchanged between agents at the run time.

Instead of physically limiting communication, the assumption can still be fulfilled with a decentralization strategy. For example, two quantum agents can be allocated from two different quantum service providers, and therefore it would be less likely to have two providers collude and compromise the blindness. More agents can also be introduced to have less chance of two adjacent blocks executed on collusive agents. Such relaxation is relatively weak since other strategies might be available if agents are honest and only exchange information the client allows. This relaxation allows the information exchange between agents even while executing the algorithm. Our protocol requires no information exchange between agents after the initial cluster state has been prepared. Our protocol would still be functioning if there were physical methods that could limit the communication between agents discovered in the future.

5. A minimal example of our protocol

In this section, we walk through a minimal example to implement a two qubits swap-test algorithm with the Hong–Ou–Mandel model [40], see figure 7. This algorithm does a CNOT gate and a Hadamard gate. The state overlap can be calculated based on the joint distribution of O_1 and O_2. We ignore the measurement obfuscation step for simplicity.

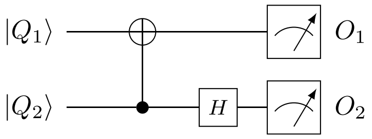


Figure 7. The original circuit.

The circuit is written into the ZX-diagram , see figure 8.

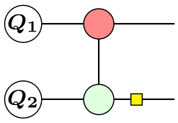


Figure 8. Obfuscation step 1: Rewriting to a ZX-Diagram.

And then converted into a graph-like ZX-diagram, see figure 9. Note that $\alpha_1 = \alpha_2 = 0$.

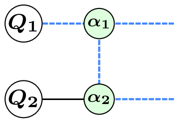


Figure 9. Obfuscation step 2: Rewriting to a graph-like ZX-Diagram.

Now, split each spider into two to make phase obfuscation. The value of $\tilde{\alpha}_1$ and $\tilde{\alpha}_3$ can be random, as long as $\tilde{\alpha}_2 = -\tilde{\alpha}_1$, $\tilde{\alpha}_4 = -\tilde{\alpha}_3$. See figure 10.

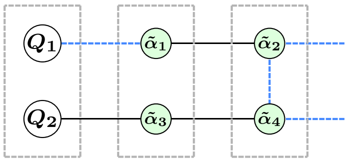


Figure 10. Obfuscation step 3: Phase obfuscation.

Move the connectivity within the same block to another spider, making it an inter-block connectivity, see figure 11.

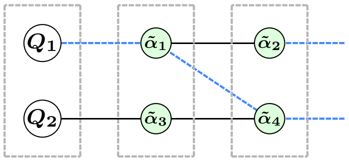


Figure 11. Obfuscation step 4: Internal connectivity to external connectivity.

Grows extra spider and finish the connectivity obfuscation, see figure 12.

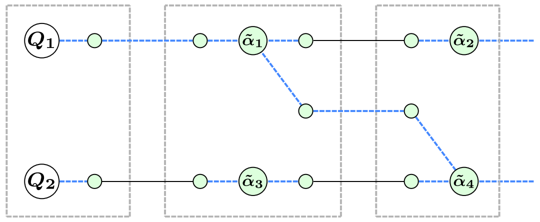


Figure 12. Obfuscation step 5: Obfuscating the connectivity between blocks.

To construct the diagram above, each agent only requires a shared Bell state at the beginning, see Figure 13.

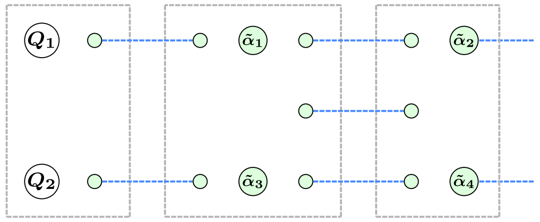


Figure 13. Execution step 1: Preparing shared bell pairs.

Then apply the Hadamard gate as an example 3.24 to convert Hadamard edges to normal edges, see figure 14.

