Categorical quantum theory and percolation methods for measurement-based quantum computing

Thesis submitted to the Mathematical Institute

at the University of Oxford in partial fulfilment for the degree of

Master of Science in Mathematical and Theoretical Physics

Joe Bacchus George

Wolfson College

Trinity Term 2024

12,641 Words

Categorical quantum theory and percolation methods for measurement-based quantum computing

Candidate 1081414
Wolfson College

Thesis submitted to the Mathematical Institute
at the University of Oxford in partial fulfilment for the degree of
Master of Science in Mathematical and Theoretical Physics

Trinity Term, 2024

Abstract

It was shown in [7] that universal resources can still be extracted from faulty cluster states in measurement-based quantum computing. This computational model lends itself naturally to the diagrammatic language of ZX-Calculus, a perspective not exploited in the original discussions. Independently, percolation methods have been used to decipher phase transitions in the computational efficiencies of resources. Whilst both percolation theory and the ZX language carry evident relations to graph theory, their symbiosis is yet to be exploited in simultaneity. We hope to promote a bridging of these two perspectives, demonstrating a natural duality for tackling faulty measurement-based systems.

In doing so, the techniques are used to build from the original work of [7], where we propose alternative insights regarding the extraction process of universal clusters. We eventually argue that for certain lattices, the supercritical regime is sufficient for universality in the thermodynamic limit. The results hold innately under both site and bond percolation, as well as differing lattice shapes obeying certain symmetries. Subsequently, we estimate a resource overhead, necessary for real-life cluster state preparations. Finally, our tools are adapted to more recently proposed fusion-based models [3], using network renormalisation techniques. A potential avenue for hypergraph treatments is briefly motivated and concludes our work.

Acknowledgements

I would like to thank Professor Renaud Lambiotte for his kind advice and encouragement in letting me pursue such an interdisciplinary project. Countless directions are immediately inspired by Lambiotte's recommmended readings, his work and our discussions.

I would also like to thank Professor Mathew Hoban for dedicating many patient hours of clarifying confusions, guiding me through mathematics and offering so much of his appreciated time.

Contents

Conventions

$O(N)$ Orthogonal group

$SO(N)$ Special orthogonal group

$U(N)$ Unitary group

$SU(N)$ Special unitary group

$\mathcal{P}_1$ Single-qubit Pauli group

$\mathcal{P}_n$ n-qubit Pauli group

$\mathcal{C}_n$ n-qubit Clifford group

$\mathcal{G}$ Exactly universal set of gates

$\mathcal{G}_{\mathcal{A}}$ Approximately universal set of gates

$\mathfrak{H}$ Hilbert space

$\mathbb{Z}^2$ Square lattice

$\mathbb{H}^2$ Hexagonal lattice

$\mathbb{T}^2$ Triangular lattice

$\mathbb{Z}^3$ Cubic lattice

$\mathcal{O}$ Asymptotic limit notation

$\mathcal{L}_f$ General faulty lattice

Acronyms

BCC Body-centered cubic

BP Bond percolation

CV Continuous variable

FBQC Fusion-based quantum computing

FTQC Fault-tolerant quantum computing

GK Gottesman-Knill (theorem)

KLM Knill, Laflamme and Milburn (protocol)

LCC Largest connected component

LOCC Local operations and classical communication

LOQC Linear-optical quantum computing

MBQC Measurement-based quantum computing

QCM Quantum circuit model

QEC Quantum error correction

RHWW Right-handed wall walking (method)

SHP Shortest path (method)

SP Site percolation

SK Solovay-Kitaev (theorem)

Clarifications

Almost surely means that in the thermodynamic limit for a sufficiently large system, the statement holds with a probability exponentially close to certainty.

Efficient will in general mean that there exists a classical polynomial algorithm in both space and time resources that can implement the described process.

Heralded means that the feature of interest can be made available to the observer.

Necessary means a condition that must first be present in order for another event to occur.

Overhead means the additional quantity required to compensate for a lack in the desired amount.

Sufficient means a condition that will produce another event.

1 Introduction

> "But let your communication be Yea, yea; Nay, nay: for whatsoever is more than these cometh of evil." – A proposal for binary code, **Matthew 5:37**
>
> - *Gary William Flake*

Progress in physics is always accompanied by illuminating representations, ranging from Dirac's bra-kets to Feynman's diagrams for quantum field theory, the trend appears most strongly in the quantum realm. The ZX-Calculus, a diagrammatic language rooted in category theory, offers its own contribution to the description of quantum computation. The tool provides a transparent approach to deciphering quantum architectures and error-correcting protocols. Its applications are particularly catered towards measurement-based quantum computing, given that this was the model that primarily motivated its original development. Here, highly entangled multipartite quantum states are subject to specifically tailored local measurements. Remarkably, this is enough to perform universal quantum computation, offering significant advantages over the original circuit model. Despite these benefits, the initial construction of cluster states has shown to be problematically erroneous.

In [7], cluster state generation is examined through the lens of percolation theory, assessing the impact of errors on underlying graphs representing atoms in optical lattices. This is where the disparate realms of ZX-Calculus and percolation theory share a surprisingly strong symbiosis, in treating faulty measurement-based systems. By introducing the core results of both frameworks, we hope to motivate their natural duality for discerning imperfect quantum resources. To do so, we extend upon the original work of [7], offering new insights and approaches to more recent models of scalable fault-tolerant quantum computing.

We will first provide an alternative to the algorithm in [7], offering a more intuitive way of identifying universal substructures within faulty lattices. By doing so, we argue

that the classical percolation threshold can serve as a sufficient condition for universal computation in the thermodynamic limit, given that the underlying lattice obeys some simple symmetries. For physical applications, a methodology for quantifying an overhead is then introduced, essential for ensuring real faulty lattices can be adapted to harness specific computational capacities. All of these investigations are carried out over both site and bond percolation, representing a different defect based on the underlying physical system.

Eventually, we extend the work to fusion-based models, one of the most recent paradigms for fault-tolerant architectures [3]. To do so, renormalisation techniques are motivated as a method for tackling their more convoluted nature of percolation. Finally, we discuss potential hypergraph extensions as promising avenues for robust cluster state development. A new diagrammatic shorthand is then suggested to further align the ZX language with percolation methods, hinting towards a duality that can be maintained when treating extensions of the current model.

Because of the large disparity between the literature of these two fields, we will spend some time building an understanding of their respective formalisms. Standard results from quantum information, graph theory and the ZX-Calculus are all introduced prior to our discussion of measurement-based computation, where the role of percolation theory falls naturally into place.

2 Quantum foundations

> "Now, what happens if you try to come up with a theory that's *like* probability theory, but based on the 2-norm instead of the 1-norm? I'm going to try to convince you that quantum mechanics is what inevitably results."
>
> - *Scott Aaronson*

Computation is fundamentally a physical process. Its strength is as much permitted as the constraints of nature restrict it. And yet, classical computation is designed to take advantage of laws that are not so fundamental. Quantum computing provides a framework to exploit more foundational rules, regardless of whether these elaborate or tighten previous constraints. In fact, many of the advantages to be gained over classical processing, such as cryptographic usages [12], rely on the strictness of quantum theory as opposed to additional freedom. With this in mind, it is practical to first remind the reader of more primitive quantum theory, given that it is entirely these results that enable a new model of computation. This original formulation will also motivate a subsequent description stemming from category theory.

2.1 Discussing the small

The Dirac formalism of quantum mechanics

To a physical quantum system, we associate a fully descriptive *pure* state to encode its properties. By pure, we mean that we do not account for classical uncertainty regarding the state of our system. We say that $|\psi\rangle$ is a state vector living in a Hilbert space $\mathfrak{H}$, which depending on the feature of interest, is finite or infinite dimensional. In standard quantum information and computing, we are mostly concerned with the finite discrete case and designate our n-dimensional Hilbert space so that $\mathfrak{H} \simeq \mathbb{C}^n$. This generally describes a *qudit*, whilst for a two-level system $n = 2$ we refer to a *qubit*. There also exists

$\langle\psi|$ belonging to the dual space $\mathfrak{H}^\star$, which is found by taking the hermitian conjugate of our state vector. We can represent our system as a linear combination of orthonormal basis states satisfying $\langle\phi_i|\phi_j\rangle = \delta_{ij}$, so that:

$$|\psi\rangle = \sum_{k=1}^{n} c_k \,|\phi_k\rangle \qquad \sum_{k=1}^{n} |c_k|^2 = 1 \tag{2.1}$$

The coefficients $c_i \in \mathbb{C}$ are *probability amplitudes*, differing from classical probability theory in that they are complex numbers whose values can interfere with one another. If there exists more than a single non-zero coefficient, the above defines a *superposition* of states. The *Born rule* specifies that the modulus squared $p_i = |c_i|^2$ offers the probability that our state is measured as $|\phi_i\rangle$ and returns a value λ_i. This is the crucial aspect that distinguishes quantum mechanics from ordinary Kolmogorov probability theory [28].

In our treatment of quantum information, we will consider only qubit systems defined with respect to the *computational basis*.

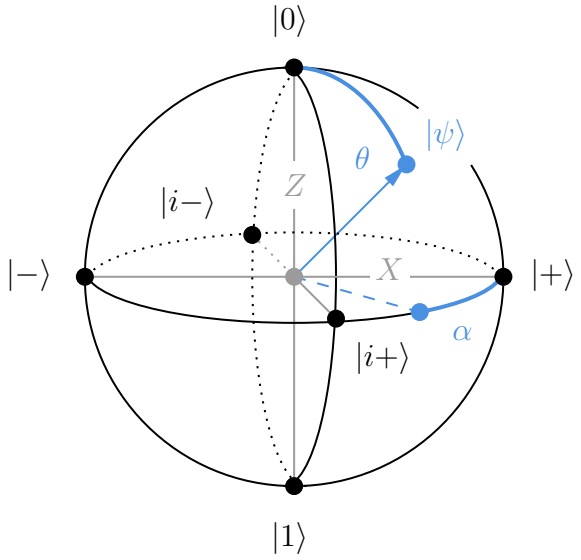
$$|\psi\rangle \in \mathfrak{H} = \text{span}\left\{ |0\rangle := \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle := \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\} \tag{2.2}$$

Eigenstates of the Pauli Z matrix span the Hilbert space, allowing for a general form to be written as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \quad |\alpha|^2 + |\beta|^2 = 1 \quad \forall\alpha,\beta \in \mathbb{C} \tag{2.3}$$

There exists a redundancy in the state vector formalism, which says that $|\psi\rangle \cong e^{i\phi}\,|\psi\rangle$, or rather two state vectors are equivalent up to a global phase. This is simply a manifestation of global $U(1)$ gauge symmetry, a constituent trait of the theory. It is possible then to parameterise our state solely in terms of Polar (θ) and Azimuthal (α) angles, so that $|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + \sin\frac{\theta}{2}e^{i\alpha}\,|1\rangle$. The radial distance is restricted to unity by probability normalisation requirements. Qubit states themselves transform under $SU(2)$, itself a double cover of $SO(3)$, enabling a convenient representation according to the *Bloch sphere* with our newly provided parameters.

However, because of this two-to-one mapping, orthogonality is to be understood between opposite poles of the sphere.



