2025.08.11

This state is again local unitarily equivalent to the well-known Greenberger-Horne-Zeilinger (GHZ) state, given by:

$$(H_1 \otimes I \otimes H_3)|C_3\rangle = \frac{|000\rangle_{1,2,3} + |111\rangle_{1,2,3}}{\sqrt{2}} =: |\text{GHZ}\rangle. \tag{3.11}$$

3.4.2 Quantum Wire

The most basic operation in MBQC is the "quantum wire," which transfers quantum information from one qubit to another. Let's say we have a qubit in some state:

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle. \tag{3.12}$$

After entangling with another qubit using a CZ gate, we obtain:

$$|\Psi\rangle = CZ(\alpha_0|0\rangle + \alpha_1|1\rangle) \otimes |+\rangle \tag{3.13}$$
$$= \alpha_0|0+\rangle + \alpha_1|1-\rangle. \tag{3.14}$$

Now let us analyze what happens when we measure the first qubit in the (Pauli-)X basis. Expanding the state $|\Psi\rangle$ in the $\{|+\rangle_1, |-\rangle_1\}$ basis and rearranging, we find:

$$|\Psi\rangle = \alpha_0|0+\rangle_{1,2} + \alpha_1|1-\rangle_{1,2} \tag{3.15}$$
$$= \frac{\alpha_0}{\sqrt{2}}|+\rangle|+\rangle + \frac{\alpha_0}{\sqrt{2}}|-\rangle|+\rangle + \frac{\alpha_1}{\sqrt{2}}|+\rangle|-\rangle - \frac{\alpha_1}{\sqrt{2}}|-\rangle|-\rangle \tag{3.16}$$
$$= |+\rangle\left(\frac{\alpha_0}{\sqrt{2}}|+\rangle + \frac{\alpha_1}{\sqrt{2}}|-\rangle\right) + |-\rangle\left(\frac{\alpha_0}{\sqrt{2}}|+\rangle - \frac{\alpha_1}{\sqrt{2}}|-\rangle\right) \tag{3.17}$$
$$= \frac{1}{\sqrt{2}}\left(|+\rangle H|\text{in}\rangle_2 + |-\rangle HZ|\text{in}\rangle_2\right), \tag{3.18}$$

where we used the relations $H|+\rangle = |0\rangle$ and $H|-\rangle = |1\rangle$.

When we measure the first qubit in the X basis and obtain outcome s (either 0 or 1), the second qubit collapses into:

$$|\text{out}\rangle = HZ^s|\psi\rangle. \tag{3.19}$$

This process works like quantum teleportation: the quantum information from the first qubit moves to the second qubit, with a Hadamard transformation applied and a possible Pauli-Z correction depending on what we measured. When we store logical information in qubit 1 and then measure it, you might expect that information to be lost forever since measurement is normally irreversible. But surprisingly, the entanglement between the two qubits allows the quantum information to completely transfer to the second qubit, preserving it despite the measurement. The measurement doesn't destroy the information - instead, it transforms how that information exists in the quantum system, relocating it from the first qubit to the second through the entanglement connection they share.

3.5 Implementation of Quantum Gates in MBQC

To establish the universality of MBQC, we need to demonstrate its ability to implement a universal set of quantum gates.

3.5.1 Implementation of Single-Qubit Unitaries

Any unitary operation $U \in SU(2)$ on a single qubit can be decomposed using the Euler decomposition:

$$U = R_z(\alpha)R_x(\beta)R_z(\gamma) \tag{3.20}$$

To implement this unitary in MBQC, we use a linear cluster state with appropriate measurements.

Implementation with Three Qubits

Consider a three-qubit linear cluster state where the first qubit contains the state $|\psi\rangle$ we wish to transform. When we measure the first qubit in the X-Y plane (in the basis of eigenstates of the observable $O(\alpha) = \cos(\alpha)X + \sin(\alpha)Y$), we obtain a result $s_1 \in \{0, 1\}$. By subsequently measuring the second qubit in an adaptive basis determined by outcome s_1, we can control the transformation.

Through mathematical transformations exploiting Pauli operator proper-

HERE, YOU DO MORE THAN THIS: THERE IS ONLY T, CNOT, H TO IMPLEMENT TO GET UNIVERSALITY.

• SUGGESTION: ADD UNIVERSALITY AS A SIDE REMARK AT THE END OF THIS SECTION.

all

of what?

Pub Hein Lakes.

ties, we derive:

$$|\text{out}\rangle = [HZ^{s_1}R_z(\alpha)][HZ^{s_2}R_z(\beta)]|\psi\rangle \tag{3.21}$$
$$= H(Z^{s_1}R_z(\alpha)H)Z^{s_2}|\psi\rangle \quad \text{(Associativity of unitaries)} \tag{3.22}$$
$$= X^{s_1}R_x(\alpha)Z^{s_2}|\psi\rangle \quad \text{(Using } HZH = X\text{)} \tag{3.23}$$
$$= X^{s_1}Z^{s_2}R_x((-1)^{s_2}\alpha)|\psi\rangle \quad \text{(Using } XZ = -ZX\text{)} \tag{3.24}$$

This demonstrates implementation of simple rotations through measurement.

Implementation with Five Qubits

For implementing complete single-qubit unitaries, we use a five-qubit linear cluster state as shown in Figure 3.3.

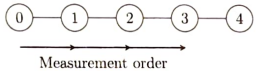


Figure 3.3: Linear cluster state for implementing a single-qubit unitary. Qubits 0-3 are measured in adaptive bases to implement the desired rotation, while qubit 4 contains the output state with the applied unitary transformation. Arrows indicate the sequence of measurements.

We perform sequential measurements on the first four qubits, leaving the fifth in the transformed state. The measurements are performed with respect to the observables:

$$O(\varphi_1 = 0) = X, \quad O(\varphi_2), \quad O(\varphi_3), \quad O(\varphi_4) \tag{3.25}$$

where φ_i are angles that depend on the specific unitary we want to implement. Through detailed calculations, we obtain:

$$|\text{out}\rangle = [HZ^{s_1}R_z(\varphi_1)][HZ^{s_2}R_x(\varphi_2)][HZ^{s_3}R_z(\varphi_3)][HZ^{s_4}R_z(\varphi_4)]|\text{in}\rangle \tag{3.26}$$
$$= X^{s_1}R_x(\varphi_1)Z^{s_2}R_x(\varphi_2)Z^{s_3}R_x(\varphi_3)Z^{s_4}R_x(\varphi_4)|\text{in}\rangle \tag{3.27}$$
$$= X^{s_1+s_3}Z^{s_2+s_4}R_x((-1)^{s_1+s_3+s_4}\varphi_4)R_z((-1)^{s_2+s_4}\varphi_3)R_x((-1)^{s_3}\varphi_2)|\text{in}\rangle \tag{3.28}$$

We can express this more compactly as:

$$|\text{out}\rangle = B(s)R_z((-1)^{s_1+s_3+s_4}\varphi_4)R_x((-1)^{s_2+s_4}\varphi_3)R_z((-1)^{s_3}\varphi_2)|\text{in}\rangle \tag{3.29}$$

where the byproduct operator $B(s) := X^{s_1+s_3}Z^{s_2+s_4}$ depends on measurement outcomes.

To obtain exactly the desired Euler decomposition, we adaptively choose measurement angles:

$$\varphi_2 = (-1)^{s_1}\gamma \tag{3.30}$$
$$\varphi_3 = (-1)^{s_2}\beta \tag{3.31}$$
$$\varphi_4 = (-1)^{s_1+s_3}\alpha \tag{3.32}$$

This adaptive approach counteracts measurement randomness, allowing deterministic implementation of any single-qubit unitary up to a byproduct operator that can be compensated in subsequent operations.

3.6 Determinism in MBQC

Quantum measurements, as we said, introduce intrinsic randomness that is in principle irreducible. The number of possible outputs of a measurement-based computation on a given input is exponential in the number of measurements. The only way to exploit this probabilistic computation is to implement a correction strategy that makes the overall computation deterministic. This strategy does not affect the probabilities of the measurement branches, but makes sure that all the branches produce the same output.

The fundamental question now is: how to decide if a given resource state on which performing the measurement-based quantum computation is deterministic or not?

A property that ensures the determinism of an MBQC resource state is referred to as *flow*, that was originally introduced in [20].

Recall that resource states in MBQC can represented as graph states, within these graph states we can identify different types of qubits: input qubits (where the computation begins), output qubits (where the final result is obtained), and other qubits in the graph. All qubits except the output

qubits are measured during the computation. On this graph structure, we can define a mathematical property called flow, which formalizes the conditions necessary for deterministic computation.

Definition 1 (Flow). *Let (G, I, O, λ) be an open graph state, where $G = (V, E)$ is an undirected graph, $I \subseteq V$ and $O \subseteq V$ are disjoint subsets representing input and output qubits respectively, and $\lambda : O^c \to \{(X,Y), (X,Z), (Y,Z)\}$ specifies the measurement plane for each non-output qubit. We denote by $N_G(v)$ the set of neighbors of vertex v in G, and by $O^c = V \setminus O$ and $I^c = V \setminus I$ the complements of O and I in V.*

An open graph state (G, I, O, λ) with $\lambda(i) = (X,Y)$ for all $i \in O^c$ possesses a flow if there exists a function $f : O^c \to I^c$ (mapping measured qubits to prepared qubits) and a partial order $>$ on V such that for all $i \in O^c$:

(F1) $i \sim f(i)$ (adjacency condition)

(F2) $i < f(i)$ (temporal ordering)

(F3) $\forall k \in N_G(f(i)) \setminus \{i\}$, $i < k$ (neighborhood ordering)

The first condition ensures that nodes are connected to their successors, the second ensures that nodes are measured before their successors, and the third ensures the neighbors of the successor of a node i are measured after i.

The traditional flow condition [20] provides a sufficient but not necessary condition for deterministic computation in MBQC, restricted to measurements in the (X,Y) plane and requiring each measured qubit to have exactly one correcting qubit. Generalized flow (gflow) [21] extends this definition by allowing measurements in the three planes (X,Y), (X,Z), and (Y,Z), and allowing each measured qubit to have a *set* of correcting the qubits rather than a single one.

3.6.1 Gflow

Generalized flow is a structural property of a graph state that ensures deterministic quantum computation in the MBQC model (*Theorem 2* in [21]).