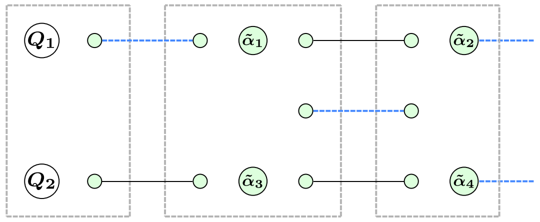


Figure 14. Execution step 2: Converting pre-shared bell pairs to the correct edge types.

Then apply entanglement operation within each agent, see figure 15.

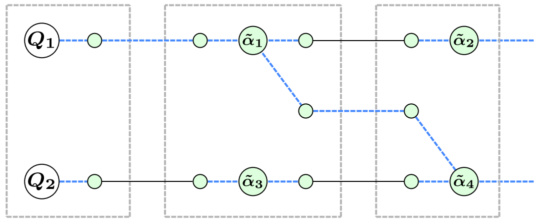


Figure 15. Execution step 3: Building obfuscated ZX-Diagram.

To execute the algorithm, here we follow the standard MBQC protocol. First, the regular edges are merged, and the new spider represents the sum of the phase from two old spiders as example 3.6. These spiders are labeled with the red star symbol, see figure 16.

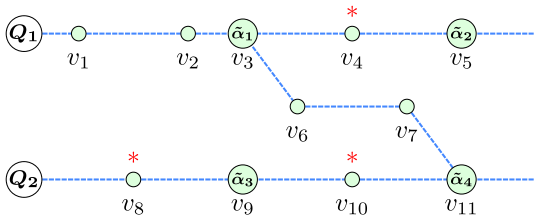


Figure 16. Execution step 4: Determining the measurement order.

Here we present a Pauli flow and the corresponding correction set. Define partial order

$$\prec := \{(v_1, v_2, v_6, v_7, v_8, v_8, v_4, v_{10}) < (Q_1, Q_2)\} \bigcup \{(Q_1, Q_2) < (v_3, v_9)\} \bigcup \{(v_3, v_9) < (v_5, v_{11})\} \tag{14}$$

where we define $A < B := \bigcup \{(a,b)\}, \forall a \in A$ and $\forall b \in B$.

Vertex u	Measurement plane $\lambda(u)$	Correction set $f(u)$	Odd($f(u)$)
Q_1,	XY	$\{v_1, v_3, v_5, v_7\}$	$\{Q_1, v_{11}\}$
Q_2,	XY	$\{v_8\}$	$\{Q_2, v_9\}$
v_1	X	$\{v_2\}$	$\{Q_1, v_3\}$
v_2	X	$\{v_3, v_5, v_7\}$	$\{v_2, v_{11}\}$
v_3	XY	$\{v_4\}$	$\{v_3, v_5\}$
v_4	X	$\{v_5\}$	$\{v_4\}$
v_5	XY	N/A	N/A
v_6	X	$\{v_7\}$	$\{v_6, v_{11}\}$
v_7	X	$\{v_6\}$	$\{v_3, v_7\}$
v_8	X	$\{v_9, v_{11}\}$	$\{v_8\}$
v_9	XY	$\{v_{10}\}$	$\{v_9, v_{11}\}$
v_{10}	X	$\{v_{11}, v_6\}$	$\{v_{10}, v_3\}$
v_{11}	XY	N/A	N/A

Based on the Pauli flow configuration, the algorithm can be executed with four steps. First all the qubits corresponds to vertices$\{v_1, v_2, v_6, v_7, v_8, v_8, v_4, v_{10}\}$ and then measure $\{Q_1, Q_2\}$,then $\{v_3, v_9\}$, and finally $\{v_5, v_{11}\}$. As an example here we demonstrate the correction when v_2 is measured with unexpected results, see figure 17. First, we highlight all the vertices from the correction set of v_2.

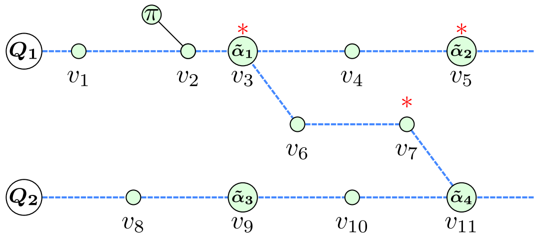


Figure 17. Implementing correction step 1. Determining correction set.