(2.4)

In the Bloch sphere diagram, we have provided shorthand basis states, acting as eigenvectors for the remaining Pauli matrices:

$$X \textbf{ eigenstates} \qquad |0\rangle = \frac{1}{\sqrt{2}}(|+\rangle + |-\rangle) \qquad |1\rangle = \frac{1}{\sqrt{2}}(|+\rangle - |-\rangle) \tag{2.5}$$

$$Y \textbf{ eigenstates} \qquad |i+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle) \qquad |i-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i\,|1\rangle) \tag{2.6}$$

$$Z \textbf{ eigenstates} \qquad |+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \qquad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{2.7}$$

In noisy systems, we may not be so fortunate as to access a full description of our system and may have to account for classical uncertainties regarding states of interest. This motivates a more general formulation using *mixed* states, replacing the familiar state vector with a *density operator* ρ. Although this formalism can be especially useful for harnessing descriptions with decoherence, it is not used in our investigations. We will assume by default that a full description of our system is readily available.

2.2 Multipartite systems

Entanglement and Bell states

To extend our description to multiple qubits, we might define a combined state by taking the tensor product of individual ones. This approach provides a separable form known as the *product state*.

$$|\Psi\rangle = \bigotimes_{k=1}^{n} |\psi_k\rangle = \sum_{i_1 \ldots i_n = 1} c_{i_1 \ldots i_n} |i_1 \ldots i_n\rangle \tag{2.8}$$

For ease of notation, we index our basis states as $|\phi_k\rangle := |k\rangle$ and further write combined tensoring as $|k_1\rangle \otimes \cdots \otimes |k_n\rangle := |k_1 \ldots k_n\rangle$. The order of these indices is associated with the qubits they describe. We may equally consider multi-qubit states not represented by tensoring across individual qubits, systems of this type are said to be *entangled*. Though mathematically the distinction seems trivial, entanglement entirely shifts our description of nature. Primarily, one can identify correlations between qubits at arbitrary distances, though crucially, these do not imply a superluminal exchange of information. It is precisely this property that leads to the topic of *non-locality*, where an object is no longer influenced directly by its immediate surroundings, a primary result that enables many of the quirks exploited in quantum algorithms.

The quintessential 2-qubit entangled states are known as Bell states and are provided below.

$$\begin{aligned} \left|\Phi^+\right\rangle &:= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \qquad \left|\Phi^-\right\rangle := \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \\ \left|\Psi^+\right\rangle &:= \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \qquad \left|\Psi^-\right\rangle := \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \end{aligned} \tag{2.9}$$

For general multipartite systems, we will adopt the convention of ordering our basis *lexicographically*, that is in accordance to binary counting.

For the above 2-qubit case we would have:

$$\mathfrak{H}_1 \otimes \mathfrak{H}_2 = \text{span}\left\{ |00\rangle := \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, |01\rangle := \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, |10\rangle := \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, |11\rangle := \begin{bmatrix}0\\0\\0\\1\end{bmatrix} \right\}$$

2.3 Evolving qubits

Quantum gates and measurement

Just as classical computation involves modifying bits according to specific protocols, quantum computing relies on linear operators, known as *gates*, to describe actions on quantum states. Single-qubit operators can be combined via tensoring to act on multi-qubit states, though some are defined innately as non-decomposable. Under a transformation, the evolved state must retain its quantum-like properties and constraints of the underlying probability theory, features that are maintained by *unitarity*. For an n-qubit system, physical operations must then belong to $SU(2^n)$. Actions on individual qubits then rotate the state around the Bloch Sphere.

In the computational basis, we denote our Pauli matrices as:

$$X := \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix} \qquad Y := \begin{bmatrix}0 & -i\\i & 0\end{bmatrix} \qquad Z := \begin{bmatrix}1 & 0\\0 & -1\end{bmatrix} \tag{2.10}$$

One can define a single-qubit gate, denoted by H for *Hadamard*, which induces a superposition with respect to a basis, a crucial capability for most manipulations in quantum computing. Alongside two other useful unitaries S, the *phase gate*, and T we find:

$$H := \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix} \qquad S := \begin{bmatrix}1 & 0\\0 & i\end{bmatrix} \qquad T := \begin{bmatrix}1 & 0\\0 & e^{\frac{\pi}{4}}\end{bmatrix} \tag{2.11}$$

With additional complex scalar factors for closure, the Paulis form generators for the *single-qubit Pauli group* $\mathcal{P}_1$:

$$\mathcal{P}_1 := \langle X, Y, Z \rangle = \{\mathbb{I}_2, X, Y, Z\} \otimes \{\pm 1, \pm i\} \tag{2.12}$$

By extension, we can define the *n-qubit Pauli group* $\mathcal{P}_n$ as:

$$\mathcal{P}_n := \left\{ P = \bigotimes_{k=1}^{n} P_k \mid P_k \in \mathcal{P}_1 \right\} \tag{2.13}$$

Non-decomposable multi-qubit gates enable much of the richness offered in quantum algorithms. We provide three of these most useful 2-qubit gates below:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \text{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \qquad \text{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

They in turn allow us to generate another useful set of gates, known as the *Clifford* group.

$$\mathcal{C}_n := \langle CNOT, H, S \rangle = \left\{ U \in U(2^n) \mid UPU^\dagger \in \mathcal{P}_n, \ \forall \ P \in \mathcal{P}_n \right\} \tag{2.15}$$

These elements are especially easy to implement experimentally and generate the Pauli group. Somewhat disappointingly, their ease of use allows for the *Gottesman-Knill* (GK) theorem, stating that any quantum circuit composed solely of Clifford members, can be efficiently simulated classically [22]. It is instead more non-conventional gates which hold the key to quantum computing supremacy, which we will explore when discussing universality.

A particular kind of operation is that of *measurement* Π. These act as projections onto basis states that model the outcome obtained in experiment. Our state requires subsequent normalisation afterwards.

Unlike quantum gates, measurements are irreversible and non-deterministic. Upon performing such projections, the state vector governing our system *collapses* to the measured states. Basis state projections are related to an *observable* O with expectation value $\langle O \rangle$.

$$O = \sum_{k=1}^{2^n} \lambda_k \Pi_k \qquad \Pi_k = |k\rangle \langle k| \qquad \langle O \rangle = \langle \psi | O | \psi \rangle \tag{2.16}$$

This enables pre-deterministic insights into likely outcomes but does not unitarily evolve our state, it is simply a tool for predicting eigenvalue statistics. The act of measuring a state in the bases $\{k\}$ is then modelled as taking $M_{\{k\}}$:

$$M_{\{k\}} : |\psi\rangle \longrightarrow \frac{\Pi_{\{k\}} |\psi\rangle}{\sqrt{\langle \psi | \Pi_{\{k\}} |\psi\rangle}} \tag{2.17}$$

Although frequently not specified, quantum computing is treated in the *Schrödinger picture*, meaning that states carry time dependence whilst operators do not. In reality, then, our system evolves in time, as it is inescapably subject to a Hamiltonian, causing time evolutions between measurements. However, the rapidity in which gates and measurements are performed throughout quantum protocols makes these transformations negligible.

3 ZX Calculus and networks

> "The calculations that eventually got me a Nobel Prize in 2004 would have been literally unthinkable without Feynman diagrams."
>
> - *Frank Wilczek*

As enticing as Dirac notation may be, for larger protocols, one can rapidly lose touch with the intuition of more complex quantum processes. Inspired by the diagrammatic notation of Roger Penrose, a graphical reformulation of quantum theory was proposed in the early 2000s. Its language, the ZX calculus with extended ZH elements, is here briefly introduced, though a more rigorous introduction is available at [35]. The language was initially motivated to treat the architectures of measurement-based quantum computing and error correction protocols, the very premise of our ensuing discussions.

3.1 Categorical quantum theory

The roots of ZX-Calculus

In 2004, Samson Abramsky and Bob Coecke proposed an alternative treatment of quantum theory through the lens of a seemingly more distant abstraction [1]. Category theory, and particularly monoidal categories, allows for diagrammatic expressions with *string diagrams*. In the quantum theoretic model, processes are captured more specifically by *dagger symmetric monoidal categories*, which obey algebraic properties reminiscent of the current framework. Here, a *category* of vector spaces is considered alongside *morphisms* as linear maps, turning string diagrams into tensor networks [4]. Consecutive unitary operations and entire quantum architectures are then succinctly represented as webs of these tensors.

Morphisms are preserving maps between mathematical structures of the same type. By structures, we mean sets of objects provided with additional features of interest.

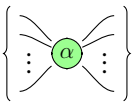
This could be an operation, metric, or even a topology. In our case the morphism maps between two vector spaces such that the operations of vector addition and scalar multiplication are preserved. String diagrams allow for mathematical relationships to be maintained across analogous manipulations diagrammatically. This is where the simplicities of the ZX-Calculus appear so prominently, via their associated rewriting rules, which consist of graphical tricks for performing mathematics. In more elaborate settings, extensions might include additional notations, such as facilitating generalised multi-qubit controlled gates. The latter of which is best visualised according to the ZH-calculus.

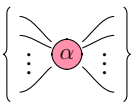
3.2 Diagrammatic tensor networks

Building blocks and their relationship to Dirac notation

ZX diagrams are constructed in terms of generator tensors, which we call *spiders*. These arachnids are equipped with a phase $\alpha \in \mathbb{R}$ which corresponds directly to the Azimuthal angle of the Bloch sphere 2.4. If $\alpha = 0$, then it is omitted visually from the spider's body. Properties are invariant under topological deformations, so it is only a matter of how the connections are made which define linear maps. Composition is carried out by connecting input to output legs in neighbouring diagrams, whilst tensor products are simply represented by stacking diagrams side by side. Scalars are written explicitly next to a diagram, '1' can therefore be ignored whilst '0' would remove the entire frame. Legless spiders are equivalent to scalars.

Only two basis spiders are required to completely define the formalism [33], and these are eponymously chosen to be Z (Green) and X (Red).

Z-spider: m  n $:=$ $Z^n_m[\alpha]$ $=$ $|\overbrace{0\cdots0}^{\times n}\rangle\langle\overbrace{0\cdots0}^{\times m}| + e^{i\alpha}\,|\overbrace{1\cdots1}^{\times n}\rangle\langle\overbrace{1\cdots1}^{\times m}|$

X-spider: m  n $:=$ $X^n_m[\alpha]$ $=$ $|\overbrace{+\cdots+}^{\times n}\rangle\langle\overbrace{+\cdots+}^{\times m}| + e^{i\alpha}\,|\overbrace{-\cdots-}^{\times n}\rangle\langle\overbrace{-\cdots-}^{\times m}|$

The Hadamard gate 2.11 can be generalised and deployed as a more convenient generator by introducing the *H-box* notation, where $a \in \mathbb{C}$:

$$\textbf{\textit{H}-box:}\quad m\left\{\begin{array}{c}\vdots\\ a\\ \vdots\end{array}\right\}n \quad := \quad \sum_{\mathbf{x},\mathbf{y}} a^{x_1\ldots x_n y_1\ldots y_m}\,|x_1\ldots x_n\rangle\,\langle y_1\ldots y_m|$$

By convention, an H-box with $a = -1$ is labelled without a phase, given that this is the standard usage for the familiar Hadamard. With these three notations, we can reconstruct the familiar arsenal for quantum computation. In particular, single-qubit operators, which naturally correspond to spiders with a single input and output, are constructed via relative phase gates:

$$\alpha \quad = \quad Z_1^1[\alpha] \quad = \quad \begin{bmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{bmatrix} \qquad \alpha \quad = \quad X_1^1[\alpha] \quad = \quad \frac{1}{2}\begin{bmatrix} 1+e^{i\alpha} & 1-e^{i\alpha} \\ 1-e^{i\alpha} & 1+e^{i\alpha} \end{bmatrix}$$

And so our Pauli matrices 2.10 alongside the standard Hadamard gate form unitary maps found to be represented as:

$$X \;\to\; \pi \qquad Y \;\to\; i\;\pi\;\pi \qquad Z \;\to\; \pi \qquad H \;\to\;$$

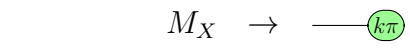
Basis states 2.5 are then implemented as spiders lacking inputs. Because of the $U(1)$ gauge redundancy, we tend to omit global phases and in doing so will instead denote proportionality.