Given an open graph state (G, I, O, λ), where:

- $G = (V, E)$ is a graph with vertex set V and edge set E
- $I \subseteq V$ is the set of input qubits,
- $O \subseteq V$ is the set of output qubits,
- $\lambda : V \setminus O \to \{(X,Y), (X,Z), (Y,Z)\}$ assigns a measurement plane to each measured qubit,

GFlow exists if there is a function $g : O^c \to P^{I^c}$ (mapping measured qubits to a subset of prepared qubits) and a partial order $<$ on V that satisfy the following conditions:

- (G1) If $j \in g(i)$ and $i \neq j$, then $i < j$, ensuring that qubit i is measured before it is used for correction.
- (G2) If $j \leq i$ and $i \neq j$, then $j \notin \mathrm{Odd}(g(i))$, preventing causal inconsistencies.
- (G3) If $\lambda(i) = (X,Y)$, then $i \notin g(i)$ and $i \in \mathrm{Odd}(g(i))$, ensuring proper correction in the (X,Y) plane.
- (G4) If $\lambda(i) = (X,Z)$, then $i \in g(i)$ and $i \in \mathrm{Odd}(g(i))$, adapting corrections for the (X,Z) measurement plane.
- (G5) If $\lambda(i) = (Y,Z)$, then $i \in g(i)$ and $i \notin \mathrm{Odd}(g(i))$, modifying correction dependencies in the (Y,Z) plane.

The function $g(i)$ determines the correction set for each measured qubit. The odd neighborhood of a set of vertices K in a graph G is defined as the set of vertices that have an odd number of neighbors in K, formally given by:

$$\mathrm{Odd}(K) = \{u \in V \mid |N_G(u) \cap K| \equiv 1 \pmod 2\}.$$

For example, in a triangle graph with vertices $\{1,2,3\}$ and edges $\{(1,2),(2,3),(3,1)\}$, if $K = \{2,3\}$, then $\mathrm{Odd}(K) = \{2,3\}$, since both vertices have an odd number of neighbors in K.

In the example in 3.4, the graph does not admit a standard flow because there is no way to assign each measured qubit to exactly one prepared qubit while also maintaining the required partial order. Specifically, vertex c cannot be matched to a single correcting qubit that satisfies all the flow conditions

don't introduce λ if you don't use it.

I suggest you do only the X-Y measurement. Just ignore the other measurement planes.

not really useful in my opinion...

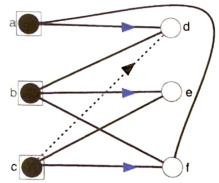


Figure 3.4: Example of a graph that has a generalized flow but no casual flow.

(F1–F3). However, a generalized flow (gflow) *does* exist because we can assign a *set* of correcting qubits to c: Each measured qubit i appears before all qubits in $g(i)$. For example, a is measured before d, and c is measured before both d and f, ensuring a valid ordering.

No earlier vertex in the ordering appears in $\mathrm{Odd}(g(i))$ unless it is already i itself. In this graph, the choice $g(c) = \{d, f\}$ ensures that the required parity conditions are satisfied without introducing conflicts with previous vertices.

Depending on the measurement plane (X, Y), (X, Z), or (Y, Z), the presence of i in $g(i)$ and the parity condition $i \in \mathrm{Odd}(g(i))$ are correctly enforced. Here, for instance, a, b, c are measured in the (X, Y) plane, meaning each satisfies $i \notin g(i)$

This ability to send one measured qubit to multiple correcting vertices is the key difference, enabling a deterministic measurement pattern where standard flow fails.

v	1	3	5
$g(v)$	{2}	{4}	{2,6}
$\mathrm{Odd}(g(v))$	{1,3}	{3,5}	{5}

Table 3.1: Assignment of the generalized flow function $g(v)$ and its odd neighborhood $\mathrm{Odd}(g(v))$ for selected vertices.

Chapter 4

Quantum Computing in NISQ Era

The term **NISQ** (*Noisy Intermediate-Scale Quantum*) was coined by John Preskill [54] in 2018 to define ~~the current class of quantum computing platforms~~ computations that can be performed in absence of fully error corrected machines.

- **Noisy:** refers to the influence of the environment, such as thermal noise, electromagnetic field noise, etc., which can result in large errors in a computation.
- **Intermediate-Scale:** Quantum computers that exist today contain 50 to a few hundred qubits. Although there are tasks for which this scale exceeds the reach of classical simulation, it is still far too small for the realization of full-scale quantum error correction.

The NISQ era represents quantum computing's transitional phase, reaching the point where devices can begin to perform certain tasks faster than classical machines - which we call a *quantum advantage* or *quantum supremacy*. The long-term goal continues to focus on building quantum hardware with lower gate error rates.

Classical computers are particularly inefficient at simulating the dynamics of highly entangled many-particle quantum systems, so *quantum dynamics* is a particularly promising area where quantum computers may achieve significant advantages [54].

Despite their potential, NISQ devices face several fundamental challenges that limit their computational capabilities and practical applications.

Main challenges for NISQ Devices

1. **High Error Rates:** Todayàs quantum computers exhibit error rates that are much higher than in conventional electronics and have insufficient quantum resources to support powerful error correction protocols [51].

2. **Short Coherence Times:** Coherence decay or decoherence is a key obstacle, as qubits spontaneously relax toward their ground state in microseconds to milliseconds, limiting the time during which computational operations can be successfully executed.

3. **Limited Qubit Numbers:** The number of available qubits is currently insufficient for executing full-fledged quantum error correction protocols, required to achieve fault-tolerant computation. Hundreds to thousands of physical qubits are still required for each logical qubit.

4. **Connectivity Constraints:** In most architectures, qubits cannot interact arbitrarily with each other, placing limits on the complexity and efficiency of quantum circuits that can be implemented.

5. **Measurement Errors:** Read-out fidelities remain at 99â99.8 % (mid-circuit even lower), often dominating total error budgets [5].

6. **Gate Calibration Drift:** Quantum devices require frequent recalibration as gate parameters drift over time due to environmental fluctuations and hardware aging.

7. **Limited Circuit Depth:** The combination of decoherence and gate errors restricts quantum circuits to shallow depths, typically 10-100 gates before errors dominate the computation.

Concerning the future of quantum computing, given the high error rates of its hardware, a first approach is to implement quantum error correction (QEC), where many physical qubits together constitute one single logical qubit, but more robust. Another possible approach is to accept the inherent noise and find applications that survive even taking it into account. One of these applications are Variational Quantum Algorithms.

4.1 Variational Quantum Algorithms (VQA)

Building upon the challenges and opportunities of the NISQ era discussed previously, Variational Quantum Algorithms (VQAs) emerges as practical approaches to extract computational value from current quantum hardware. These hybrid algorithms leverage both classical and quantum computing resources to find approximate solutions to optimization and eigenvalue problems, with applications spanning chemistry, finance, and logistics.

VQAs serve as a bridge between classical algorithms and the fault-tolerant quantum algorithms of the future. While current NISQ devices lack the qubit count, coherence times, and error rates needed for implementing full-scale fault-tolerant quantum algorithms, VQAs are specifically designed to work within these constraints.

Despite the limitations of today's quantum processors, researchers continue to explore whether meaningful computational advantages can still be achieved. As noted by Zapata AI, "Variational quantum algorithms provide a framework for attempting to use these 'along-the-way' quantum computers" [56].

For a broad perspective on VQAs - their challenges and future - read the paper in Nature Reviews Physics - [55].

On a high level, VQAs consist of four steps:

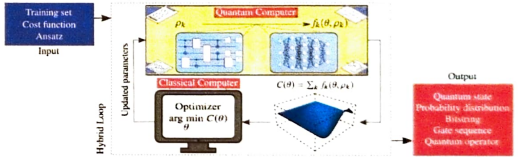


Figure 4.1: Schematic diagram of a Variational Quantum Algorithm (VQA) [55]

1. **Parameterized quantum circuit preparation:** Start with a parameterized quantum circuit (ansatz) in which a vector of parameters, $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)$, are rotation angles in radians associated to a given

set of quantum gates of the circuit. This gives us a mapping from parameters to quantum states, $\boldsymbol{\theta} \mapsto |\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|0\rangle^{\otimes n}$, where $U(\boldsymbol{\theta})$ is a sequence of parameterized unitary rotations.

2. **Gradient-Descent Optimization:** Optimize the parameters $\boldsymbol{\theta}$ by using classical optimization algorithms, such as gradient descent, in order to minimize a problem-specific cost function $C(\boldsymbol{\theta})$. This function is usually written as an expectation value of some Hamiltonian $C(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|\hat{H}|\psi(\boldsymbol{\theta})\rangle$, where $\hat{H}$ contains the problem to be solved.

3. **Convergence criterion:** Stop iterations whenever is the representative desired set of the learned dictionary. Convergence conditions often require observation $\nabla_{\boldsymbol{\theta}} C(\boldsymbol{\theta})$ or change in the cost function in successive evaluations $|C(\boldsymbol{\theta}_{i+1}) - C(\boldsymbol{\theta}_i)| < \epsilon$ for some ϵ.

4. **Solution extraction and post-processing:**. Using the optimized parameters $\boldsymbol{\theta}^\star$ to prepare the final state $|\psi(\boldsymbol{\theta}^\star)\rangle$ and classically post-process the state to extract the solution of the original eigenvalue or optimization problem. In the framework of eigenvalue problems the minimum value, $C(\boldsymbol{\theta}^\star)$, becomes an approximation of the ground state energy; in optimization problems the solution can be extracted by simulating sample measurements.

4.1.1 Variational Quantum Eigensolver (VQE)

The Variational Quantum Eigensolver (VQE) is one of the most important examples of variational quantum algorithms. Introduced by Peruzzo et al. in 2014 [57], addresses the challenge of determining the ground state energy of many-body quantum systems.

This algorithm is based on the variational principle of quantum mechanics. For a physical system described by a Hamiltonian H and any normalizable wave function $|\Psi\rangle$, we define the energy functional:

$$E[\Psi] = \frac{\langle\Psi|H|\Psi\rangle}{\langle\Psi|\Psi\rangle} \tag{4.1}$$

According to this principle, the energy expectation value is always greater than or equal to the true energy of the ground state E_0, with equality achieved only when $|\Psi\rangle$ exactly matches the ground state.

The insight of VQE is to leverage this principle using parameterized quantum circuits. By preparing a trial state $|\psi(\boldsymbol{\theta})\rangle$ on a quantum computer, we can implement a hybrid quantum-classical optimization loop to find parameters that minimize the energy.

The Hamiltonian is decomposed into Pauli operators:

$$\mathcal{H} = \sum_{i\alpha} h^i_\alpha \sigma^i_\alpha + \sum_{ij\alpha\beta} h^{ij}_{\alpha\beta} \sigma^i_\alpha \sigma^j_\beta + \dots \tag{4.2}$$

where h are real coefficients, Roman indices identify the subsystem on which the operator acts, and Greek indices identify the type of Pauli operator, e.g., $\alpha = x$.