Now we apply (π) and $(i1)$ and (f) to emit red spiders (See figure 18) with pi phase on all the vertices in the correction set (See figure 19).

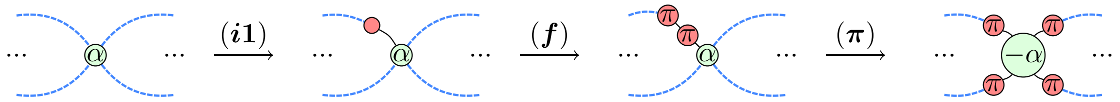


Figure 18. Implementing correction step 1: Generating spiders from a neighbor in the correction set.

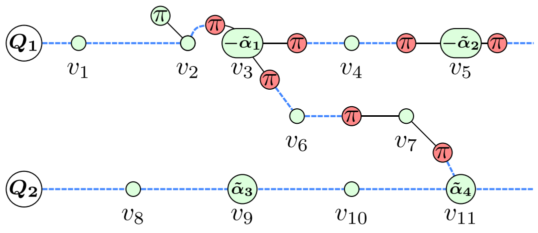


Figure 19. Implementing correction step 2: Generating spiders from the entire correction set.

Now push these red spiders through the Hadamard edge, and they become green spiders, see figure 20.

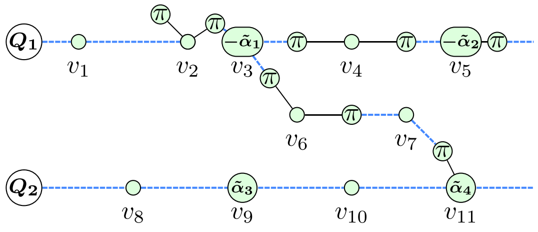


Figure 20. Implementing correction step 3: Propergating the spiders.

Merge green spiders to cancel the unexpected measurement result, see figure 21.

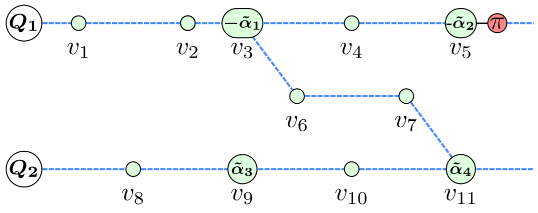


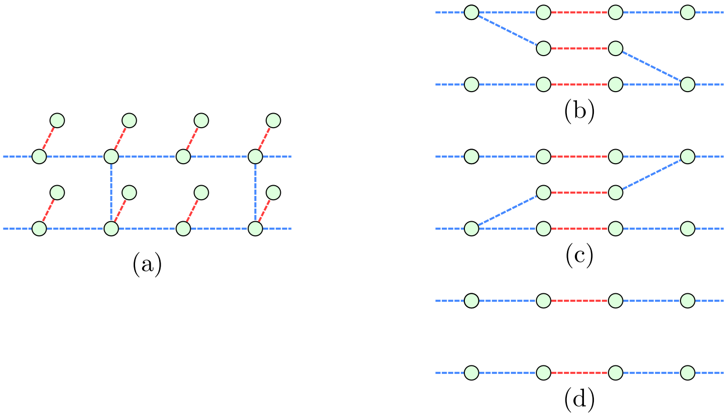
Figure 21. Implement correction step 4: Merging the spiders.

6. Discussion

6.1. Resource cost comparison between UBQC and our protocol

In this section, we quantify the resource requirements for implementing the UBQC and our proposed protocols and demonstrate the significant advantage of resource cost reduction compared to the UBQC protocol. Our proposed protocol distinguishes itself from the UBQC protocol primarily in how it implements connectivity obfuscation of the quantum algorithm. In the context of GBQC, connectivity refers to the layout of the quantum circuit, while in MBQC, connectivity is represented by the edges of the cluster state. In the ZX diagram, connectivity is depicted using a similar representation as wires between spiders. The resources we consider include the total number of qubits utilized by the protocol, the quantity of pre-shared Bell pairs necessary to establish entanglement between agents, and the overall number of two-qubit entanglement gates required within each agent.