X-basis $\quad |+\rangle \;=\; \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \;\propto$ $\qquad |-\rangle \;=\; \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \;\propto\; \pi$

Y-basis $\quad |+i\rangle \;=\; \frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle) \;\propto\; \frac{\pi}{2}$ $\qquad |-i\rangle \;=\; \frac{1}{\sqrt{2}}(|0\rangle - i\,|1\rangle) \;\propto\; -\frac{\pi}{2}$

Z-basis $\quad |0\rangle \;=\; \frac{1}{\sqrt{2}}(|+\rangle + |-\rangle) \;\propto$ $\qquad |1\rangle \;=\; \frac{1}{\sqrt{2}}(|+\rangle - |-\rangle) \;\propto\; \pi$

Flipping a diagram horizontally takes the self-adjoint, and so mirroring the above would provide dual-state vectors. How might we then choose to represent projection measurements, given that these are how we extract classical information from our diagrams? Let us consider the act of measuring a single qubit in the $|\psi\rangle = |+\rangle$ state along the Z-axis. One would expect an outcome of $|0\rangle$ or $|1\rangle$ with a 50% chance each. To model this post-deterministically with projections, we would say that on the occasion we find 0 then $M_0 |\psi\rangle = |0\rangle$. It would seem as though M_0 would exist as a two-legged spider, but it is not always necessary to consider the collapsed state. In many of the protocols we introduce, the very act of performing the measurement is sufficient in itself, and the collapsed state is not so much of interest. For this reason, we simply associate measurements to single-wired dual states, with an additional phase shift accounting for the unknown outcome.

$M_X \rightarrow$  $M_Y \rightarrow$ $M_Z \rightarrow$ (3.1)

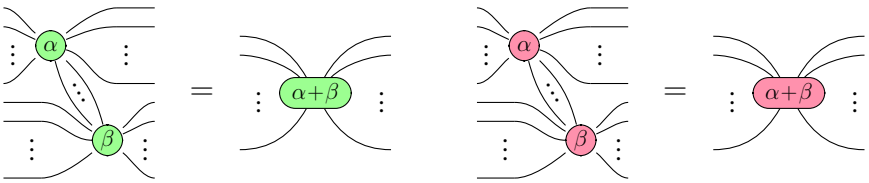
Where $k \in \{0, 1\}$ is dependent on the outcome and would be assigned afterwards according to the obtained experimental value.

3.3 The spider constitution

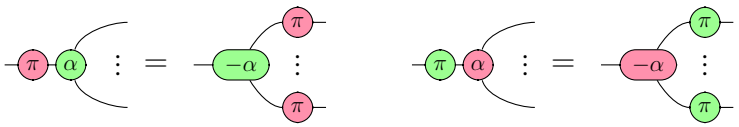
Useful diagrammatic simplification rules

Graphical rewriting rules offer shortcuts for otherwise cumbersome tensor calculus. An exhaustive list is not provided but can be found in [35]. We restrict ourselves to the manoeuvres necessary for our particular investigations. All of the following rules hold equivalently when inverting spider colours, with $\alpha, \beta \in \mathbb{C}$ and $c \in \{0, 1\}$.

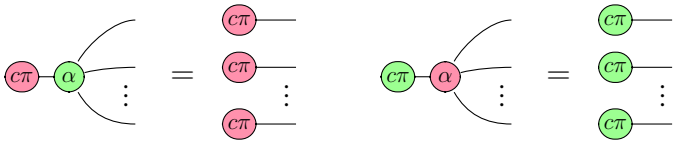
Spider Fusion (SF)



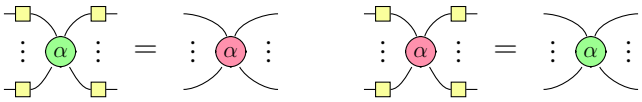
Phase Commutation (PC)



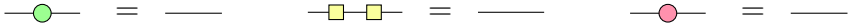
State Copy (SC)



Colour Change (CC)



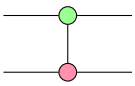
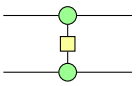
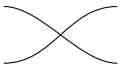
Identity Rules (IR)



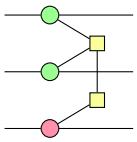
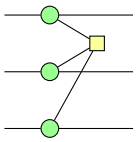
3.4 Extending diagrams

Multi-qubit representations

The familiar non-decomposable 2-qubit gates 2.14 offer representations in the following way:

CNOT →  CZ →  SWAP →  (3.2)

For generalised multi-qubit controlled gates, there exists a natural representation in the ZH formalism, that considers additionally controlled qubits by attaching them to a central H-box. For the 3-qubit alternatives to the CNOT and CZ gate, we have

CCNOT →  CCZ →  (3.3)

The 3-qubit multi-controlled CNOT extension is usually called the *Toffoli* gate and is especially useful for larger algorithms.

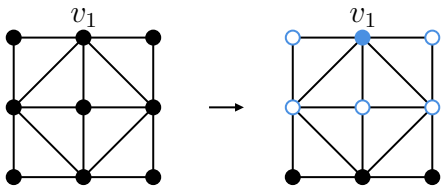
Beyond the H-box, other notations are optionally included for tackling specific aspects of quantum computing. These additions only offer clarity, and can always be translated back to the original spider generators. As it turns out, we will eventually confront graphs uniquely composed of green spiders with Hadamard-equipped legs. Given its evident imitation of a network in the mathematical sense, this allows graph-theoretic tools to come into play, so it is necessary to introduce these terms as well.

3.5 Graph vernacular

Essential terminology from graph theory

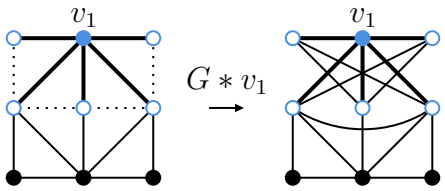
We first define the *neighbourhood* of a vertex v_i to be all nodes directly connected to v_i by an edge. That is all vertices which contribute to the *degree* of our node.

Below we consider a graph G and highlights $N_1(G)$ making up the neighbourhood of v_1.



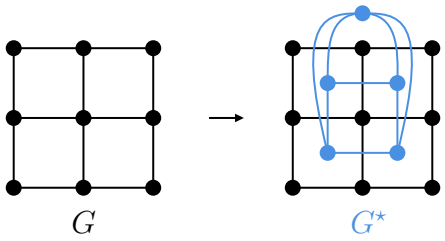
(3.4)

One can then consider the *local complementation* of a graph around a vertex, denoted $G * v_i$. This inverts the edges of the subgraph induced by the neighbourhood of v_i. For the same vertex above we would find:



(3.5)

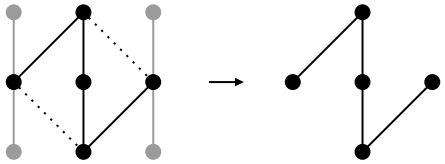
The *graph dual* $G^\star$ takes on a vertex for every face on a planar G, carrying a dual-edge for every pair of faces in G that are separated from one another by an edge.



(3.6)

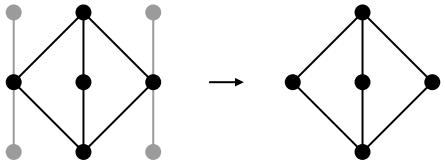
When discussing infinite lattices, the outside vertex on the dual graph tends to be omitted. One can also classify a subgraph H with respect to its parent G. We say that H is a *graph minor* of G if it can be formed from the latter solely by deleting edges, vertices

or contracting edges. An example is provided below, converting a novel G into a minor.



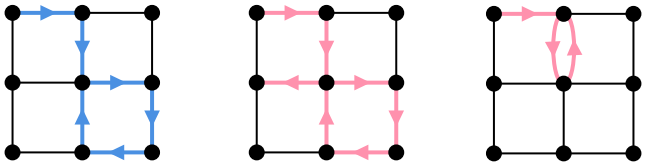
(3.7)

A more restrictive, though useful classification is that of a *topological minor*, which obeys the above principles whilst ensuring that the topology of the original graph G is retained.



(3.8)

Whilst the previous definitions mostly concern structural features of graphs, it is also useful to consider passages within them for computational arguments. A *path* is a finite sequence of edges where both the set of vertices and edges traversed are distinct. An *open path* has differing beginning and ending vertices.



Here we offer three trails, the first graph highlighting a valid *open path*. The other two trails are not paths since the second passes through the central vertex twice and the third passes through its final edge twice. If we then consider any two vertices v_1 and v_2 on a connected graph, the *shortest path* takes the least amount of edges between these two nodes.

4 Measurement-based quantum computing

> "The sculpture is already complete within the marble block, before I start my work. It is already there, I just have to chisel away the superfluous material."
>
> - *Michelangelo*

Quantum computing is, in a sense, a strategic evolution of multipartite states associated with classical information. This is largely the entire ambition, regardless of how it might be achieved. Whilst differing physical setups lend themselves to their respective advantages, different theoretical protocols hold irrespective of the system. We here introduce a particular method of computation that is measurement-based, and although the framework holds generally, we will tackle its implementation via low-energy atomic and photonic-based systems. The proofs regarding universality and feed-forwarding are adapted from [10], though tailored to our particular line of discussion.

4.1 Flavours of computation

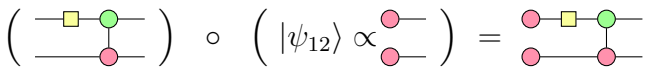
Comparing circuit and measurement-based models

There are many equivalent ways of achieving quantum computation. The ZX-Calculus was primarily born to transparently represent a certain class known as measurement-based quantum computing (MBQC), though it equally provides an excellent range of tools for more standard formalisms. The quantum circuit model (QCM), being the default, is here introduced as a comparison against MBQC,

In QCM, we simply consider an initialised state and perform sequences of unitary operations, acting as our 'computation'. Easily retrievable product states are usually

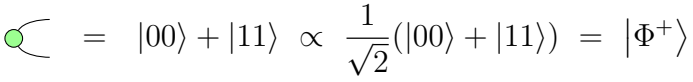
implemented as a first input, though entangled states might also be considered. After evolving, the final state is measured, the action which translates our results into a classical output. Because this induces a collapse, frequently measurements are only made across a subset of qubits, to keep part of the output state for further manipulation.