One can exploit the linearity of quantum observables, it follows that:

$$\langle\mathcal{H}\rangle = \sum_{i\alpha} h^i_\alpha \langle\sigma^i_\alpha\rangle + \sum_{ij\alpha\beta} h^{ij}_{\alpha\beta} \langle\sigma^i_\alpha \sigma^j_\beta\rangle + \dots \tag{2}$$

The VQE approach can handle a large range of physical systems by working with Hamiltonians expressed as a polynomial number of terms. This includes electronic structure Hamiltonians from quantum chemistry, the quantum Ising model, the Heisenberg model, and more generally any k-sparse Hamiltonian. Quantum modules compute expectation values for individual Hamiltonian terms: the a sum of a polynomial number of expectation values of simple Pauli operators for a quantum state, multiplied by some real constants, and a classical unit aggregates these values and optimizes the variational parameters.

With an n-qubit state, a quantum device can efficiently compute expectation values for Hamiltonians of dimension $2^n \times 2^n$. Attempting similar calculations on classical hardware would encounter the N-representability problem, which is known to be computationally intractable (in the quantum complexity class QMA-Hard [61]). Quantum hardware circumvents this challenge by storing the global quantum state with exponentially fewer resources than would be required classically.

The algorithm has shown good results for small molecular systems and continues to develop with extensions for excited states, adaptive ansätze, and error mitigation techniques. As quantum hardware improves, the VQE is expected to tackle increasingly complex systems, potentially delivering some

of the first practical quantum advantages in scientific research [58] [59] [60].

4.2 VQE Implementation in MBQC

The measurement-based variational quantum eigensolver (MB-VQE) combines the principles of VQE and MBQC. It was introduced for the first time by Ferguson and Dellantonio in 2021 [1].
While traditional VQE requires sequential gate operations, the MB-VQE trades circuit depth for additional qubits, using measurements on preprepared resource states instead of deep-gate sequences.

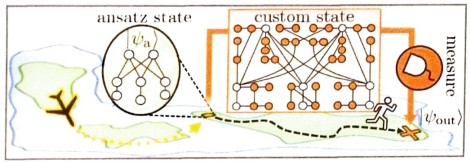


Figure 4.2: MB-VQE edge decoration approach, Figure from [1]

In [1] the authors introduce an approach for the implementation of MB-VQE called "edge decoration" 4.2 and addresses a limitation of circuit-based VQE: the restricted class of quantum states accessible through polynomial-depth circuits. Their method starts with an ansatz graph state $|\psi_a\rangle$ that provides a reasonable approximation to the target ground state. This graph state is a stabilizer state defined by the operators $\hat{S}_n = \hat{X}_n \prod_k \hat{Z}_k$, where k indexes all vertices connected to the site n.

The idea is to construct a custom state through edge decoration: for each pair of connected qubits in the original graph state $|\psi_a\rangle$, some auxiliary qubits are added and connected to form an expanded graph. These auxiliary qubits modify the entanglement properties of the output qubits when measured. The custom state includes both the original output qubits (white circles) and the new added auxiliary qubits (orange circles) (4.2).

Variational optimization occurs through measurements of auxiliary qubits in rotated bases $R(\theta) = \{(|0\rangle \pm e^{i\theta}|1\rangle)/\sqrt{2}\}$, and the measurement angles

θ serve as variational parameters optimized by the classical algorithm to minimize the energy expectation value.

This approach enables access to quantum state transformations that are expensive in the circuit model. For example, a single auxiliary qubit connected to m output qubits can implement the operation $e^{i\theta/2\hat{Z}_1\otimes\hat{Z}_2\otimes\cdots\otimes\hat{Z}_m}$ through one measurement. The equivalent circuit implementation would require $O(m)$ two-qubit gates.

However, there is a limitation in this approach: attaching multiple edge-wise decorations to a single ansatz vertex breaks the flow condition, so the computation is not deterministic [63].

This limitation was overcome by Schroeder et al. [63] in 2023, who introduced a deterministic approach called "node-wise decoration". Instead of connecting auxiliary qubits to edges, layers of qubits are connected vertically to create paths for measurement corrections.

The method creates copies of a base graph G_0, where each new layer G_l has the same connections as G_0 plus vertical links to the previous layer $(l-1)$. This builds a multi-level structure with identical horizontal patterns and vertical connections between layers.

This method guarantees deterministic results without needing expensive post-selection. Schroeder et al. tested their approach using simulations of realistic Hamiltonians (Schwinger model and XY-model) and experiments on IBM quantum hardware, showing it works well even on current noisy devices, especially when dealing with larger quantum systems. In the following chapter we analyze both the decorations schemes.

Chapter 5

Analysis

① We present the edge decoration scheme.
② We perform the theoretical analysis of the scheme (edge decoration)
③ We show where the problem is.
④ We propose a fix.
⑤ We report on the implementation of the fix.

We start this analysis from the edge decoration scheme in MBQC based on the work by Ferguson et al. [1].

• First, we characterize how auxiliary qubit measurements control the entanglement between the output qubits. We derive equivalent unitary representations of the decoration protocol, following the procedure in [34], which serve as the basis for numerical simulations before implementing full measurement-based simulation.

• Second, we address the critical limitation: decorated graph states lack generalized flow (gflow), preventing deterministic measurement-based computation. After analyzing gflow in both existing schemes (node-wise and edge-wise decoration), we propose a solution using dynamic qubit relabeling that preserves stabilizer structure while enabling flexible decoration sequences.

We test numerically this method on the perturbed toric code Hamiltonian, and then compare it with the node-wise decoration scheme [63].

put subsection ?

The edge decoration scheme applies to graph edges connecting computational qubits, as shown in Figure 5.1. By controlling auxiliary measurements, we can precisely tune two-qubit entanglement, providing building blocks for larger MBQC protocols.

The decoration introduces four auxiliary qubits, labeled $i = 1, 2, 3, 4$ (orange circles), to the two output qubits m and n (white circles). Both computational qubits m and n are initially prepared in the $|+\rangle$ state. The key insight is that measuring auxiliary qubits in a specific order and with specific angles directly controls the entanglement properties of the m–n system, allowing continuous tuning between extremal cases.

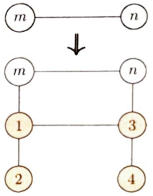


Figure 5.1: Edge decoration scheme between computational qubits m and n. The decoration transforms a simple edge into a structured measurement pattern involving four auxiliary qubits (orange circles).

The protocol enables tuning from the maximally entangled state, obtained by applying the controlled-Z gate to the initial product state:

$$\begin{aligned}
\mathrm{CZ}_{mn}|+\rangle_m|+\rangle_n &= \mathrm{CZ}_{mn}\frac{1}{2}(|0\rangle_m + |1\rangle_m)(|0\rangle_n + |1\rangle_n) \\
&= \frac{1}{2}\mathrm{CZ}_{mn}(|00\rangle_{mn} + |01\rangle_{mn} + |10\rangle_{mn} + |11\rangle_{mn}) \\
&= \frac{1}{2}(|00\rangle_{mn} + |01\rangle_{mn} + |10\rangle_{mn} - |11\rangle_{mn}) \\
&= |0\rangle_m|+\rangle_n + |1\rangle_m|-\rangle_n
\end{aligned}$$

to the completely separable state:

$$|+\rangle_m|+\rangle_n = \frac{1}{2}(|0\rangle_m + |1\rangle_m) \otimes (|0\rangle_n + |1\rangle_n)$$

The theoretical analysis proceeds through extraction of unitary operations equivalent to the measurement-based protocol. This extraction reveals the connection between adaptive quantum measurements and parametrized unitary circuits, providing both theoretical insight and practical implementation pathways.

5.1 Unitary Circuit Extraction

I think this should be included into a Todd Methods section.

To establish equivalence between the measurement-based protocol and conventional quantum circuits, I extracted the sequence of unitary gates that implements the same linear transformation as the measurement pattern. This extraction follows the procedure described by Simmons [34], with technical details in Appendix A.5. The method relies on systematic exploitation of stabilizer formalism and the focused generalized flow structure.

The first step requires identifying the focused gflow in the graph state of Figure 5.1. The correction sets, shown in Table 5.1, determine how measurement outcomes on individual qubits propagate corrections throughout the system.

v	$g(v)$
2	$\{1, n\}$
4	$\{3, m\}$
1	$\{m\}$
3	$\{n\}$

Table 5.1: Correction sets for the qubits in Figure 5.1, defining the focused generalized flow structure.

This gflow establishes the causal structure of the measurement protocol: measuring qubit 2 triggers corrections on qubits 1 and n, measuring qubit 4 affects qubits 3 and m, and so forth. The partial ordering $2, 4 < 1, 3$ [1] specifies that qubits 2 and 4 must be measured before qubits 1 and 3, ensuring the protocol remains deterministic despite probabilistic individual measurements.

The auxiliary qubits are measured in rotated bases within the XY plane. A measurement at angle α in the XY plane can be decomposed as a rotation followed by measurement in the computational basis. Let $|\pm_X\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$ denote the eigenstates of the Pauli-X operator with eigenvalues ± 1. The measurement eigenstates are obtained by rotating the X-basis states around the Z-axis:

$$|\pm_{XY,\alpha}\rangle = e^{-i\frac{\alpha}{2}Z}|\pm_X\rangle$$

The corresponding projection operators are:

$$\hat{P}_X^{\pm} = |\pm_X\rangle\langle\pm_X| = \frac{1}{2}(\mathbb{I} \pm X)$$
$$\hat{P}_{XY,\alpha}^{\pm} = |\pm_{XY,\alpha}\rangle\langle\pm_{XY,\alpha}| = e^{-i\frac{\alpha}{2}Z}\hat{P}_X^{\pm}e^{i\frac{\alpha}{2}Z}$$

Throughout this analysis, I use the compact notation $Z(\theta) = e^{-i\frac{\theta}{2}Z}$ for rotations around the Z-axis by angle θ. The stabilizer generators of the graph state in Figure 5.1 are:

$$S = \langle X_1 Z_m Z_2 Z_3, X_3 Z_1 Z_4 Z_n, X_m Z_n Z_1, X_n Z_m Z_3 \rangle$$

where the stabilizers of input qubits 2 and 4 are excluded, as these qubits will be consumed by measurement.

Following the established measurement order, the complete measurement sequence can be expressed as:

$$\prod_{i\in\{3,1,4,2\}} \hat{P}_{X_i}^{m_i} Z_i(\theta_i)|\psi, S\rangle = \hat{P}_{X_3}^{m_3} Z_3(\theta_3)\hat{P}_{X_1}^{m_1} Z_1(\theta_1)\hat{P}_{X_4}^{m_4} Z_4(\theta_4)\hat{P}_{X_2}^{m_2} Z_2(\theta_2)|\psi, S\rangle$$

where $|\psi, S\rangle$ represents the graph state stabilized by S, and $\hat{P}_{X_i}^{m_i} = \hat{P}_{X_i}^{+}$ if $m_i = +1$ and $\hat{P}_{X_i}^{m_i} = \hat{P}_{X_i}^{-}$ if $m_i = -1$.