The UBQC protocol achieves connectivity obfuscation by creating identical graphs for all algorithms, requiring the preparation of a universal cluster state that is algorithm-independent and information-free. Such universal cluster states restrict information to be stored in the measurement angles, thereby introducing redundancy to the graph. In contrast, our proposed protocol obfuscates connectivity by separating the two endpoints of the wires into two agents, enabling the layout of the ZX-diagram to carry information. Each agent is aware of a pre-shared Bell pair between itself and a neighboring agent, however, it is uncertain which qubit the Bell pair is entangled with, resulting in a loss of information carried by the entanglement of the Bell pair. An example of resource reduction is shown in example 6.1:



Example 6.1. An example of comparing the required resources to implement an arbitrary gate on two qubits. Here we denote the blue line as the entanglement generated from local entanglement gates within an agent, and the red lines are entanglement established by pre-shared Bell pairs or teleportation between agents. Note that this gate does not have to be a two-qubit entanglement gate; it can be two single qubit gate acts on two qubits separately. (a) The brickwork cluster state is used in the UBQC protocol for implementing a quantum gate. (b)–(d) The implementation of the protocol proposed in this study. The entanglement gate can be implemented by both (b) and (c), and the two single qubit gates can be implemented by (d). The number of qubits and entanglement gates and pre-shared Bell pairs from our protocol (b)–(d) are less than UBQC protocol (a). The required measurement steps are one fewer than those in the brickwork state. As each column can be measured together, each block showing in (b)–(d) requires a maximum of four steps to execute. On the other hand, the brickwork state (a) incorporates an additional step: measuring all remotely entangled qubits (attached by red-colored edges) to implement the teleportation. As for the number of measurements executing each block, the brickwork typically requires measuring 16 qubits, whereas our protocol requires a maximum of 10.

Now we move on to more general cases. For simplicity, we suppose the algorithm is decomposed into a gate set containing only local single-qubit gates and a CZ gate on nearest neighbors. These gates can be implemented directly with one 'brick', the fundamental component in the brickwork cluster state. Denote d as the circuit depth, w as the circuit width or the number of qubits and t as the number of two-qubit gates. For the worst that all gates are two-qubit gates, there are $\frac{1}{2}dw$ two-qubit gates. Therefore $t \leqslant \frac{1}{2}dw$. Here the entanglements between different agents are established by pre-shared Bell pairs, and entanglements within the same agent are implemented by entanglement gates performed by the agent. The transmission of quantum data between agents and clients is also considered using a Bell pair.

For the brickwork state, each 'brick' includes eight qubits and eight internal entanglement operations. Each 'brick' hosts two qubits from the original algorithm. For both single-agent and multi-agent versions, each qubit needs to share a Bell pair with another agent or the client. In total, the qubits required to implement brickwork state is $\frac{1}{2}w \times 8 \times d + w = (4d+1)w$. For the semi-classical client UBQC, the brickwork state can be constructed and executed in sequence to recycle qubits. This strategy reduces the requirement qubit number to $2w+1$. The amount of Bell pairs is $(4d+1)w$. The amount of local entanglement gates is $\frac{1}{2}w \times 8 \times d$. For the single-agent version, an extra qubit is required; for the multi-agent version $(4d+1)w$ qubits are required.

For our proposal, without any simplification, each qubit is split into three, and the qubits in the input and output blocks are split into two. Each two-qubit gate requires two more qubits, two more internal entanglements, and one extra external entanglement. For each dummy connection, two extra qubits, one Bell pair, and two local entanglements gates are required. Therefore our proposal requires $3(d-2)w+2t$ qubits, $(d-1)w+t$ Bell pairs, $2(d-2)w+2t$ local entanglement gates.

See the table below to summarize the comparison.