As a tangible example, it may be desirable to entangle two qubits in the hopes of constructing a Bell state 2.9. To do so, we could consider manipulating both states initialised in $|0\rangle$. The combined object to manipulate is then the product $|\psi_{12}\rangle \equiv |00\rangle$, acted on according to the following circuit.



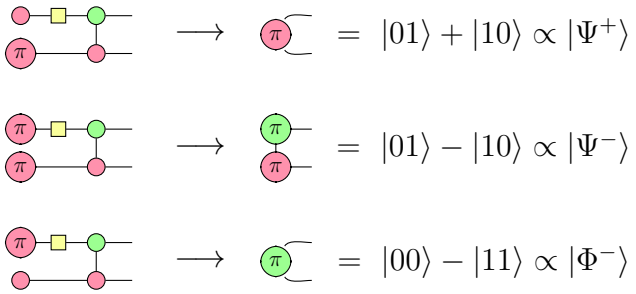
Once again, proportionality is denoted instead of scalar factors because of global phase redundancy. The sequence of operations can now be manipulated with ZX rules, giving the following:

One can read this off directly from the formal spider definition to check that we have the desired result.



To produce the remaining three bell states, one only needs to offer different combinations of initialised states from the Z-basis. These maximally entangled entities are then

obtained with the following circuits.

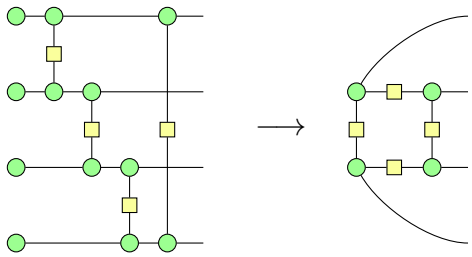


$$= |01\rangle + |10\rangle \propto |\Psi^+\rangle$$

$$= |01\rangle - |10\rangle \propto |\Psi^-\rangle$$

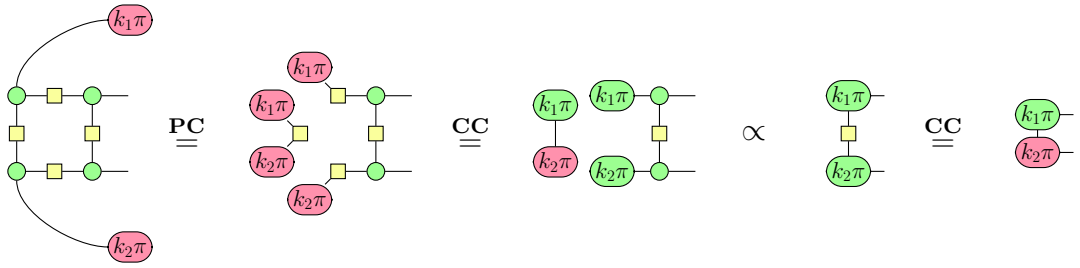
$$= |00\rangle - |11\rangle \propto |\Phi^-\rangle$$

Consider now a much larger, and more convoluted sequence of operations. Regardless of the physical setting, it is not cheap or easy to perform. The initialised state may be desired in an alternative more complex and entangled form, or maybe more convoluted multi-qubit gates 3.3 are required to be implemented. This is the primary motivation behind MBQC, which instead considers an approach where 'computation' is achieved entirely via the measurements themselves. MBQC is sometimes called the *one-way model* of quantum computing to emphasise the non-reversible aspect of computing with observation. What is particularly powerful about the method, is that measurements are performed locally, meaning that we only need to consider individual qubits at a time. Before introducing the technicalities of this approach, let us first rework the above example to see how this might make any sense at all. We begin with a 4-qubit circuit, where tactically placed CZ gates 3.2 are implemented as follows.



The rewritten form on the right is now entangled, but it is not a bell pair as desired. However, if we are tactical about committing measurements, then we can indeed show

that the result is entirely analogous to the previous cases.



In the above process, we implemented measurements on the first and final qubit, leading to four distinct possible outcomes $k_1, k_2 \in \{0, 1\}$. All of the bell states can thus be obtained, regardless of an explicit implementation of the CNOT gate. At larger scales, this is precisely the advantage of MBQC, in that we first construct an easily entangled state to which desirable measurements are performed as a means of computation. However, the outcomes above are probabilistic, and as shown by the end of this chapter, we must adapt succeeding measurements to make our way towards deterministic outputs. In some sense, MBQC chisels away at a highly entangled state, to carve out an equivalent circuit found in QCM.

Initially entangled states seem to appear in a graph-like form, which happens to mirror the physical layout of qubits in the system. This graphical equivalence is one of the reasons why the ZX formalism lends itself so fluently to MBQC: the abstract tensor networks map authentically to their underlying physical relationships. The choice for these particular structures is by no means arbitrary, it provides the correct framework to enable *universal* computation. To see what we mean by this, we must first describe the graph-like forms more generally.

4.2 Cluster states

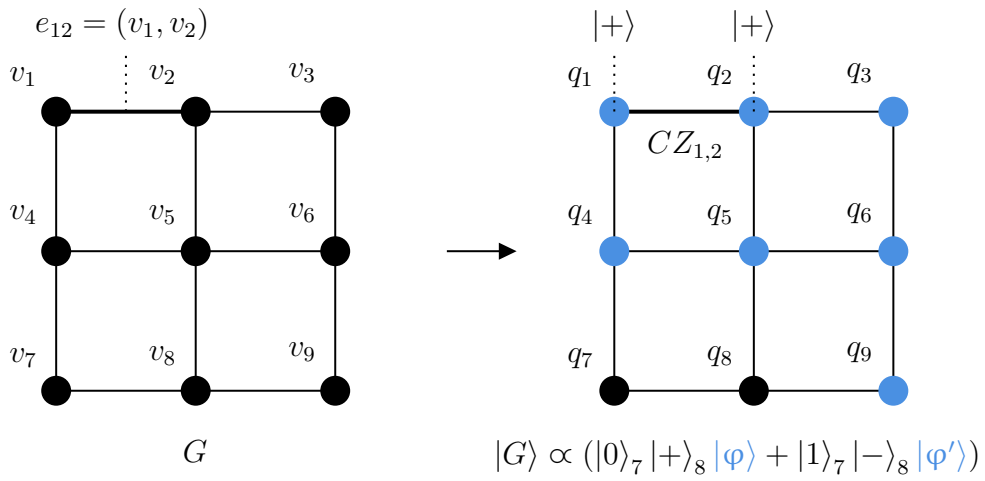
The entangled resources for MBQC

Resources are the quantum states that we initialise our models with. In MBQC, these are called *cluster states* when the underlying graph comprising entangled qubits obeys

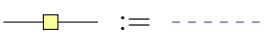
a lattice-like structure. In general, arbitrarily connected graphs can just as well provide excellent resources, though they are not constructed easily in experiments. Consider an undirected and unweighted graph $G = (V, E)$ with V the set of vertices and E the set of edges. Edges are defined as $e_{ij} = (v_i, v_j)$, drawing a connection between both entries to the 2-tuple. A *graph state* is given as:

$$|G\rangle = \prod_{(v_i,v_j)\in E} CZ_{v_i,v_j} \, |+\rangle^{\otimes |V|} \tag{4.1}$$

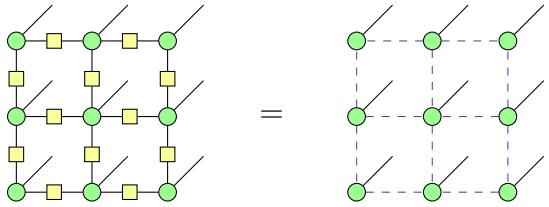
In other words, we define a graph whose vertices are $|+\rangle$ state qubits, and for every edge, we implement a CZ gate across the associated qubits. An illustrative 3×3 grid graph state is shown below.



In the ZX calculus, we say that structures of this form are *graph-like* when all internal wires carry a Hadamard and all spiders are phaseless in the Z basis. The following shorthand makes graph-like states more digestible:



We call the blue dotted wire a *Hadamard edge*, and its conciseness enables the identification of rewriting rules most commonly used in graph-like states, two of which are introduced later. With the above conventions and wires understood as outputs, our grid example in ZX becomes:



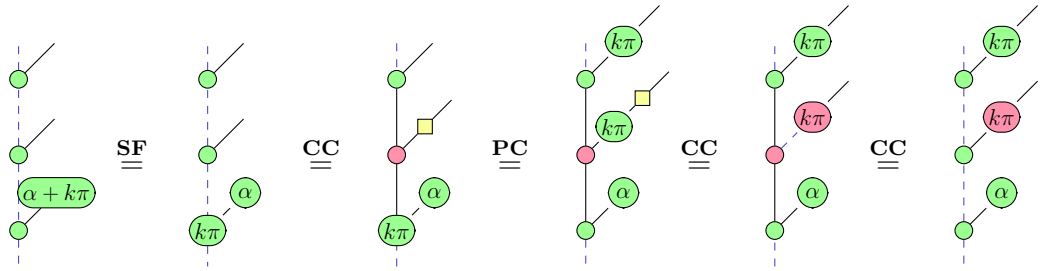
The initial lattice size will entirely determine its computational capacity. But this aspect is secondary to whether its shape is practical. As it turns out, only certain geometries are useful to quantum computing. Given that we are still considering a protocol that produces probabilistic outcomes, how might this be?

4.3 Feed-Forward Corrections

The method of producing deterministic outcomes

The cure to non-determinism is a trick known as *classical feed-forwarding* [10], which makes our measurement tools *adaptive*. Let us consider a 1-dimensional cluster to demonstrate the idea, where we make a measurement in the XY-plane on the first qubit. This means we use an α phased single-wired Z spider, while accounting for the probabilistic outcomes with an additional shift $k\pi$, where $k \in \{0, 1\}$. ZX rules show this to be entirely equivalent to deterministically implementing a measurement on the desired qubit whilst migrating the unwanted outcome-dependent shift to inputs of nearby qubits. In vertical

diagrams, we read the sequence of processes from bottom to top.



The result of k is known to the observer, and so one can adapt subsequent measurements to account for these unwanted phases. Equivalently, one might instead treat the $k = 1$ outcome as an error, and in its manifestation implement corresponding local Pauli gates to correct them, before making another measurement. The process continues across all qubits, something that determines the order in which we should perform measurements across the lattice. In fact, the speed of our computation is solely determined by the classical processing of outcomes, since measurements are instantaneous. This is usually permitted by a classical computer, operating polynomially efficiently. Feed-forwarding further restricts the shape of desired clusters, in that they must enable correctly time-ordered feed-forwarding. All of the lattices we will introduce satisfy this.

A linear one-dimensional cluster state is correctly time-ordered, allowing for feed-forwarding, but it cannot perform the bell-state preparation explored before. To see what sort of clusters can, or host any other arbitrary circuit, the quality of being *universal* is additionally required.