The key insight for extracting equivalent unitary circuits lies in systematic exploitation of stabilizer projectors. For each stabilizer generator $C_j \in S$, we define the stabilizer projector $\hat{P}_{C_j} = \frac{1}{2}(\mathbb{I} + C_j)$. By the defining property of stabilizer states, this projector acts as the identity when applied to any state in the stabilizer space: $\hat{P}_{C_j}|\psi, S\rangle = |\psi, S\rangle$. This allows insertion of stabilizer projectors into the measurement sequence without altering the final result, after which they can be systematically commuted through the measurement operators using algebraic manipulation.

Consider the stabilizer $C_j = X_n Z_m Z_3$. Inserting its projector into the measurement sequence yields:

$$\hat{P}_{X_3}^{m_3} Z_3(\theta_3)\hat{P}_{X_1}^{m_1} Z_1(\theta_1)\hat{P}_{X_4}^{m_4} Z_4(\theta_4)\hat{P}_{X_2}^{m_2} Z_2(\theta_2)\hat{P}_{X_n Z_m Z_3}|\psi, S\rangle$$

Since the stabilizer projector commutes with all measurement operators acting on different qubits (due to the stabilizer property), it can be moved

forward to immediately follow the $Z_3(\theta_3)$ rotation:

$$\hat{P}^{m_3}_{X_3} Z_3(\theta_3) \hat{P}_{X_n Z_m Z_3} \hat{P}^{m_1}_{X_1} Z_1(\theta_1) \hat{P}^{m_4}_{X_4} Z_4(\theta_4) \hat{P}^{m_2}_{X_2} Z_2(\theta_2) |\psi, S\rangle$$

The crucial step involves applying the product rotation lemma, which enables commutation of rotations through projectors via operator conjugation. Specifically:

$$\begin{aligned} Z_3(\theta_3) \hat{P}_{X_n Z_m Z_3} &= e^{-i\frac{\theta_3}{2} Z_3} \hat{P}_{X_n Z_m Z_3} \\ &= e^{-i\frac{\theta_3}{2} Z_3 X_n Z_m Z_3} \hat{P}_{X_n Z_m Z_3} \\ &= e^{-i\frac{\theta_3}{2} X_n Z_m} \hat{P}_{X_n Z_m Z_3} \\ &= [X_n Z_m](\theta_3) \hat{P}_{X_n Z_m Z_3} \end{aligned}$$

where the identity $Z_3 X_n Z_m Z_3 = X_n Z_m$ follows from the anticommutation relation $\{Z_3, Z_3\} = 2\mathbb{I}$, which causes the Z_3 operators to cancel. Since the extracted unitary $[X_n Z_m](\theta_3)$ acts exclusively on qubits m and n while the measurement $\hat{P}^{m_3}_{X_3}$ acts on qubit 3, they commute, yielding:

$$[X_n Z_m](\theta_3) \hat{P}^{m_3}_{X_3} \hat{P}^{m_1}_{X_1} Z_1(\theta_1) \hat{P}^{m_4}_{X_4} Z_4(\theta_4) \hat{P}^{m_2}_{X_2} Z_2(\theta_2) |\psi, S\rangle$$

Applying the same procedure to the stabilizer $X_m Z_n Z_1$ produces:

$$[X_n Z_m](\theta_3) [X_m Z_n](\theta_1) \hat{P}^{m_3}_{X_3} \hat{P}^{m_1}_{X_1} \hat{P}^{m_4}_{X_4} Z_4(\theta_4) \hat{P}^{m_2}_{X_2} Z_2(\theta_2) |\psi, S\rangle$$

For the stabilizer $X_3 Z_1 Z_4 Z_n$, we encounter a complication: the presence of Z_1 prevents direct commutation through the measurement sequence. To resolve this, we multiply this stabilizer with the previously inserted $X_m Z_n Z_1$ to eliminate the problematic Z_1 terms:

$$X_3 Z_1 Z_4 Z_n \cdot X_m Z_n Z_1 = X_3 Z_4 X_m$$

where we used the identity relations $Z_n^2 = Z_1^2 = \mathbb{I}$. Applying the rotation lemma with this combined stabilizer gives:

$$[X_n Z_m](\theta_3) [X_m Z_n](\theta_1) [X_m](\theta_4) \hat{P}^{m_3}_{X_3} \hat{P}^{m_1}_{X_1} \hat{P}^{m_4}_{X_4} \hat{P}^{m_2}_{X_2} Z_2(\theta_2) |\psi, S\rangle$$

where the X_3 factor has been absorbed into the measurement projector $\hat{P}^{m_3}_{X_3}$ due to the idempotent property of projection operators.

Finally, processing the remaining stabilizers $X_1 Z_m Z_2 Z_3$ and $X_n Z_m Z_3$ through multiplication:

$$X_1 Z_m Z_2 Z_3 \cdot X_n Z_m Z_3 = X_1 Z_2 X_n$$

where we again used $Z_m^2 = Z_3^2 = \mathbb{I}$. This yields the final result:

$$[X_n Z_m](\theta_3) [X_m Z_n](\theta_1) [X_m](\theta_4) [X_n](\theta_2) \hat{P}^{m_3}_{X_3} \hat{P}^{m_1}_{X_1} \hat{P}^{m_4}_{X_4} \hat{P}^{m_2}_{X_2} |\psi, S\rangle$$

5.1.1 Equivalent Unitary Circuit

This systematic analysis demonstrates that the measurement-based computation is mathematically equivalent to applying the unitary transformation:

$$\mathcal{U}_{\text{MBQC}} = [X_n Z_m](\theta_3) [X_m Z_n](\theta_1) [X_m](\theta_4) [X_n](\theta_2)$$

followed by projective measurements on the auxiliary qubits. The unitary operators act exclusively on the output qubits m and n, establishing a direct correspondence between the measurement angles $\{\theta_1, \theta_2, \theta_3, \theta_4\}$ and the rotation parameters of the equivalent circuit model.

This result provides an equivalence between measurement-based and circuit-based quantum computation.

The numerical implementation is demonstrated through quantum circuits that directly apply the extracted Pauli exponentials to the output qubits. Using Qiskit's PauliEvolutionGate class, each extracted unitary operator $[P_i](\theta_j)$ can be efficiently implemented as a parametrized quantum gate, enabling direct experimental realization of the equivalent circuit model. The implementation, including graph state preparation (only one edge for a simple example) and application of extracted unitaries, is provided in Appendix A.6.

5.1.2 Generalization to Arbitrary Graph Structures

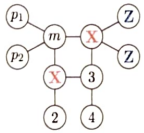
The unitary extraction protocol can be extended to all graph topologies. When the output vertices m, n are adjacent to extra vertices compared to the minimal configuration, the resulting Pauli exponentials must be modified to

take into account the extended neighborhood:

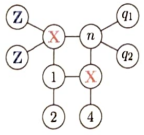
$$\mathcal{U}_{\text{general}} = \left[X_n Z_{N_n \setminus \{m\}}\right](\theta_3) \cdot \left[X_m Z_{N_m \setminus \{n\}}\right](\theta_4) \cdot \left[X_n Z_{N_n}\right](\theta_2) \cdot \left[X_m Z_{N_m}\right](\theta_1)$$

where $N_k = \{i \mid (k,i) \in E\}$ is the set of neighbours of vertex k in graph $G = (V, E)$. This generalization means that the extraction method based on stabilizers scales naturally with the complexity of the graph, one neighbor simply resulting in one additional Z operator in the extracted Pauli exponents.

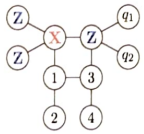
The form of the correction for each measurement follows from the focused generalized flow construction, as explained in Figure 5.2. For an auxiliary qubit, each measurement triggers some set of Pauli corrections to the other qubits as dictated by the correction sets specified by the gflow.



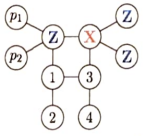
(a) Measurement of qubit 2: X corrections on qubits 1 and n, Z corrections on neighbors of n excluding m (i.e., q_1, q_2).



(b) Measurement of qubit 4: X corrections on qubits 3 and m, Z corrections on neighbors of m excluding n (i.e., p_1, p_2).



(c) Measurement of qubit 1: X correction on m, Z corrections on all neighbors of m (i.e., n, p_1, p_2).



(d) Measurement of qubit 3: X correction on n, Z corrections on all neighbors of n (i.e., m, q_1, q_2).

Figure 5.2: Pauli corrections applied after each sequential measurement, illustrating the correction pattern determined by the focused gflow.

Bad name. Verification ?? → let's discuss the appropriate title.

5.1.3 Quantum State Characterization

To provide a mathematical characterization of the computational process, I analyzed the final quantum state $|\psi_{m,n}\rangle$ of the output qubits m and n after measuring all auxiliary qubits in bases rotated by angles θ_i $(i = 1, \ldots, 4)$ in the XY plane.

The resulting state can be expressed in the computational basis as:

$$|\psi_{m,n}\rangle = C_{00}|0_m 0_n\rangle + C_{01}|0_m 1_n\rangle + C_{10}|1_m 0_n\rangle + C_{11}|1_m 1_n\rangle$$

The probability amplitudes are determined by the following analytical expressions:

$$C_{00} = 1 + \cos\theta_2 \sin\theta_3 \sin\theta_4 + \cos\theta_4 \sin\theta_1 \sin\theta_2 + \cos\theta_1 \cos\theta_3 \sin\theta_2 \sin\theta_4$$

$$C_{01} = \frac{1}{2}(\cos^2\theta_4 + \cos\theta_4 - 1) + \sin\theta_1 \sin\theta_2 + i \sin\theta_4(\cos\theta_2 \cos\theta_3 - \cos\theta_1 \sin\theta_2 \sin\theta_3)$$

$$C_{10} = \cos\theta_2 + \sin\theta_3 \sin\theta_4 + i \sin\theta_2(\cos\theta_1 \cos\theta_4 - \cos\theta_3 \sin\theta_1 \sin\theta_4)$$

$$C_{11} = -\cos\theta_2 \cos\theta_4 - i \cos\theta_1 \sin\theta_2 + \sin\theta_4(\sin\theta_1 \sin\theta_2 \sin\theta_3 - i \cos\theta_3)$$

These expressions represent unnormalized probability amplitudes for each computational basis state, with the physical quantum state obtained by normalization: $|\psi_{m,n}\rangle_{\text{norm}} = |\psi_{m,n}\rangle / \sqrt{\sum_{ij} |C_{ij}|^2}$. The coefficients C_{ij} encode complete information about the entanglement structure and quantum correlations between the output qubits as explicit functions of the four measurement angles $\{\theta_1, \theta_2, \theta_3, \theta_4\}$.

These expressions has been verified numerically, the code is shown in Appendix A.1.

5.2 Flow conditions in decoration schemes

Let's consider again the decoration in 5.1. As shown in Figure 5.3, vertex 2 participates in decorations for both edges 1-2 and 2-3. While this seems natural, it creates a conflict in the measurement correction protocol. The shared vertex creates competing correction requirements that cannot be simultaneously satisfied.