Number of	Single-agent UBQC	Multi-agent UBQC	Our proposal
Agents	1	$\geqslant 2$	$\geqslant 2$
Qubits	$2w+1$	$2(4d+1)w$	$3(d-2)w+2t$
Bell pairs	$(4d+1)w$	$(4d+1)w$	$(d-1)w+t$
Local entanglement gates	$8dw$	$8dw$	$2(d-2)w+2t$

There are extra advantages to our protocol compared to UBQC. First, our protocol can implement non-nearest-neighbor entanglement directly. With the UBQC protocol, two qubits must be swapped to an adjacent position to perform the two-qubit gate, which adds extra cost to the implementation. Secondly, although the brickwork cluster state is universal, it is not intuitive to directly implement gates such as controlled single-qubit arbitrary rotation. Such gates can be decomposed into a ZX-diagram and directly implemented. Also, the quantum circuits can be simplified first with existing techniques from ZX-calculus [28] before applying our protocol. The graph-like ZX-diagram can be optimized until it only contains nodes representing non-Clifford operations. Such optimization can be considered the classically simulatable part of the quantum algorithm simplified from the diagram. Suppose the non-Clifford operation count is c, then the optimal qubit number would be $2c$, and the entanglement number would depend on the algorithm. Such a method can significantly reduce the resource requirement of our protocol.

6.2. Compatibility with existing verification protocols

The universality of ZX-diagram provides compatibility with most of the existing verification protocols. However, since some verification protocols require a universal cluster state, combining these verification protocols would invalidate our resource requirement advantage compared to the UBQC protocol. Here we discuss the 'first-order' compatibility of verification protocols [41]. The rigorous compatibility and full analysis of security with detailed proof, however, is beyond the topic of this paper.

Verification can be implemented by embedding a quantum circuit into the original algorithm that gives a deterministic result when the algorithm has been faithfully computed. The authentication-based verification method [42] extends the quantum authentication schemes as the embedded circuit. The trap-based verification method [5, 43] utilize tapped wires or stabilizer codes for the embedded circuit. These embedded circuits can be converted into ZX-diagram and processed with our protocol. It is worth mentioning that work from [34] makes it even more convenient to embed the stabilizer codes based on ZX-diagram.

Verification can also be implemented with the run and test scheme. The agent is asked to do calculations multiple times. The client randomly selects some of these calculations as test runs that run an algorithm in which the measurement distribution is known. The proposal from [18] suggests running the circuit in different initial states indistinguishable from the agent and using some of them as the tests. This method is compatible with our protocol since the initial state can be prepared arbitrarily and indistinguishable from the agent. The proposal from [44] implement the test run with the same cluster state as the computation but modifies the measurement angle. Since our protocol no longer uses a universal cluster state, this proposal is invalid. However, we can still use the ZX calculus to find phases for the same diagram layout but it gives a known probability distribution. If each spider's phase is chosen carefully, the ZX-diagram can be efficiently simulated [28].

Verification can be implemented with entanglement-based protocols. Proposals from [45, 46] make use of CHSH games and proposals from [47] utilize a self-testing graph states for verification. These methods all use self-testing results and pass the verification when the winning rate agrees with the prediction of quantum mechanics. These proposals are all compatible with ours; however, the self-testing graph protocol requires implementing a complicated graph state, which would invalidate our advantage compared to the UBQC protocol.

7. Conclusion

We propose a multi-agent BQC protocol based on ZX-calculus in this work. The quantum algorithm is first written into a ZX-diagram and then modified to be extracted into an MBQC-style algorithm. Then the algorithm is executed across multiple agents. We show that the information leakage to every agent is minimal, and our protocol's security can be guaranteed under the assumption that communication between agents is limited. Our proposal does not require a universal cluster state compared to the UBQC protocol. This advantage makes our protocol more flexible and efficient.

Data availability statement

No new data were created or analysed in this study.

Acknowledgments

The author thanks the support from UKRI [EP/W027992/1], and the Eric and Wendy Schmidt AI in Science Postdoctoral Fellowship, a Schmidt Future program. We thank Miriam Backens for providing the proof of theorems 3.13 and 3.16 from her work before publication. We thank John van de Wetering for reviewing this work and providing constructive comments. We thank Anne Broadbent, Lia Yeh, Niel de Beaudrap, Quanlong Wang, Xiao Yuan, Brian Vlastakis, and Peter Leek for insightful discussions. We acknowledge the usage of the PyZX package [48] and the TikZiT tool. For the purpose of Open Access, the author has applied a CC BY public copyright license to any Author Accepted Manuscript version arising from this submission.