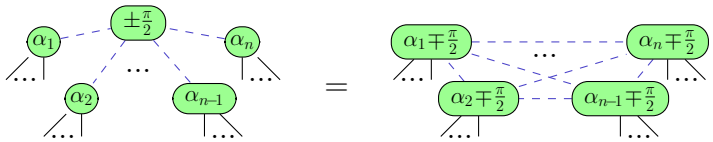
4.4 The measurement arsenal

Tricks that are exploited to reshape clusters

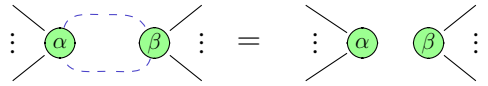
Though we considered the effects of a general XY-plane measurement above, it is not obvious if other variations are even useful. As it turns out, Y and Z basis measurements demonstrate vital preliminary effects for shaping lattices.

We first introduce two additional rewriting strategies:

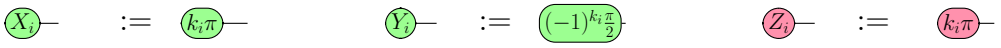
Local Complement (**LC**)



Hopf Algebra (**HA**)



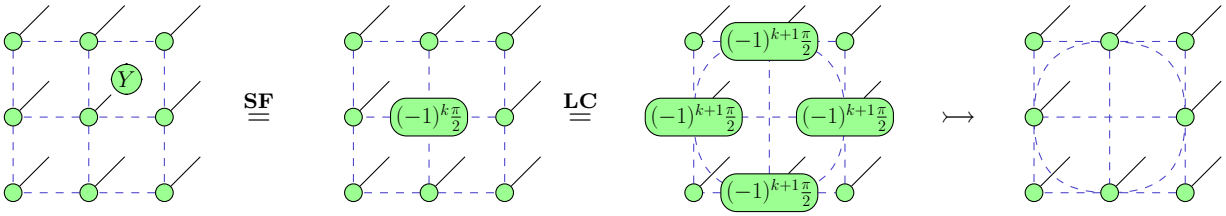
The above equivalently holds for inverted spider colours, though in graph-like diagrams we are not concerned with those cases. If we then consider measurement outcomes on a lattice, their effects can be immediately translated in terms of graph operations. A local single-qubit basis measurement 3.1 on a particular vertex v_i is defined to be $M_i^b : |G\rangle \rightarrowtail |G'\rangle$ where b indexes the basis. We will adopt a tailed arrow notation to mean 'equal up to feed-forwarding and corrections'. In other words, there is some subsequent set of deterministic local corrections that can be made to translate from one end of the arrow to the other. When we interest ourselves in the outcome-independent effects of these measurements we will denote for simplicity:



Our Y basis measurement in effect complements the graph 3.5 around the measured qubit and subsequently removes it.

$$M_i^Y |G\rangle \rightarrowtail |G * v_i - v_i\rangle \tag{4.2}$$

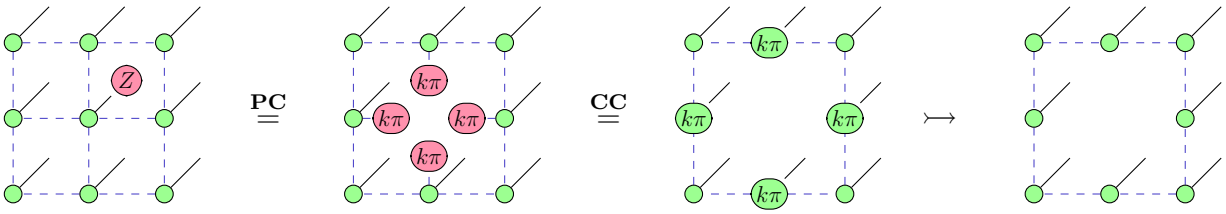
We can see this in effect on our 3×3 grid example:



On the other hand, our Z basis measurement simply removes the measured qubit and all neighbouring wires attached 3.4.

$$M_i^Z \left|G\right\rangle \rightarrowtail \left|G - v_i\right\rangle \tag{4.3}$$

Using the same initial grid, we can see again how this appears with the ZX calculus.



In later applications, we will find the linear results to be most convenient. Here, Y and Z measurements simply contract and delete neighbouring qubits respectively.

(4.4)

It is precisely the effects of these basis measurements that can be exploited to modify the topology of entangled cluster states, whilst YZ-planar measurements will be used for the subsequent implementation gates.

4.5 Universal computation

How to compute anything desired

For single-qubit unitary gates, one can manipulate the Bloch sphere to see that chaining together sequences of arbitrary Z and X rotations allows us to reach any point on the surface. In essence, this is why they are sufficient generators. The separation of a unitary in this form is known as an *Euler decomposition*, enabling a more insightful identification of their operation.

(4.5)

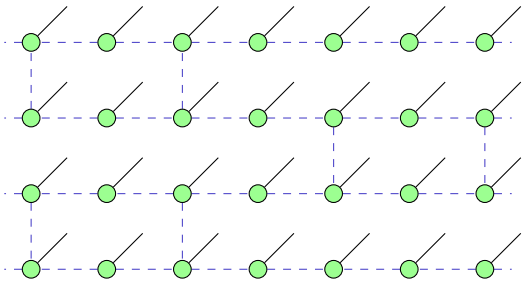
Given that $HZ_1^1[\alpha]H = X_1^1[\alpha]$ by **CC**, we could alternatively represent this decomposition with only Hadamard and Z gates. For generalised computations, unitaries are defined to act on n-qubits, so how might we express these larger unitaries in terms of their generators?

Fortunately, it is sufficient to only introduce the 2-qubit CNOT 3.2 gate for providing a decomposition on arbitrary dimensions []. One can then chain these together with single-qubit decompositions to construct any other desired gate. This is the premise of universality, where we say that a set of gates $\mathcal{G}$ are *exactly universal* if they suffice in constructing all possible n-qubit unitaries. One possibility is:

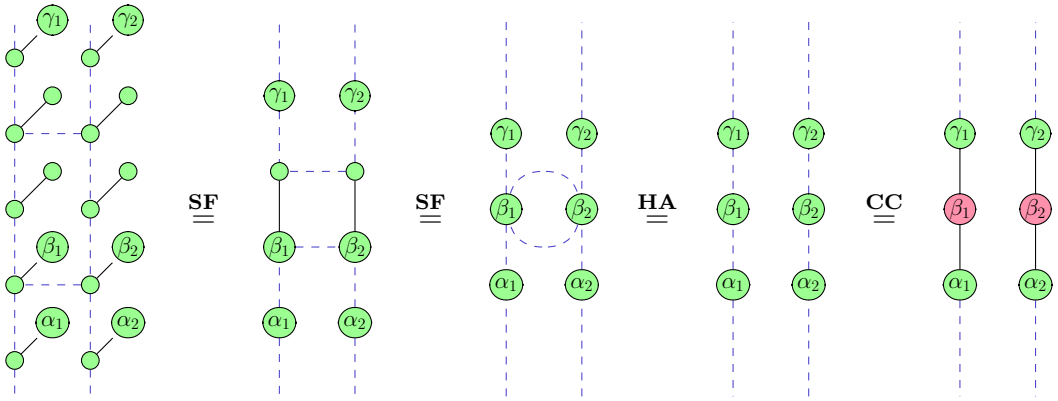
$$\mathcal{G} = \left\{CNOT, H, Z_1^1[\alpha] \mid \forall \alpha \in [0, 2\pi)\right\} \tag{4.6}$$

If we can find a lattice where CNOT and arbitrary single-qubit unitary gates can be constructed via the MBQC scheme, then universal computation is possible on the state. Cluster resources must not only offer a topology capable of being corrected through feed-forwarding but now also allow for such universal computation.

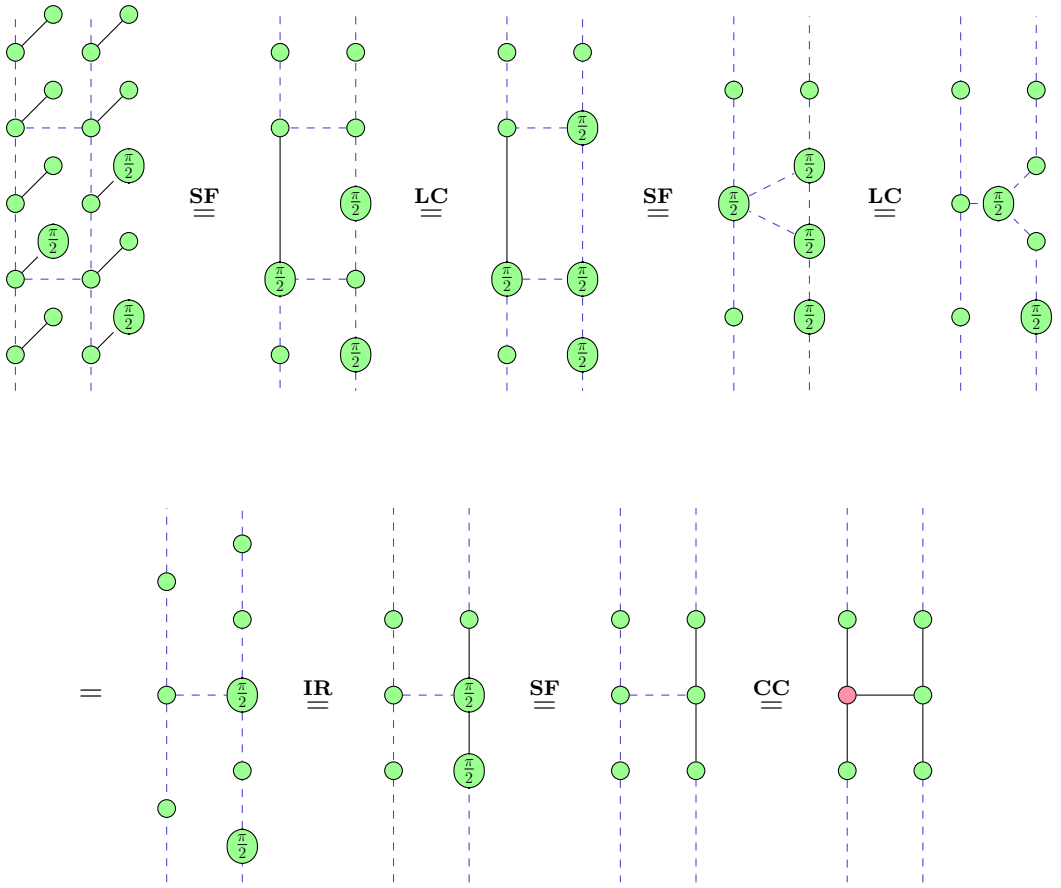
One of the most prevalent structures to satisfy universality is the *Brickwork* lattice, a form that is already correctly time-ordered [34].



Given the repeated structure of the cluster state, it suffices to consider a single component *brick*, and discern if measurements exist to implement any of the elements required from $\mathcal{G}$. Neglecting feed-forward correctable phase shifts, we can perform the following measurements in the XY-plane:



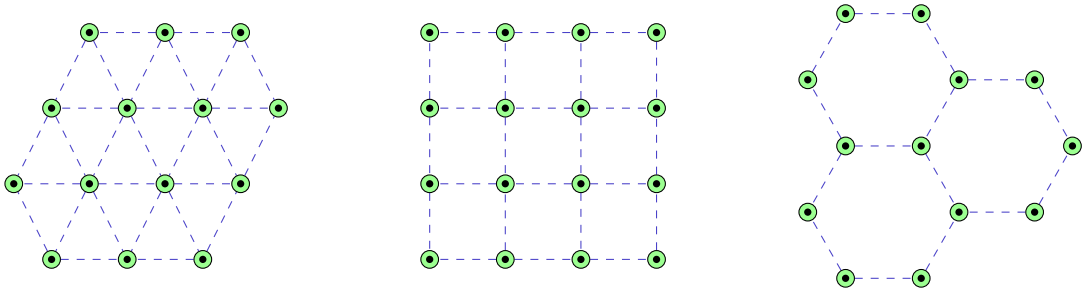
And so both H and $Z_1^1[\alpha]$ are twice implementable within a single brick, which is precisely two adjacent Euler decompositions for unitary gates 4.5. Alternatively, we could try a different set of measurements, to produce a CNOT gate.