• In the Figure of the Graph defined at ...

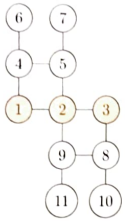
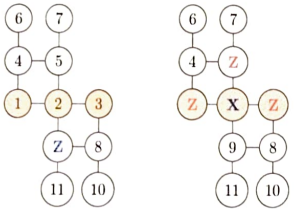


Figure 5.3: Problematic configuration: output qubits 1 2 3 (orange) with decorations on edges 1-2 and 2-3 sharing vertex 2.

Here's what goes wrong. When measuring auxiliary qubits in the rotated basis, unfavorable outcomes ($s_i = 1$) generate Z byproducts requiring correction. Suppose a measurement creates byproduct Z_9 on qubit 9. To correct this, we apply stabilizer $X_2Z_1Z_3Z_5Z_9$. This eliminates Z_9 but creates new byproducts on qubits 1, 3, and 5. The crucial problem is that byproduct Z_5 (in the upper gadget) cannot be corrected without affecting already-processed measurements, violating gflow causality.



(a) Initial byproduct Z_9 requires correction

(b) Correction creates uncorrectable cascade

Figure 5.4: Correction failure: stabilizer correction creates byproduct Z_5 that violates measurement causality.

The problem runs deeper than correction cascades. Different measurement outcomes produce inequivalent post-measurement states, making deterministic computation impossible. For instance, when projecting qubits 9 and 5 onto different outcome combinations, we get:

For outcomes $|\theta^+\rangle_9$ and $|\theta^-\rangle_5$:

$$\begin{aligned}\langle\theta^+|_9\langle\theta^-|_5|G\rangle &= \langle\theta^+|_9\langle\theta^+|_5 Z_5|G\rangle \\ &= \langle\theta^+|_9\langle\theta^+|_5 X_2Z_1Z_3Z_9|G\rangle \\ &= \langle\theta^-|_9\langle\theta^+|_5 X_2Z_1Z_3|G\rangle \end{aligned} \tag{5.1}$$

For outcomes $|\theta^-\rangle_9$ and $|\theta^+\rangle_5$:

$$\begin{aligned}\langle\theta^-|_9\langle\theta^+|_5|G\rangle &= \langle\theta^+|_9\langle\theta^+|_5 Z_9|G\rangle \\ &= \langle\theta^+|_9\langle\theta^+|_5 X_2Z_1Z_3Z_5Z_9|G\rangle \\ &= \langle\theta^+|_9\langle\theta^-|_5 X_2Z_1Z_3|G\rangle \end{aligned} \tag{5.2}$$

For outcomes $|\theta^-\rangle_9$ and $|\theta^-\rangle_5$:

$$\begin{aligned}\langle\theta^-|_9\langle\theta^-|_5|G\rangle &= \langle\theta^+|_9\langle\theta^+|_5 Z_9Z_5|G\rangle \\ &= \langle\theta^+|_9\langle\theta^+|_5 X_2Z_1Z_3|G\rangle \qquad (5.3) \\ &= \langle\theta^+|_9\langle\theta^+|_5|G\rangle \qquad (5.4) \end{aligned}$$

The postmeasurement states in equations (5.1), (5.2), and (5.4) reveal inequivalence. While individual by-products on qubits 1, 2, and 3 can be handled by classical postprocessing, the state inequivalence forces the protocol to use postselection, accepting only specific outcome combinations and discarding others.

To verify this limitation, we used the `graphix` library, that constructs measurement patterns from graph specifications using the function `generate_from_graph`. Testing configurations with shared vertices confirms that such patterns fail the gflow condition. The code is provided in Appendix A.2.

Let us now consider the node-wise decoration scheme, illustrated in Figure 5.5. The first digit of each white node represents its layer (e.g., nodes 10-12 belong to layer 1, nodes 20-22 to layer 2) and the measurement order is given by the arrow. The simple linear graph with three qubits is the same as 5.3.

but in this case the vertex in the middle (1 in this case) no longer causes gflow problems. In fact, the eventual presence of a byproduct Z on qubit 11 can be easily corrected by using the stabilizer of qubit 1. In general, each Z byproduct is corrected for with the stabilizer of the neighboring qubit in the subsequent measurement layer.

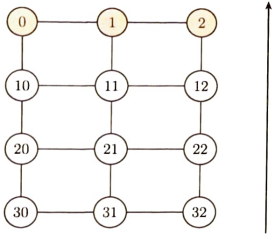


Figure 5.5: Node-wise decoration with 3 layers. Orange nodes are physical output qubits, white nodes are decoration qubits measured from bottom to top.

In the appendix Appendix A.7 we provide a code that verifies determinism by running multiple trials. The implementation confirms that states remain equivalent across all runs, validating the gflow properties.

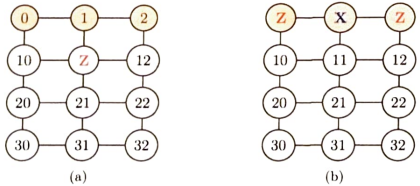


Figure 5.6: Byproduct propagation in node-wise decoration

To address the gflow violation in edge-wise decoration. I analyzed several potential solutions, each with distinct limitations and trade-offs.

5.3 Pattern Decomposition

again this is not the best...

The first approach involves partitioning the decorated graph into subgraphs that can be processed sequentially, thereby avoiding the shared-vertex conflicts that violate gflow. Each subgraph admits a local gflow and can be measured independently without interference.

Given a graph $G = (V, E)$ with edge decorations, we seek a decomposition into subgraphs $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$ such that:

- $E_A \cup E_B = E$ (all edges are covered)
- No edge decoration in G_A shares auxiliary qubits with decorations in G_B
- Each subgraph admits a valid gflow: $(g_A, \prec_A)$ for G_A and $(g_B, \prec_B)$ for G_B

The computation proceeds in two sequential stages. In the first stage, we apply measurement pattern M_A to the auxiliary qubits in G_A, obtaining intermediate state $|\psi_1\rangle = M_A|\psi_{\text{in}}\rangle$ on the output qubits. In the second stage, we apply pattern M_B to the remaining auxiliary qubits in G_B, with $|\psi_1\rangle$ serving as the input state. The final computational result is $|\psi_{\text{out}}\rangle = M_B|\psi_1\rangle$.

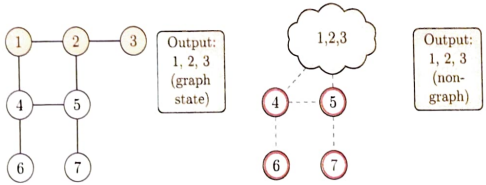
This decomposition problem is closely related to edge coloring: we assign colors to edges such that no two edges sharing a vertex receive the same color. Each color class defines a set of edges whose decorations can be applied simultaneously without shared-vertex conflicts. For a graph requiring k colors, the measurement patterns are applied in k sequential stages corresponding to the k color classes. The chromatic index $\chi'(G)$ thus determines the minimum number of sequential stages required.

5.3.1 Limitations of Pattern Decomposition

The sequential decoration approach faces a fundamental limitation: the output qubits from the first stage no longer exist in a graph state configuration after measurements are performed. Instead, they emerge in a

measurement-dependent entangled state that is incompatible with the graph state requirements of the second decoration stage.

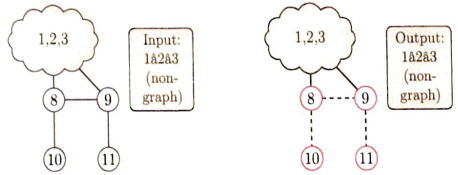
Since edge decorations and the gflow formalism are specifically designed for graph state inputs, the second measurement pattern M_B cannot properly process the non-graph intermediate state $|\psi_1\rangle$ it receives from the first stage. This incompatibility stems from the loss of the stabilizer structure that defines graph states.



(a) Before measurement of the first decoration qubits

(b) After measurement of first decoration qubits

Figure 5.7: First stage of pattern decomposition showing transition from graph state to non-graph state after partial measurements.



(a) Before measurement of second decoration qubits

(b) After measurement of second decoration qubits

Figure 5.8: Second stage of pattern decomposition attempting to process non-graph state input, illustrating the fundamental incompatibility.

The transition from graph state to arbitrary entangled state is illustrated in Figures 5.7 and 5.8. In Figure 5.7, we observe that after measuring the first

decoration qubits (4, 5, 6, 7), the output qubits (1, 2, 3) no longer maintain their graph state structure. The subsequent stage in Figure 5.8 shows that when we attempt to apply the second decoration pattern to qubits 8, 9, 10, and 11, the input state is no longer a graph state but an arbitrary entangled state represented by the cloud-like blob.

The loss of graph state structure implies that the stabilizer formalism no longer applies, making byproduct corrections impossible to implement within the standard MBQC framework. For example, if we need to correct a Z byproduct on qubit 8 (as shown in the left panel of Figure 5.8), we cannot utilize the stabilizer operator that would normally be associated with the connected output qubits, as these operators are no longer well-defined after the first stage measurements have destroyed the graph state structure.

5.4 Sequential edge decoration with dynamic relabeling

To address the limitation presented in the previous section, I present an alternative strategy through dynamic qubit relabeling. This approach maintains the stabilizer structure throughout the computation, ensuring efficient deterministic corrections.

We begin with a register of qubits initialized in a stabilizer state. We decorate edges sequentially, one at a time. At each step, we select any two qubits from this register and designate them as input qubits for the current decoration operation. We apply the local decoration gadget to these selected qubits and introduce the necessary ancilla qubits. We then perform measurements on both the designated input qubits and the ancilla qubits. Deterministic corrections follow immediately using the available stabilizer generators. The measured qubits are then discarded from the system.

The decoration produces two output qubits that are integrated back into our qubit register, replacing the previously selected inputs. These new output qubits can be dynamically reassigned in subsequent decoration steps. Both output qubits, neither of them, or only one may serve as inputs for the next decoration operation. This dynamic relabeling maintains a manageable register size while increasing entanglement complexity. The flexible assignment enables strategic construction of the desired entanglement structure while preserving efficient stabilizer-based computation.

Step-by-step procedure:

1. **Setup and input selection:** At step k, we begin with a stabilizer quantum state $|\psi_{k-1}\rangle$ defined on qubit register V_{k-1}. We select any two qubits $a_k, b_k \in V_{k-1}$ and designate them as input qubits for the current decoration, regardless of their existing connectivity.

2. **Decoration and measurement:** We introduce four ancillary qubits $A_k = \{a_1^{(k)}, a_2^{(k)}, a_3^{(k)}, a_4^{(k)}\}$, each initialized in $|+\rangle$, creating the state $|\psi_{k-1}\rangle \otimes |+\rangle^{\otimes 4}$. A local Clifford unitary U_k entangles the input qubits with all ancillae through CZ and Hadamard gates, maintaining the stabilizer structure. We then perform XY-plane measurements on a_k, b_k, $a_1^{(k)}$, and $a_2^{(k)}$, recording outcomes as $m_k = (m_{a_k}, m_{b_k}, m_{a_1^{(k)}}, m_{a_2^{(k)}})$.