ORCID iD

Shuxiang Cao https://orcid.org/0000-0001-7178-4250

References

[1] Krantz P, Kjaergaard M, Yan F, Orlando T P, Gustavsson S and Oliver W D 2019 A quantum engineer's guide to superconducting qubits *Appl. Phys. Rev.* **6** 021318

[2] Bruzewicz C D, Chiaverini J, McConnell R and Sage J M 2019 Trapped-ion quantum computing: progress and challenges *Appl. Phys. Rev.* **6** 021314

[3] Kloeffel C and Loss D 2013 Prospects for spin-based quantum computing in quantum dots *Annu. Rev. Condens. Matter Phys.* **4** 51–81

[4] Slussarenko S and Pryde G J 2019 Photonic quantum information processing: a concise review *Appl. Phys. Rev.* **6** 041303

[5] Fitzsimons J F 2017 Private quantum computation: an introduction to blind quantum computing and related protocols *npj Quantum Inf.* **3** 23

[6] Broadbent A, Fitzsimons J and Kashefi E 2008 universal blind quantum computation *Proc.—Annual IEEE Symp. on Foundations of Computer Science (FOCS)* pp 517–26

[7] Michielsen K, Nocon M, Willsch D, Jin F, Lippert T and De Raedt H 2017 Benchmarking gate-based quantum computers *Comput. Phys. Commun.* **220** 44–55

[8] Briegel H J, Browne D E, Dür W, Raussendorf R and Van den Nest M 2009 Measurement-based quantum computation *Nat. Phys.* **5** 19–26

[9] Raussendorf R, Harrington J and Goyal K 2006 A fault-tolerant one-way quantum computer *Ann. Phys., NY* **321** 2242–70

[10] Gross D, Eisert J, Schuch N and Perez-Garcia D 2007 Measurement-based quantum computation beyond the one-way model *Phys. Rev.* A **76** 052315

[11] Kissinger A and van de Wetering J 2019 universal MBQC with generalised parity-phase interactions and Pauli measurements *Quantum* **3** 134

[12] Nielsen M A 2006 Cluster-state quantum computation *Rep. Math. Phys.* **57** 147–61

[13] Mahadev U 2022 Classical verification of quantum computations *SIAM J. Comput.* **51** 1172–229

[14] Brakerski Z 2018 Quantum FHE (almost) as secure as classical *Advances in Cryptology—Crypto 2018* ed H Shacham and A Boldyreva (Springer International Publishing) pp 67–95

[15] Cojocaru A, Colisson L, Kashefi E and Wallden P 2019 Qfactory: classically-instructed remote secret qubits preparation *Advances in Cryptology—Asiacrypt 2019* ed S D Galbraith and S Moriai (Springer International Publishing) pp 615–45

[16] Morimae T and Fujii K 2013 Blind quantum computation protocol in which Alice only makes measurements *Phys. Rev.* A **87** 050301

[17] Sano Y 2020 Blind quantum computation using a circuit-based quantum computer (arXiv:2006.06255)

[18] Broadbent A 2018 How to verify a quantum computation *Theory Comput.* **14** 1–37

[19] Fisher K A, Broadbent A, Shalm L K, Yan Z, Lavoie J, Prevedel R, Jennewein T and Resch K J 2014 Quantum computing on encrypted data *Nat. Commun.* **5** 1–7

[20] Broadbent A 2015 Delegating private quantum computations *Can. J. Phys.* **93** 941–6

[21] Coecke B and Kissinger A 2018 Picturing quantum processes *Diagrammatic Representation and Inference* (*Lecture Notes in Computer Science* vol 10871 LNAI) (Springer) pp 28–31

[22] Biamonte J and Bergholm V 2017 Tensor networks in a nutshell (arXiv:1708.00006)