Once we scale our brickwork lattice to the appropriate size, we can achieve n-qubit universal computation. Preliminary modifications were not needed and we can immediately select planar measurements for gate implementation. Unfortunately, this is a rare convenience of the bricks themselves. In other graphs that may still be universal, redundant parts are initially measured out before making the computation-inducing measurements.

For a general cluster, we can search for graphs whose topological minors 3.8 are the brickwork lattice. This is possible because of the one-dimensional effects of Y and Z measurements 4.4, which have outcomes corresponding to the allowed transformations defining minor graphs. All three regular lattices spanning the 2-dimensional plane are universal resources whereas neither one-dimensional lattices, tree-like graphs nor certain

complete graphs are [23]. We provide the regular triangular, square and hexagonal forms below. An inner dot is a shorthand for an input wire.



4.6 The efficiency of approximation

Using fewer gates to achieve universality

Apart from Clifford gates 2.15, which take on integer multiples of $\frac{\pi}{2}$ spider phases in ZX, more refined angles are both difficult and expensive to construct. So given the necessity of requiring all infinitesimal values of these phases for the required $Z_1^1[\alpha]$ gate in $\mathcal{G}$, how can we possibly retain universality in the experimental setting?

In practice, we usually refer to an *approximately* universal set of gates when we mention universality. That is, for any desired unitary U, we can construct U_A from a finite set of gates whose sequential combination can arbitrarily approximate the exact form to a desired precision $\varepsilon > 0$. That is to say:

$$|U|\psi\rangle - U_A|\psi\rangle| < \varepsilon \qquad U_A = \prod_i^{N(\varepsilon,U)} U_i \qquad \forall\, U_i \in \mathcal{G}_A \tag{4.7}$$

Where $N(\varepsilon, U) \in \mathbb{N}$ is a finite number of gates to be implemented to approximate U dependent on the precision ε. The set $\mathcal{G}_A$ contains gates that enable approximate universality. Most commonly, we can consider the Clifford gates with an additional, although more costly, T gate.

$$\mathcal{G}_A = \{CNOT, H, T\} \tag{4.8}$$

This additional unitary disables the GK theorem, meaning that is uniquely efficient to achieve circuits comprised of $\mathcal{G}_A$ with a quantum computer. The S gate from the original Clifford set can then be recovered by $T^2 = S$. The underlying argument as to why this novel collection of gates suffices in approximating U, is that the sequences $THTH$ and $HTHT$ offer two distinct *irrational* rotations around the Bloch Sphere, meaning we can chain them together to approach arbitrary surface points. A more complete discussion is found in [10].

Although finite, we would expect the number of elements in $\mathcal{G}_A$ required to approximate a gate to depend strongly on the desired operation. Surprisingly, the *Solovay–Kitaev* theorem (SK) [11] finds an efficient way of finding short sequences of gates that reach the desired precisions of ε. The number of replacement unitaries then scales according to:

$$\exists\, \gamma \in \mathbb{R} \quad N(\varepsilon, U) \sim \mathcal{O}\left(\log^{\gamma}\left(\frac{1}{\varepsilon}\right)\right) \tag{4.9}$$

Not only does this hold for single-qubit unitaries, but also more generally, meaning that one can always find combinations of elements in $\mathcal{G}_A$ to efficiently approximate $SU(2^n)$. From here onwards, by universality, we are always implying approximate universality. The implications on MBQC are then manifested through restrictions of measurement.

We have then a framework of universal resources, to which non-deterministic measurement outcomes can be corrected. The available computational power can still be exploited by a limited set of measurements, as long as these can generate all the elements of $\mathcal{G}_A$. The last step is to harness these theoretic models physically.

4.7 Entanglement bakeries

The physical constructions of cluster states

The preparation of a cluster state seems facilitated by the commutativity of CZ gates among themselves. Because of this, the order in which they are applied does not matter, theoretically allowing for the easy growth of larger states. It is instead the physical setting

of applying these entangling operations, which harms their successful development. To see this, we will consider two differing methods of implementing resources, motivated in [16], that are innately related to their physical setting of computation.

In the *static* approach, the underlying lattice of the cluster state exploits the physical layout of qubits. This is the case for low-energy atoms in optical lattices, where periodic arrays of laser trapping sites determine the locations of qubits. These are then subject to Ising-type interactions, implementing the desired CZ gates. Such protocols are usually performed on a square lattice, though they are particularly prone to site defects, where imperfections leave empty sites without atoms [7]. Failed CZ gates in this model are easily correctable, as one can perform *surgical* measurements to disentangle failed regions before reattempting entanglement. However, the loss of qubits themselves persists as missing vertices in the associated graph.

Alternatively, the *dynamic* approach uses optical CZ gates to actively entangle the system. These are especially useful for linear optical quantum computing (LOQC) and other photonic-based systems [25]. Until 2001, it was thought that the limitations of bosonic systems prevented the construction of quantum circuits using only linear optics. That was until Knill, Laflamme and Milburn proposed a protocol (KLM) [17] allowing for scalable universal computation, results that extend to MBQC. In these schemes, gate implementation is only achievable non-deterministically, so many qubits are left unentangled from the global lattice. Unlike the vertex-based faults of the static scheme, errors here are instead related to missing edges of the underlying graph. Whilst photonic loss, causing missing vertices, can also be apparent in this scheme, we will leave our treatment of simultaneous errors to the final chapter.

Both methods offer unique advantages for generating resources but are equally prone to specific production errors that need treatment. Fortunately, both types of defects are heralded, meaning that we can detect the presence and location of these errors. The challenge then is to use this information to ensure that the faulty clusters are still computationally useful.

5 Sculpting from the erroneous

"It is the system and its fragility, not events, that must be studied."

- *Nassim Nicholas Taleb*

With more defects, the topology of the underlying cluster states will change. Too many faults may eventually destroy the structure that itself allows for universality. In [7], an algorithm was devised for finding a universal lattice within a faulty one. The approach considers a mapping of the problem to percolation theory, locating a phase transition in the computational power of MBQC resource states subject to random errors. We here provide an overview of this approach in the language of ZX, and extend on the work from [7]. Whilst the technique bridges two seemingly distant fields, we seek to promote their symbiosis via the diagrammatic language.

5.1 Percolation thresholds

An outline of important results

The study of graphs which are subject to probabilistic vertex and edge removals form the premise of *percolation theory*. The seminal results concern phase transitions across these networks at critical probabilities. Faulty cluster states that lack either qubit or entanglement gates, are mapped to problems of *site* (SP) and *bond* percolation (BP) respectively. Given that these errors occur as independent events, we are more specifically interested in the results of *Bernoulli percolation*, and will initially focus on models considering planar graphs.

In [7], only errors reminiscent of static preparation are considered, meaning that the model considers site percolation where the chances of a qubit successfully forming is p. One might prefer to call this faulty graph a *network* and expand on the native terminology

to this related field. Here, we are frequently interested in the sizes of *clusters* within a non-trivial graph. These are isolated but fully connected chunks of the graph. The *percolation threshold* p_c is the concentration p at which a *unique infinite cluster* suddenly appears for the first time over an infinite lattice. Below this threshold, only finite clusters exist. The results we discuss are true *almost surely* in the thermodynamic limit otherwise definitions are poorly defined. However, the approximations remain robust even at smaller scales. For an $L \times L$ square lattice we find that the finite-size scaling accuracy at the critical threshold obeys [31]:

$$p_c(L) - p_c(\infty) \propto L^{-\frac{3}{4}} \tag{5.1}$$

In other words, even for finite graphs, we can still expect good approximate results at criticality. The following universal lattices have known, although not all exact, values for p_c as provided by [9]:

Percolation Type	**Site** $p_c(\infty)$	**Bond** $p_c(\infty)$	**Degree**
Hexagonal Lattice ($\mathbb{H}^2$)	~ 0.6962	~ 0.65271	3
Square Lattice ($\mathbb{Z}^2$)	$= 0.592746$	$= 0.5$	4
Triangular Lattice ($\mathbb{T}^2$)	$= 0.5$	~ 0.34729	6
Cubic Lattice ($\mathbb{Z}^3$)	~ 0.3116	~ 0.2488	6

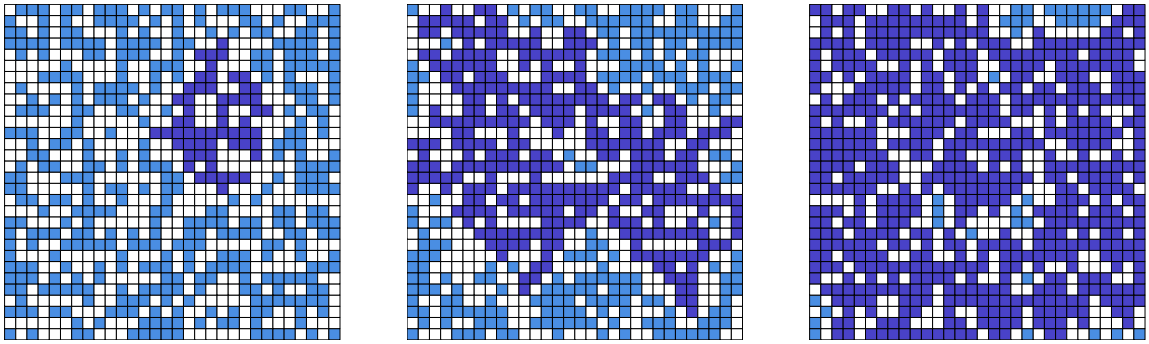
Some insights into the above properties can be obtained by a relationship relating the dual lattices 3.6, which states that the critical bond threshold satisfies:

$$p_c = 1 - p_c^\star \tag{5.2}$$

Since $\mathbb{Z}^2 = (\mathbb{Z}^2)^\star$ then for square lattices the threshold must be one-half. Likewise, we have $\mathbb{H}^2 = (\mathbb{T}^2)^\star$, which is apparent from their complementing bond values.

Consider the standard case of a finite $\mathbb{Z}^2$, where it becomes apparent what distinguishes the regimes outside criticality. We here provide an example for $L = 25$ subject to site percolation at the *subcritical* ($p < p_c$), *critical* ($p = p_c$) and *supercritical* ($p > p_c$)

regimes respectively. We illustrate the graph in a cellular form for clarity, though it is entirely analogous to treating it as a network. The dark blue regions represent the largest connected component, lighter blue are available sites and empty ones are left blank.