3. **Output correction and relabeling:** The unmeasured ancillae $a_3^{(k)}$ and $a_4^{(k)}$ become our output qubits. We apply deterministic Pauli corrections based on m_k and available stabilizer generators, then relabel them as $o_k^{(1)} := a_3^{(k)}$ and $o_k^{(2)} := a_4^{(k)}$. The register updates as $V_k = (V_{k-1} \setminus \{a_k, b_k\}) \cup \{o_k^{(1)}, o_k^{(2)}\}$, with these outputs available for dynamic selection in subsequent decoration steps.

The overall state after step k can be compactly expressed as:

$$|\psi_k\rangle = C_k \left(|\psi_{k-1}\rangle \otimes |+\rangle^{\otimes 4} \right),$$

where C_k encapsulates the full sequence: local Clifford application, measurement and deterministic byproduct correction.

The crucial advantage of this sequential approach is that byproduct corrections are determined entirely by the decoration gadget itself. When we measure qubits a_k, b_k and ancillae $a_1^{(k)}, a_2^{(k)}$, the required stabilizer generators for corrections come directly from the output qubits $a_3^{(k)}, a_4^{(k)}$ within the same gadget. We do not need stabilizers from the rest of the graph state. This ensures deterministic corrections at every step, regardless of how complex the overall state becomes.

Basic? Simple? First?

5.4.1 ~~Worked~~ Example: Linear Graph State

I illustrate the dynamic qubit relabeling procedure on a simple three-qubit linear graph state. We begin with $V_0 = \{1, 2, 3\}$ and initial state

$$|\psi_0\rangle = CZ_{12} CZ_{23} |+\rangle^{\otimes 3},$$

representing edges 1–2 and 2–3.

Decoration step 1: We select qubits 1 and 2 as input qubits for our first decoration operation.

1. Add four ancillae $A_1 = \{a_1^{(1)}, a_1^{(2)}, a_1^{(3)}, a_1^{(4)}\}$, all initialized in $|+\rangle$.

2. Apply local Clifford unitary U_1 to inputs 1, 2 and all ancillae according to the decoration pattern.

3. Perform measurements on qubits 1, 2, $a_1^{(1)}$, and $a_1^{(2)}$ in appropriate bases.

4. Apply deterministic Pauli corrections based on measurement outcomes, as shown in Figure 5.9.

5. Relabel the corrected qubits as $o1 := a_1^{(3)}$ and $o2 := a_1^{(4)}$.

6. Update the register: $V_1 = \{o1, o2, 3\}$.

Decoration step 2 - Dynamic reassignment: I now demonstrate the flexibility of the approach by selecting $o2$ (an output from step 1) together with qubit 3 as our new input pair.

1. Add new ancillae $A_2 = \{a_2^{(1)}, a_2^{(2)}, a_2^{(3)}, a_2^{(4)}\}$.

2. Apply U_2 to inputs o_2, 3, and ancillae A_2.

3. Measure $o2$, 3, $a_2^{(1)}$, and $a_2^{(2)}$.

4. Apply corrections and relabel $a_2^{(3)} \to out_3$ and $a_2^{(4)} \to out_4$.

5. Final register: $V_2 = \{o1, o3, o4\}$.

This example demonstrates how output qubits from previous decorations can be dynamically reassigned as inputs for subsequent operations, enabling flexible construction of complex entanglement structures while maintaining deterministic control throughout. This approach offers several practical advantages:

- **Modularity:** Each decoration acts locally and independently. Decorations can be applied in any order to any pair of qubits, enabling step-by-step construction of complex entanglement patterns.

- **Efficient tracking:** Since each step involves only a few qubits, measurement results and corrections are easy to track. The stabilizer formalism enables straightforward state updates.

- **Circuit-like structure:** This mirrors gate-based quantum circuits, where operations act on few qubits at a time while preserving logical structure.

- **Generality:** The method applies to any local gadget operating on qubit pairs with ancillae.

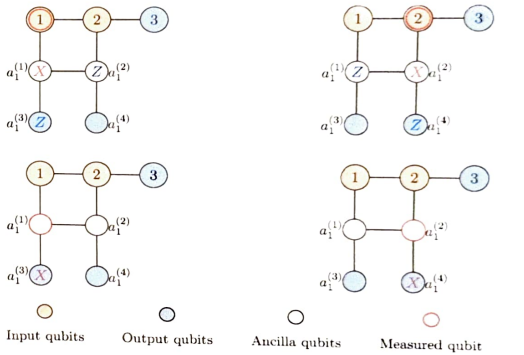


Figure 5.9: Sequential measurement and byproduct correction pattern for decoration step 1.

The qubit relabeling approach can be visualized as a sequential cascade of decoration gadgets. Figure 5.10 illustrates this process.

5.5 Numerical simulations

I performed numerical simulations to validate the theoretical framework developed in the previous sections. The simulations were conducted using two complementary approaches.

First, I was implemented circuit-based simulations using the unitaries derived in Section 5.1.2. These simulations allowed me to directly construct and manipulate the quantum states using the gate sequences obtained from the stabilizer analysis.

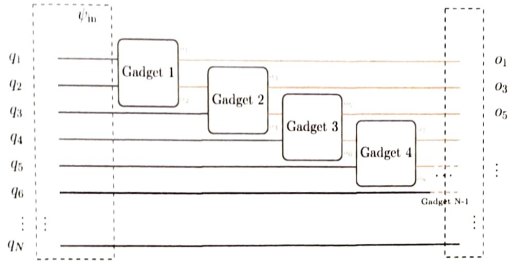


Figure 5.10: Diagrammatic representation of sequential gadget decorations with dynamic qubit role updates for N qubits. Starting with N input qubits $q_1, q_2, \ldots, q_N$, each gadget connects one output from the previous decoration with the next unused original input qubit.

Subsequently, I carried out measurement-based simulations. This approach enabled me to explore the computational aspects of the decorated graph states and verify the consistency between the circuit-based and measurement-based representations. Moreover, the results obtained from the measurement-based computation allowed me to verify the correctness of the proposed theoretical solution to guarantee deterministic measurements.

5.5.1 Perturbed Toric Code ground state

For the numerical simulations, I consider the toric code Hamiltonian with an added perturbation term:

$$H = H_0 + H_{\text{pert}} \tag{5.5}$$

where H_0 is the unperturbed toric code Hamiltonian and H_{pert} represents a longitudinal magnetic field perturbation:

$$H_{\text{pert}} = -\lambda \sum_{i=0}^{7} Z_i \tag{5.6}$$

with λ controlling the strength of the perturbation. For simplicity, consider the case in which λ is equal for all qubits, so that the perturbation is homogeneous. I start by computing the ground state of the unperturbed Hamiltonian H_0, because I want this ground state to be in a graph-state form in order to decorate it within the MB-VQE framework.

Consider the toric code on a 2×2 lattice with periodic boundary conditions, defined on $N_x \times N_y = 2\times2 = 4$ plaquettes and 8 qubits total. The unperturbed Hamiltonian is given by:

$$H_0 = -\sum_s A_s - \sum_p B_p \tag{5.7}$$

where $A_s = \prod_{i\in s} X_i$ are the star operators acting on qubits around each vertex s, and $B_p = \prod_{i\in p} Z_i$ are the plaquette operators acting on qubits around each face p.

The stabilizers of the unperturbed toric code Hamiltonian can be written down analytically; however, I employed the algorithm presented in [36] to automatically recover the stabilizer generators and their graph state representation. This step served as both a consistency check and as a convenient way to interface with graph-based simulation tools.

The algorithm takes as input a text file that describes the Hamiltonian. In this case, the input file has the following structure:

```
8 8
-1.0 XXXX 0 1 2 3
-1.0 XXXX 4 5 6 7
-1.0 ZZZZ 0 1 4 5
-1.0 ZZZZ 1 2 5 6
-1.0 ZZZZ 2 3 6 7
-1.0 ZZZZ 3 0 7 4
-1.0 XXXX 0 2 4 6
-1.0 ZZZZ 1 3 5 7
```

The text file starts with two integers: n and k, representing the number of qubits and the number of terms in the Hamiltonian. Each of the following

k lines has the form:

```
[coefficient] [Pauli string] [qubit indices]
```

For example, the line `-1.0 XXXX 0 1 2 3` encodes the term $-X_0X_1X_2X_3$.

The stabilizer generators of the ground state are given by:

$$S = \langle S_1, S_2, S_3, S_4, S_5, S_6 \rangle, \quad S_i |\psi\rangle = |\psi\rangle \ \forall i, \quad \langle\psi| H_0 |\psi\rangle = -8$$

with generators:

$$\begin{aligned} S_1 &= ZIIZZIIZ & S_2 &= XIXIIXIX \\ S_3 &= IZIZIZIZ & S_4 &= IXIXIXIX \\ S_5 &= IIZZIIZZ & S_6 &= IIIIXXXX \end{aligned}$$

The toric code has a four-dimensional ground state space due to its topological degeneracy. Among these degenerate states, the logical state $|0,0\rangle_L$ serves as the most natural choice for our analysis, as it provides the best approximation to the ground state of the perturbed Hamiltonian $H_0 + H_{\text{pert}}$ for small positive values of the perturbation parameter λ. The graph state representation of $|0,0\rangle_L$ can be calculated efficiently using classical algorithms and exhibits a complex connectivity pattern that reflects the underlying topological correlations.

5.5.2 Circuit-based simulation

A variational quantum eigensolver was implemented using the unitary representations derived in Section 5.1.2. The implementation centers on the `DecorAnsatz` class, which builds parameterized quantum circuits in Qiskit (see Appendix ??).

For each decorated edge (i, j), the class adds four variational parameters

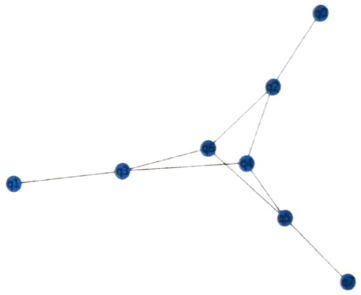


Figure 5.11: Graph state corresponding to the logical state $|0,0\rangle_L$ of the 2×2 toric code.

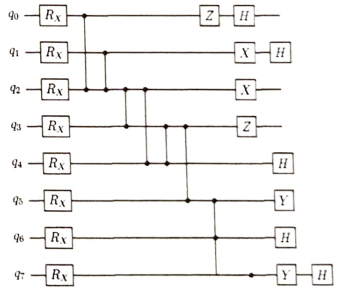


Figure 5.12: Circuit preparing the graph state from Fig. 5.11.