[23] Peng T, Harrow A W, Ozols M and Wu X 2020 Simulating large quantum circuits on a small quantum computer *Phys. Rev. Lett.* **125** 150504

[24] Backens M 2016 Completeness and the ZX-calculus (arXiv:1602.08954)

[25] Jeandel E, Perdrix S and Vilmart R 2020 Completeness of the ZX-calculus *Log. Methods Comput. Sci.* **16** 72

[26] de Witt C S and Zamdzhiev V 2014 The ZX-calculus is incomplete for quantum mechanics *Electron. Proc. Theor. Comput. Sci.* **172** 285–92

[27] Backens M, Miller-Bakewell H, de Felice G, Lobski L and van de Wetering J 2020 There and back again: a circuit extraction tale (arXiv:2003.01664)

[28] Duncan R, Kissinger A, Perdrix S and van de Wetering J 2019 Graph-theoretic simplification of quantum circuits with the ZX-calculus (arXiv:1902.03178)

[29] Aaronson S and Gottesman D 2004 Improved simulation of stabilizer circuits *Phys. Rev.* A **70** 052328

[30] Danos V and Kashefi E 2006 Determinism in the one-way model *Phys. Rev.* A **74** 052310

[31] Browne D E, Kashefi E, Mhalla M and Perdrix S 2007 Generalized flow and determinism in measurement-based quantum computation *New J. Phys.* **9** 250–250

[32] Duncan R and Perdrix S 2010 Rewriting measurement-based quantum computations with generalised flow *Automata, Languages and Programming* (*Lecture Notes in Computer Science*) (Springer) pp 285–96

[33] Duncan R 2012 A graphical approach to measurement-based quantum computing *Quantum Physics and Linguistics* (Oxford University Press) pp 50–89

[34] Backens M 2013 The ZX-calculus is complete for stabilizer quantum mechanics *New J. Phys.* **16** 093021

[35] Duncan R and Perdrix S 2010 Rewriting measurement-based quantum computations with generalised flow *Automata, Languages and Programming* ed S Abramsky, C Gavoille, C Kirchner, F Meyer auf der Heide and P G Spirakis (Springer) pp 285–96

[36] McElvanney T and Backens M 2022 Complete flow-preserving rewrite rules for MBQC patterns with Pauli measurements (arXiv:2205.02009)

[37] McElvanney T and Backens M 2023 Flow-preserving ZX-calculus rewrite rules for optimisation and obfuscation (arXiv:2304.08166)

[38] Choi M-D 1975 Completely positive linear maps on complex matrices *Linear Algebr. Appl.* **10** 285–90

[39] Gilchrist A, Langford N K and Nielsen M A 2005 Distance measures to compare real and ideal quantum processes *Phys. Rev.* A **71** 062310

[40] Garcia-Escartin J C and Chamorro-Posada P 2013 SWAP test and Hong-Ou-Mandel effect are equivalent *Phys. Rev.* A **87** 052330

[41] Gheorghiu A, Kapourniotis T and Kashefi E 2019 Verification of quantum computation: an overview of existing approaches *Theory Comput. Syst.* **63** 715–808

[42] Aharonov D, Ben-Or M, Eban E and Mahadev U 2017 Interactive Proofs for Quantum Computations (arXiv:1704.04487)

[43] Fitzsimons J F and Kashefi E 2017 Unconditionally verifiable blind quantum computation *Phys. Rev.* A **96** 517

[44] Hayashi M and Morimae T 2015 Verifiable measurement-only blind quantum computing with stabilizer testing *Phys. Rev. Lett.* **115** 220502

[45] Gheorghiu A, Kashefi E and Wallden P 2015 Robustness and device independence of verifiable blind quantum computing *New J. Phys.* **17** 083040

[46] Reichardt B W, Unger F and Vazirani U 2013 Classical command of quantum systems *Nature* **496** 456–60

[47] McKague M 2016 Interactive proofs for BQP via self-tested graph states *Theory Comput.* **12** 1–42

[48] Kissinger A and van de Wetering J 2019 PyZX: large scale automated diagrammatic reasoning (arXiv:1904.04735)