$p = 0.5 < p_c$ $p = 0.6 \sim p_c$ $p = 0.7 > p_c$

Crossings consist of paths that traverse a lattice from one side to another. For $\mathbb{Z}^2$, we define H and V crossings for horizontal and vertical paths between their associated borders, features that are exploited in the algorithm we introduce shortly. When we consider alternative lattices like $\mathbb{T}^2$, crossings are less natural to define given the odd number of sides, but we do not require these considerations in our investigation.

5.2 The chiselling algorithm

A method for identifying universality from faulty lattices

We will first consider a resource state subject to site percolation, as does [7] for the static low-energy atomic case. Optical lattices are frequently implemented as a $\mathbb{Z}^2$ structure, though when subject to unpredictable errors, it is not clear how the graph might evolve. We again are looking to identify an alternative universal cluster, whose graph is a topological minor 3.8 of a subset of qubits stemming from the initial faulty graph $\mathcal{L}_f$. By reintroducing the contraction and deletion effects of the Y and Z basis measurements, and neglecting feed-forward post-processing, we can further simplify less obvious subgraphs

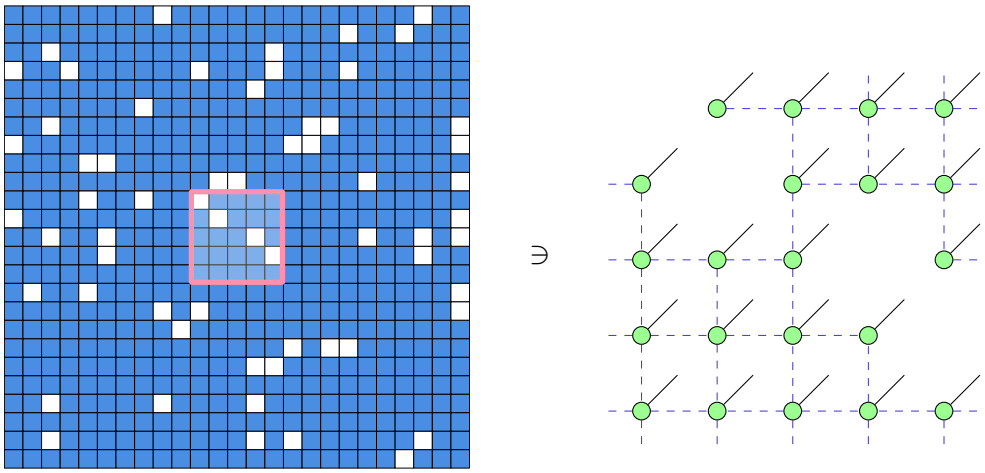
into more familiar universal shapes. The simplest regular lattice to devise from $\mathbb{Z}^2$ is that of $\mathbb{H}^2$, since it carries the lowest average degree for internal nodes. This is useful when finding methods to trace it out from $\mathcal{L}_f$, given that high-degree nodes permeate less as faults increase. If we successfully identify a suitable universal structure, then we can confidently reclaim universal computation in the faulty state, though it's computational capacity may be weaker.

The algorithm of [7] uses a range of intricate methods to both optimally and efficiently identify a useful subgraph. We here propose an alternative method that allows for a more intuitive illustration of the underlying process. Later, it is shown that our approach still manages to identify the same proportion of crossings as the original proposal. We call this new method the shortest path (SHP) algorithm, the code for which is available in the **appendix**.

Firstly, classical pre-processing locates a suitable region for a universal subgraph and then assigns corresponding measurements to specific cells. Though these assignments are made at the end, the associations between the cellular representation of the physical system and ZX form are provided. Computationally, we assign the colour scheme to numerical inputs of a matrix.

Type	Fault	Qubit	Path	Junction
Grid scheme	□	■	■	■
ZX scheme				

The method of SHP is here described whilst providing examples for $L = 25$ and $p = 0.9$. We begin with a faulted lattice, where the ZX scheme above is yet to be applied, and so associate all ■ cells to $|+\rangle$ states as shown.

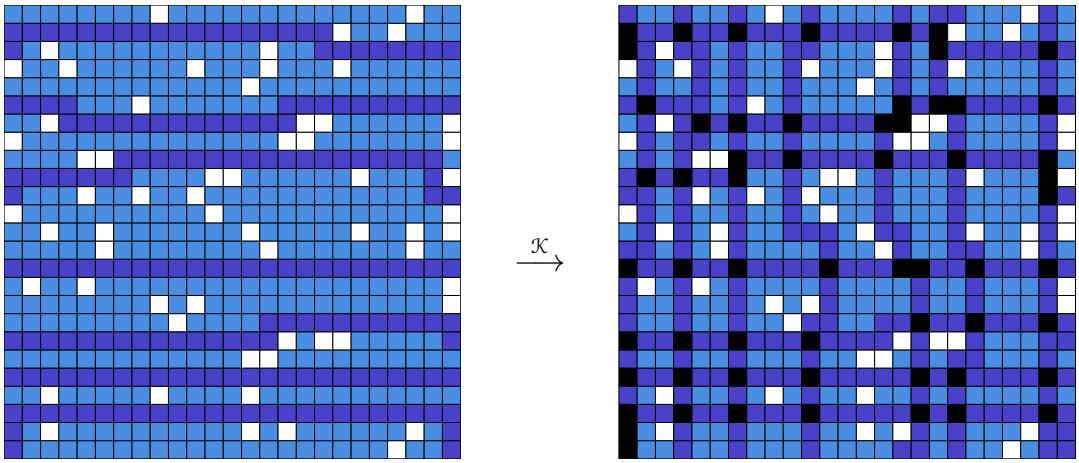


Cells are associated to v_{ij} with $i, j \in R = \{1, ..., L\}$, reflective of the underlying vertex correspondence. A border is then chosen, and scanned along, checking whether a path exists between the current vertex and the opposite side. We restrict these to only attempt vertices directly across, to better balance out spacing and efficiency purposes. If identified, and validated through the additional conditions considered below, the shortest path is drawn. These form subsets of the H and V crossings, and we label them as S^H and S^V respectively. For clarity, we have defined boundary indices $B = \{1, L\}$ and a function $\mathfrak{S}$ which returns the shortest path between any two input cells $v_{i_1 j_1}, v_{i_2 j_2}$ subject to the condition $\hat{c}$, but returns nothing if there is no path at all.

$$S^H = \left\{ s_k^H = \mathfrak{S}(v_{k1}, v_{kL}, \hat{c}) \ \forall \ k \in R \ \mid v_{kb} \neq 0 \ \forall \ b \in B, \ \hat{c} \to \blacksquare \notin \mathcal{N}(v_{ij}) \right\} \tag{5.3}$$

$$S^V = \left\{ s_k^V = \mathfrak{S}(v_{1k}, v_{Lk}, \hat{c}) \ \forall \ k \in R \ \mid v_{bk} \neq 0 \ \forall \ b \in B, \ \hat{c} \to \blacksquare \notin \mathcal{N}(v_{ij}) \right\} \tag{5.4}$$

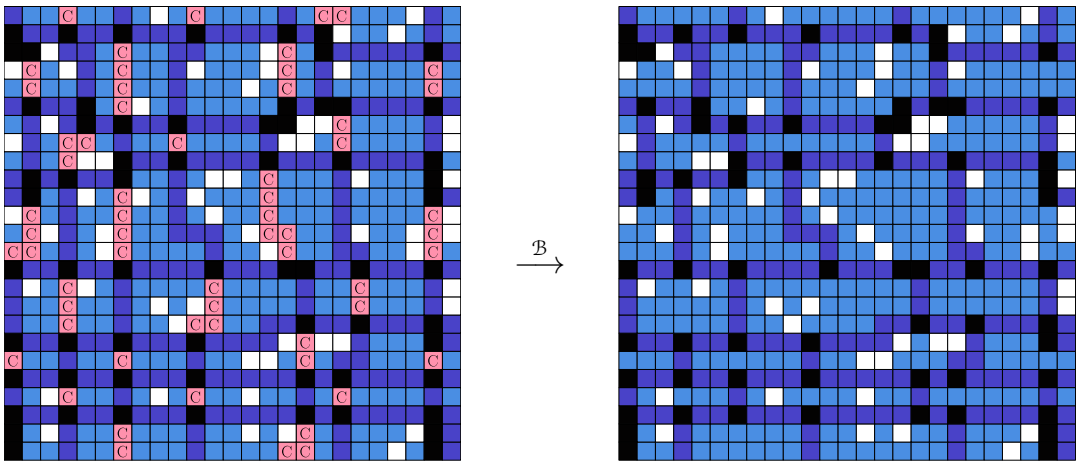
In short, we collect the shortest paths between opposite borders under the condition that two paths of the same crossing type are never adjacent nor intersecting. This ensures that we have the correct spacing to later perform measurements between, vital for creating the universal minor. Both orientations of crossings are identified and combined in a process called $\mathcal{K}$, with junctions specified as their overlap.



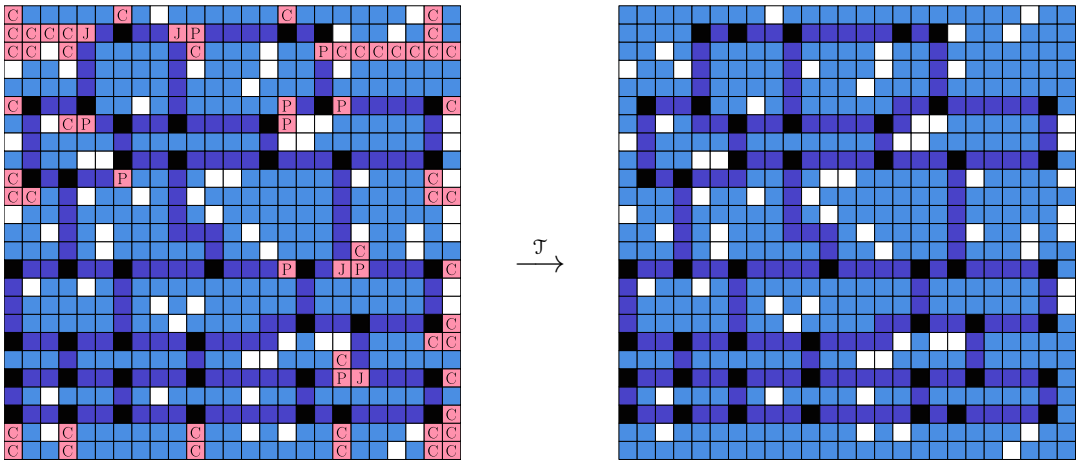
Two parts of processing then take place. The first strategically neglects every other inner *bridge* over a single orientation. These bridges consist of path sections between two junction cells, best revealed in the exemplary diagrams. This action is referred to as *bridge decomposition* $\mathcal{B}$. Highlighted regions are labelled according to the transformation imposed by the procedure as specified below, where the neutral shade represents any non-empty cell type.

C : ■ ⟶ ■ $\quad$ P : ■ ⟶ ■ $\quad$ J : ■ ⟶ ■

The results of bridge decomposition are then shown with the identified neglected regions highlighted.