$\{\theta_1, \theta_2, \theta_3, \theta_4\}$. These appear in the generalized unitary:

$$\mathcal{U}(\theta) = \prod_{\text{edges}(i,j)} \exp\left(-i\frac{\theta_4}{2} X_j Z_{N_j \setminus \{i\}}\right) \times \exp\left(-i\frac{\theta_3}{2} X_i Z_{N_i \setminus \{j\}}\right) \times \exp\left(-i\frac{\theta_2}{2} X_j Z_{N_j}\right) \times \exp\left(-i\frac{\theta_1}{2} X_i Z_{N_i}\right) \tag{5.8}$$

where N_k denotes the neighbors of vertex k. Each exponential is implemented as a Qiskit `PauliEvolutionGate`.

The ansatz states are:

$$|\psi(\theta)\rangle = \mathcal{U}(\theta)|\psi_{\text{init}}\rangle \tag{5.9}$$

where $|\psi_{\text{init}}\rangle$ is either the graph state $|G\rangle$ or the computational basis state $|1\rangle^{\otimes n}$. The graph state ansatz encodes the stabilizer structure; the computational basis provides a classical baseline.

For each λ, the VQE minimizes:

$$E(\theta) = \langle\psi_{\text{init}}|\mathcal{U}(\theta)^\dagger H(\lambda)\mathcal{U}(\theta)|\psi_{\text{init}}\rangle \tag{5.10}$$

The optimal parameters are:

$$\theta^\star = \arg\min_\theta E(\theta) \tag{5.11}$$

Optimization uses COBYLA with tolerance 10^{-6}. Parameters initialize uniformly in $[0, 10^{-3}\pi/2]$.

Edge selection requires non-overlapping edges only. This constraint addresses gflow requirements but severely limits ansatz expressivity. Only edge subsets can be decorated simultaneously, preventing full ground state capture in dense graphs or strongly interacting regimes.

Simulations used Qiskit's statevector simulator for 8-qubit systems. Optimization converged within 200-500 iterations with $\mathcal{O}(n^3)$ scaling. Statistical analysis over 10 runs confirmed reliability.

Results show crossover at $\lambda^\star \approx 1.0 \pm 0.1$. For $\lambda < \lambda^\star$, graph state ansatz achieves errors below 10^{-4} while computational basis stagnates near 10^{-1}. For $\lambda > \lambda^\star$, computational basis becomes superior with errors below 10^{-3}.

Below $\lambda^\star$, ground states exhibit high entanglement. Above $\lambda^\star$, local Z-interactions dominate. The decoration unitaries capture both regimes despite the edge overlap limitation.

The non-overlapping constraint represents the main limitation of this approach.

...see if the graph works....

5.5.3 Measurement-Based simulation

For the measurement-based simulation, the main algorithm consists of three components.

The `gadget_decorate` function creates a measurement pattern that transforms edge (u, v) into a parametrized quantum circuit. The function takes input qubits `u` and `v`, preserved qubits `pres`, and four measurement angles `t = [t1,u, t1,v, t1,a, t1,b]`. It creates four auxiliary qubits `a, b, o1, o2` and prepares them in state $|+\rangle$. The function builds the specific entanglement topology of the decoration with edges `(u,a), (v,b), (a,o1), (a,b), (b,o2)`. Each qubit is then measured in the XY-plane with its corresponding angle parameter. The measurements of `u` and `v` trigger conditional Pauli corrections on multiple qubits to ensure deterministic computation regardless of measurement outcomes. These corrections are applied using the `t` and `s` domain of the *Graphix* library. The auxiliary qubits `a` and `b` are also measured, with their outcomes determining corrections on the final output qubits `o1` and `o2`. The pattern standardization reorders all commands into the canonical form, and the function returns the complete pattern along with the two output qubits that replace the original edge endpoints. The `build_state` function starts by creating the graph state $|G\rangle$ for the input graph. This gives the initial quantum state `st`. The function loops through each edge (u, v). For edge i, it calls `gadget_decorate` with qubits `nds[u]` and `nds[v]`. This returns a pattern `pg` and output qubits `o1, o2`. The function runs `sg.run(input_state=st)`. This applies pattern `pg` to the current state `st`. The pattern reads qubits `nds[u]` and `nds[v]` from state `st`, applies the decoration, and outputs a new state. After running the pattern, `nds[u] =`

`o1` and `nds[v] = o2`. The mapping changes because the decoration replaced the original qubits with new ones. The next edge that uses node `u` will use qubit `o1`. The state `st` gets updated to the new state after each decoration. By the end, `st` contains the quantum state with all edge decorations applied. The `optimize_for_lambda` function implements the variational optimization for a given parameter `lam`. The Hamiltonian is constructed as `H = H0_mat + lam*Zsum_mat`, and the exact ground state energy `e0` is computed using `np.linalg.eigvalsh(H)[0]`. The optimization uses a parameter vector `x` of size `4*E + N`, where the first `4*E` elements contain four measurement angles for each of the `E` graph edges, and the remaining `N` elements are additional parameters. The objective function `obj(x)` unpacks the parameter vector into edge angles `G` and other parameters `O`, constructs the quantum state using `build_state(g,G,O)`, and returns the squared difference between the variational energy and the exact ground state energy. The COBYLA optimizer minimizes this objective function starting from random initial parameters sampled uniformly from `[0, 2π]`. The optimization uses 3000 maximum iterations, initial trust region radius 0.5, and convergence tolerance `1e-6`. After optimization, the function builds the final state with the optimized parameters, computes the variational energy, and returns the exact energy, variational energy, and relative error.

The cost function minimizes the squared difference between the variational energy and the exact ground state:

$$\mathcal{L}(\mathbf{x}) = \left(\langle\psi(\mathbf{x})|H(\lambda)|\psi(\mathbf{x})\rangle - E_0(\lambda)\right)^2 \tag{5.12}$$

where the parameter vector $\mathbf{x} \in [0, 2\pi]^{4E}$ contains four measurement angles for each of the E edges in the graph.

Optimization employs the Constrained Optimization BY Linear Approximations (COBYLA) algorithm.

We set the initial trust region radius $\rho_{\text{beg}} = 0.5$, convergence tolerance 10^{-6}, and maximum 3000 iterations. Implementation details are in Appendix A.3.

In figure 5.13 the blue curve represents the relative error obtained by the measurement-based variational quantum eigensolver, calculated as:

$$\text{Relative Error}_{\text{MB-VQE}} = \frac{|E_{\text{MB-VQE}}(\lambda) - E_{\text{exact}}(\lambda)|}{|E_{\text{exact}}(\lambda)|} \tag{5.13}$$

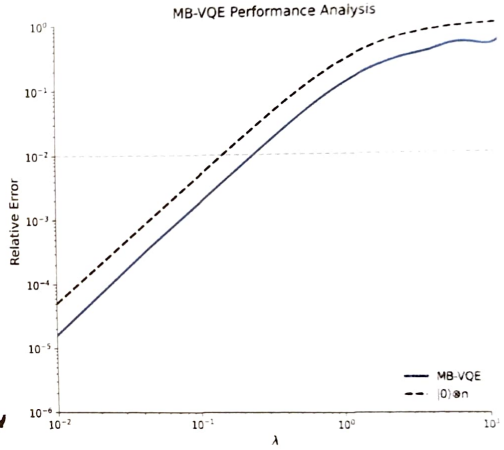


Figure 5.13: MB-VQE with dynamic relabeling method vs perturbation strengths. Results are computed over 50 logarithmically spaced values of λ from 10^{-2} to 10.

where $E_{\text{MB-VQE}}(\lambda)$ is the energy obtained through our variational optimization and $E_{\text{exact}}(\lambda)$ is the true ground state energy of the perturbed Hamiltonian computed by exact diagonalization.

The black dashed line shows the reference error computed using the unperturbed logical state $|0\rangle^{\otimes n}$ as a trial wavefunction:

$$\text{Relative Error}_{\text{ref}} = \frac{|\langle 0^{\otimes n}|H(\lambda)|0^{\otimes n}\rangle - E_{\text{exact}}(\lambda)|}{|E_{\text{exact}}(\lambda)|} \tag{5.14}$$

This reference represents the best approximation achievable without variational optimization, using only the ground state of the unperturbed toric code.

The consistent position of the blue curve below the black curve across all λ values confirms that the MB-VQE ansatz provides a superior approximation to the true ground state compared to the simple logical state. The MB-VQE approach maintains relative errors below 10^{-1} for perturbation strengths up to $\lambda \approx 0.9$, demonstrating good accuracy across a substantial range of the perturbation parameter.

However, for strong perturbations ($\lambda > 1$), MB-VQE performance degrades significantly. The decorated ansatz lacks sufficient expressivity to capture ground states where perturbation dominates the Hamiltonian. In this regime, the ground state deviates substantially from the original toric code structure, and edge decoration fails to provide adequate variational freedom for accurate representation. While preserving deterministic gflow, this technique cannot represent the transition toward separable states.

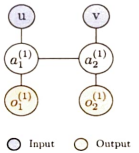
5.5.4 Adaptive Ansatz

A possible solution to this problem is proposed and consists of the use of an adaptive ansatz strategy that changes as a function of the strength of the perturbation λ. The idea is simple: we use two different ansatz types, each designed for a different physical regime. A threshold value λ_{th} separates the two approaches. The choice of the threshold is discussed later.

- **Entanglement-based Ansatz** (used for $\lambda < \lambda_{\text{thresh}}$): suitable when the ground state is close to a highly entangled stabilizer state. In this regime, I initialize the system in the graph state corresponding to H_0 and apply local MBQC gadgets on each edge, following the approach of sequential edge decoration with dynamic relabeling described in Section 5.4.
- **Separable Ansatz with Local Corrections** (used for $\lambda \geq \lambda_{\text{thresh}}$): works better when the ground state approaches a simple separable state. I start with isolated qubits (single vertices with no initial entanglement) and apply MBQC decorations to each pair of qubits.

The procedure for the regime $\lambda \geq \lambda_{\text{thresh}}$ consists of sequential edge decorations applied to a linear chain.

Each decoration takes two input qubits and produces two output qubits through the following process:
As shown in 5.14, I first create the measurement pattern using the graphix function *generate_from_graph*. This function takes an open graph G = (nodes, edges, input, outputs) specified by the network structure and supports only XY-plane measurements. The function searches for the gflow in the open graph using *graphix.gflow.find_gflow* and constructs the measurement pattern according to theorem 1 of [20]. Thanks to this function, we ensure the determinism of the computation.




```
def decorate(u, v, thetas):
ℶ_u, ℶ_v, ℶ_a1, ℶ_a2 = thetas
G = nx.Graph()
G.add_nodes_from([u, v])
mid1, mid2 = max(u, v) + 1, max(u, v) + 2
out1, out2 = max(u, v) + 3, max(u, v) + 4
G.add_nodes_from([mid1, mid2, out1, out2])
G.add_edges_from([(u, mid1), (v, mid2),
(mid1, mid2), (mid1, out1), (mid2, out2)])
angles = {u: ℶ_u, v: ℶ_v, mid1: ℶ_a1, mid2:
ℶ_a2}
pattern = generate_from_graph(G, angles, ...)
```

Figure 5.14: Single edge decoration structure with input and output nodes highlighted

I specify that the input qubits are u and v, and I measure u, v, $a_1^{(1)}$, and $a_2^{(1)}$. I then simulate this measurement pattern using *PatternSimulator* and store the resulting quantum state on the output qubits $o_1^{(1)}$ and $o_2^{(1)}$.

For subsequent decorations, I add each new qubit to the existing quantum state in the $|+\rangle$ state and connect it to $o_1^{(i-1)}$ from the previous step.

This dynamic decoration process continues until all qubits in the target Hamiltonian are incorporated (for example, 8 qubits in the case of the Toric code. All qubits are measured in the XY-plane, byproduct corrections are properly applied to ensure determinism, and each edge contributes 4 varia-

```
pattern1, out1, out2 = decorate(qubit_ids[0], qubit_ids[1],
thetas[0])
sim1 = PatternSimulator(pattern1, backend="statevector")
sim1.run()
backend = sim1.backend
out_nodes[1] = (out1, out2)
```

Figure 5.15: Initial decoration: create and simulate the first measurement pattern

```
backend.add_nodes([qubit_ids[i+1]], data=BasicStates.PLUS)
pattern, out_a, out_b = decorate(out_nodes[i][0], qubit_-
ids[i+1], thetas[i])
sim = PatternSimulator(pattern, backend=backend)
sim.run(input_state=None)
```

Figure 5.16: Sequential decoration: add new qubit and connect to previous output

tional angles to the optimization. In figure 5.17 there is an example with 4 qubits. The two main functions are shown in Appendix A.4.

5.5.5 Step 2: Single-Qubit Rotations

The second critical component of the ansatz involves the application of parameterized single-qubit rotations to the output qubits. On each of the N output qubits, I apply a complete set of Euler angle rotations:

$$R(\theta_x^j, \theta_y^j, \theta_z^j) = R_z(\theta_z^j) \cdot R_y(\theta_y^j) \cdot R_x(\theta_x^j) \tag{5.15}$$

The complete ansatz comprises $4N + 3N$ variational parameters: $4(N-1)$ from the N edge decorations and $3N$ from the single-qubit rotations. For the rotation parameters, I employ physics-informed initialization with $\theta_x^j = 0$, $\theta_y^j = \pi$, and $\theta_z^j = 0$, which biases the state toward $|1\rangle^{\otimes N}$âthe true ground state in the $\lambda \to \infty$ limit.

For optimization, I employ the L-BFGS-B algorithm with maximum iterations of 8000, function tolerance of 10^{-14}, and gradient tolerance of 10^{-10}. I choose L-BFGS-B.

I use 50 logarithmically spaced λ values from 10^{-2} to 10^{1}. For each λ

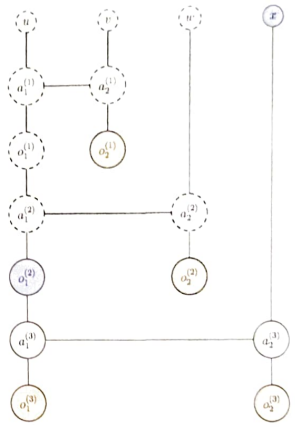


Figure 5.17: Adaptive ansatz example with 4 qubits

value, I perform 10 independent computational runs to ensure statistical reliability. Each run involves 10 optimization attempts with different random initializations of decoration angles uniformly sampled from $[0, 2\pi)$. This uniform initialization provides an unbiased exploration of the parameter space while avoiding any preferential bias toward specific angle configurations. I select the best result among the 10 attempts based on the lowest cost function value.

For each λ value, I compute the mean relative error and standard deviation across the 10 runs. The plotted results show mean values with error bars representing one standard deviation. Statistical analysis reveals mean relative errors of $\langle \epsilon \rangle \approx 2.8 \times 10^{-3}$ with standard deviations $\sigma \approx 1.3 \times 10^{-3}$ in the well-optimized regimes.

From Figure 5.18, the red dash-dot line shows the reference error computed using the separable state $|1\rangle^{\otimes n}$ as a trial wavefunction:

$$\text{Relative Error}_{|1\rangle^{\otimes n}} = \frac{|\langle 1^{\otimes n}|H(\lambda)|1^{\otimes n}\rangle - E_{\text{exact}}(\lambda)|}{|E_{\text{exact}}(\lambda)|} \tag{5.16}$$

The adaptive ansatz outperforms both references across all regimes. For large $\lambda > 0.9$, MB-VQE achieves remarkable accuracy with errors below 10^{-3}. The method successfully bridges weak and strong perturbation regimes.

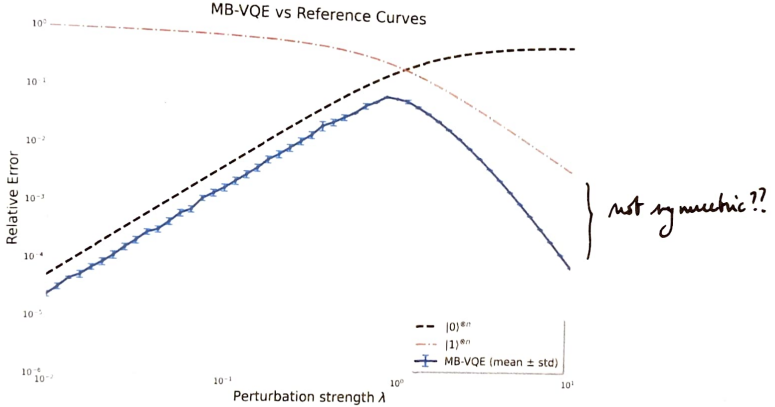


Figure 5.18: MB-VQE with adaptive ansatz method across perturbation strengths. Results are computed over 10 runs of 50 logarithmically spaced values of λ from 10^{-2} to 10.

→ Need to discuss how qubits are selected for the "decoration" → following the levels of the initial ground state... (unperturbed).
→ do simulation where we add "bad" spurious decorations → what happens??

5.6 Transverse field cluster Hamiltonian

Consider now the one-dimensional transverse field *cluster* Hamiltonian over a ring of N qubits:

$$\hat{H}(\lambda) = -\sum_{i=1}^{N} Z_{i-1} X_i Z_{i+1} - \lambda \sum_{i=1}^{N} X_i \tag{5.17}$$

The system has periodic boundary conditions forming a ring where the qubit N is adjacent to the qubit 1.

When $\lambda = 0$, the Hamiltonian reduces to the pure one dimensional linear cluster Hamiltonian, whose ground state is the one-dimensional cluster state, where each qubit is stabilized by the operator $Z_{i-1} X_i Z_{i+1}$.

When λ is large, the transverse field term dominates and the ground state approaches the product state where all qubits are aligned in the $|+\rangle$ state, that is a completely unentangled product state: $|\psi\rangle = |+\rangle^{\otimes N}$.

The parameter λ thus controls a quantum phase transition between two phases.

This system has been studied [15] and emerges that there are two different phases in the thermodynamic limit: one where the ground state of the Hamiltonian can be used as an universal resource state for MBQC, so it has *computational power*, and the other in which the ground state can't be used for a measurement-based computation. There is a value of lambda at which this transition occurs. To assess computational power, we consider the *string order parameter* [15] defined, for 1D systems, by:

$$\sigma = Z_0 \otimes X_1 \otimes I_2 \otimes X_3 \otimes \cdots \otimes I_{k-2} \otimes X_{k-1} \otimes Z_k, \tag{4.8}$$

where k is half the size of the ring, in our case k = 3. This parameter changes from non-zero to zero when we change λ from 0 to large values. The larger the chain length N, the sharper the drop. In the thermodynamic limit, the change-over becomes a phase transition.

Inserisci grafico in funzione di lambda...

5.6.1 Determining the ground state of the Transverse Field Cluster Hamiltonian

I start by computing the ground state of the unperturbed cluster state Hamiltonian, because I want this ground state to be in a graph-state form in order to decorate it.

The stabilizer generators of the ground state are given by:

$$S = \langle S_1, S_2, S_3, S_4, S_5, S_6 \rangle, \quad S_i |\psi\rangle = |\psi\rangle \ \forall i, \quad \langle\psi| H |\psi\rangle = -6$$

with generators:

$$S_1 = XZIIIZ \qquad S_2 = ZXZIII$$
$$S_3 = IZXZII \qquad S_4 = IIZXZI$$
$$S_5 = IIIZXZ \qquad S_6 = ZIIIZX$$

The resulting graph is a two-dimensional topology that reflects the entanglement structure of the state and the periodic boundary conditions 5.19.

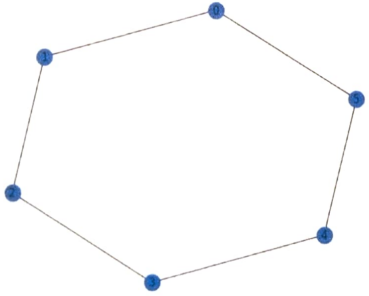


Figure 5.19: Graph state corresponding to the ground state of the 6-qubit periodic cluster Hamiltonian.

Appendix A

A.1 Numerical Verification of the Output State

The following minimal Python snippet numerically confirms the separable limit $|+\rangle|+\rangle$ when all measurement angles $\theta_i = \pi/2$:

```python
import numpy as np

def get_mbqc_state(thetas=[0,0,0,0]):
    plus = np.array([1,1]) / np.sqrt(2)
    psi0 = np.kron(plus, plus)
    t1, t2, t3, t4 = thetas
    c = [1 + np.cos(t4)*np.sin(t1)*np.sin(t2)
          + np.cos(t1)*np.cos(t3)*np.sin(t2)*np.sin(t4)
          + np.cos(t2)*np.sin(t3)*np.sin(t4),
         np.cos(t4/2)**2 + 0.5*(np.cos(t4)-1)
          + np.sin(t1)*np.sin(t2)
          + 1j*np.sin(t4)*(np.cos(t2)*np.cos(t3)
                           - np.cos(t1)*np.sin(t2)*np.sin(t3)),
         np.cos(t2) + np.sin(t3)*np.sin(t4)
          + 1j*np.sin(t2)*(np.cos(t1)*np.cos(t4)
                           - np.cos(t3)*np.sin(t1)*np.sin(t4)),
         -np.cos(t2)*np.cos(t4)
          - 1j*np.cos(t1)*np.sin(t2)
          + np.sin(t4)*(-1j*np.cos(t3)
                        + np.sin(t1)*np.sin(t2)*np.sin(t3))]
    K = np.diag(c)
    return (K @ psi0) / np.linalg.norm(K @ psi0)

if __name__ == "__main__":
    print(get_mbqc_state([np.pi/2]*4))
```