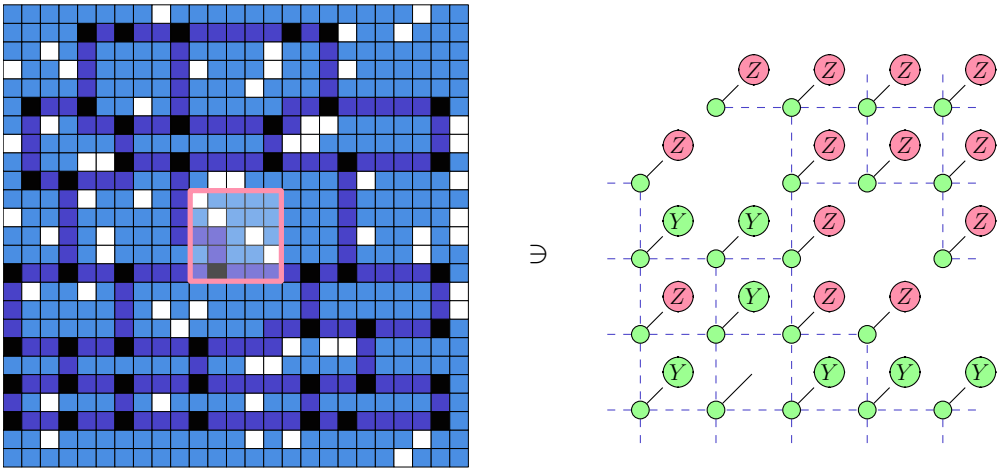


This is now more revealing of the desired hexagonal form, though more intricate modifications are required to sharpen the results. This is where the secondary process comes into play, as it removes redundant trails and clustered path regions. We call this chain of modifications *trimming* $\mathcal{T}$. It also includes some internal modifications that re-identify and retain necessary junction cells, to correctly designate them to suitable measurements.

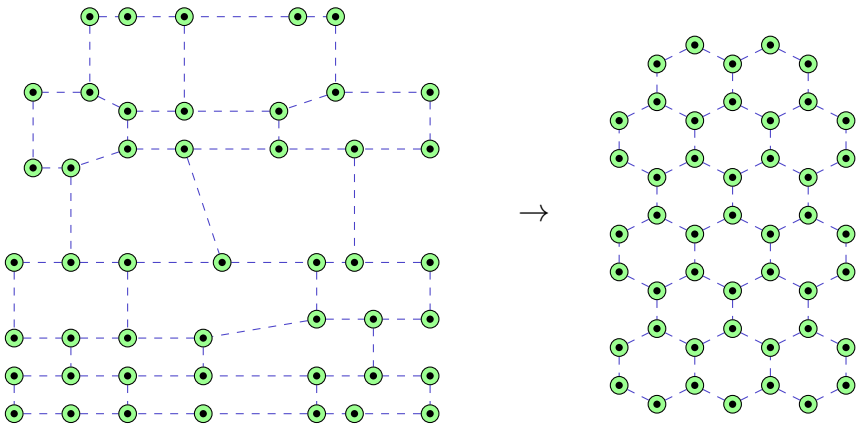


The identified structure and cell mappings are passed over for quantum processing so that measurements are made according to the ZX scheme. We can see how this manifests

more closely on the same subsection of the processed lattice:



Because ZX diagrams are invariant under deformations, the new resource state is then innately equivalent to the universal $\mathbb{H}^2$. This can now be used as a completely unfaulted universal resource for MBQC.



However, it is not always possible to spot these topological minors, given that the errors may be severe enough to prevent any universal formations in $\mathcal{L}_f$. Their appearance is in fact entirely correlated to percolation thresholds, the reason for which the symbiosis between these distant fields exists. To best quantify these, [7] proposes entanglement as

an *order parameter* for a phase transition of computational efficiency.

5.3 Monotones and phase transitions

Quantifying critical error rates with entanglement

In general, it is extremely difficult to classify the amount of entanglement in a multipartite system, so many differing approaches can be found across the literature [32]. *Entanglement monotones* μ, are functions which attempt to quantify this and share the following set of conditions.

- $\mu(|\psi\rangle) \geq 0$ (Positive semi-definite)
- Invariance under local unitary transformations.
- Invariance under *local operations and classical communication* (LOCC).

This last condition occurs when a local operation is executed on a subsystem of a multipartite state, and the outcome is subsequently communicated classically to another party. They then conditionally implement a local operation on their respective subsystem. Despite appearing arbitrary, this definition reflects a reality of entanglement that monotones themselves should obey, in order to be considered viable.

For regular lattices, a particularly suitable measure is that of *entanglement width* μ_W, a natural candidate for tackling cluster states given its inspiration from graph theory [36]. The most useful feature of this monotone to our purposes is that it obeys the following equality:

$$\mu_W \left(\bigotimes_k |\psi_k\rangle \right) = \max_k \mu_W \left(|\psi_k\rangle \right) \tag{5.5}$$

That is, the measure of entanglement on a product state is entirely equivalent to the largest contribution of any individual sub-state. Disconnected clusters within the lattice are themselves no longer entangled with their surroundings, so μ_W for a cluster state can be entirely deduced from the single most entangled component. Because of the regular

lattice structure, the largest connected component (LCC) then bounds the amount of entanglement available to the global cluster state.

In MBQC, the amount of entanglement present in the resource state predefines the amount that can possibly be exploited in a quantum algorithm [10]. One can never use measurements to produce a more entangled state, a property emphasised by its invariance under LOCC. Relatedly, entanglement provides a baseline for the computational capacity of a state. In fact, it is *necessary* that the entanglement width of a resource is unbounded for it to ever be universal [24]. By unbounded, we mean that there exists no upper limit to the amount of entanglement that the resource can possess, regardless of its scaling. This can be understood from CNOT gates, the only 2-qubit elements of $\mathcal{G}_A$ which perform entanglement 4.8. In a faulty MBQC lattice, regions capable of containing these gates must then already exist, and should not be limited to a finite region if the lattice necessitates scaling.

From this perspective, it is clear that the only fully connected component that scales in accordance to these requirements appears in the supercritical regime, as predicted by classical percolation theory. Yet 5.5 immediately translates this into the upper bound for available entanglement. It is then necessary for our faulty lattice to be situated beyond p_c for it to ever be universal. On the contrary, it is almost surely guaranteed that the subcritical regime offers no deployable subgraph.

The argument in [7] considers the scaling properties of LCC components on $\mathbb{Z}^2$ to decipher the above remarks with μ_W. It is further shown that in the subcritical regime, one-way computation can be efficiently classically simulated. These findings allow us to distinguish the two regimes of percolation theory as a phase transition for universal quantum computation. But what does it tell us about the available computational power, if we do successfully identify a universal minor?

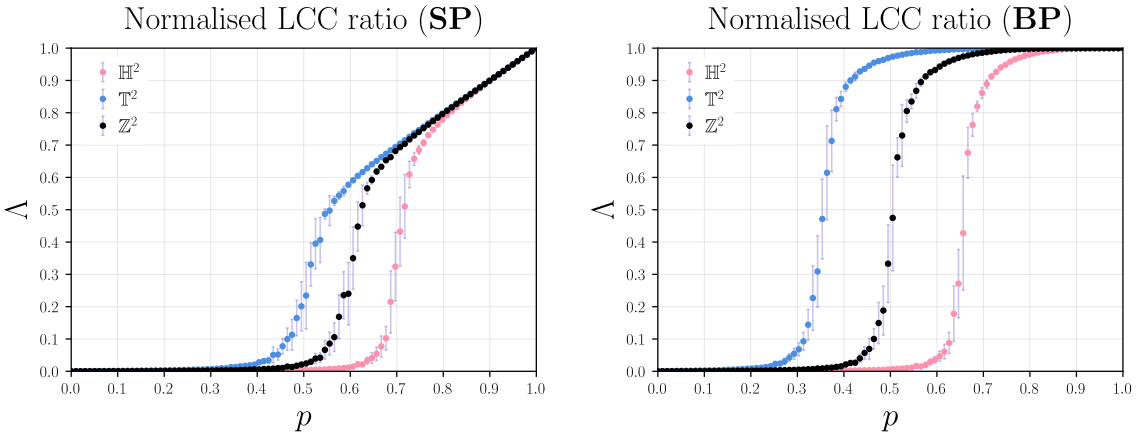


Figure 5.1: Numerical calculation of normalised LCC ratios Λ for all 2-dimensional regular universal lattices, subject to site percolation (left) and bond percolation (right). Computed for $L = 100$ statistically averaged over 50 repeats with standard deviations displayed.

Numerical solutions for Λ, which monitors the LCC size normalised to the total lattice, are evaluated for the universal planar lattices in *fig. 4.1*. According to 5.5, the LCC limits entanglement linearly in the supercritical regime for SP. Whereas nearly *all* of the available capacity could be provided above p_c for BP. It would seem as though the two forms of percolation differ strongly in terms of how much computational capacity could be available to a universal subsystem. But this does not account for the actual structure that the LCC is likely to manifest.

Recall that topology played the initial governing role in identifying universality, meaning that much of the contained entanglement could well be unexploitable from the LCC. In fact, we already encountered many non-universal graphs, whose geometries would in turn exhibit high entanglement width. In the next section we will propose an argument from the perspective of crossing paths to instead motivate the supercritical regime as both a *necessary* and *sufficient* condition for cluster state universality in the thermodynamic limit.

5.4 Symmetry is sufficient

Why the supercritical phase can sometimes guarantee universal subsystems

When a percolating LCC cluster appears, crossings innately manifest, since the largest component is expected to span the range of it's underlying lattice. Crossing path intersections are further guaranteed, given that the component is fully connected. If our lattice is asymmetric, some crossings appear sooner than others, because their respective boundaries are closer. This is what the analytical work of John Cardy presented in his work [8]. Here, hypergeometric functions and tools from conformal field theory helped develop mathematical estimations for *crossing probabilities* ϱ_k, the likelihood that crossing paths appear between the sides specified by k. Though in general, the project of determining ϱ_k remains largely incomplete. It is best then to approach these quantifications numerically, though we can first make some simplifications given the symmetries of real cluster states.

We again consider the $\mathbb{Z}^2$ case, to lead from the work of [7]. The regular grid presents equal sides, meaning that a $\frac{\pi}{2}$ rotational invariance is exhibited. Accordingly, H and V crossings are expected at the same frequency, in the scaled limit, given that errors are independent events. These arguments extend naturally to other regular lattice shapes, as long as they present both even-numbered and equal-lengthed sides. The immediate simplifications are twofold:

$$\varrho_H \sim \varrho_V \sim \varrho_{H\cap V} := \varrho \longrightarrow \left|S^H\right| \sim \left|S^V\right| :\simeq |S| \tag{5.6}$$

These results can also be found via the self-duality of $\mathbb{Z}^2$ [18], but we maintain the previous explanation for generality. In essence, 5.6 finds the likelihood of both H and V crossings to appear on $\mathbb{Z}^2$ to be equal. In turn, the number of crossing paths are also expected to be equivalent from either orientation. Whilst ϱ will remain independent of the algorithm devised, $|S|$ will depend on our method of extracting crossings within the conditions necessary to weave a universal subsystem.

Crossings will still manifest regardless of whether a designated algorithm can identify

them. For a symmetric lattice in the above sense, the presence of a spanning LCC above p_c by definition will always contain such intersecting crossings. This is shown via our SHP method, though not exclusively to it, to be sufficient criteria for tracing out an underlying universal structure. For the finite case, on close proximity to p_c, the lattice itself might not exhibit a desirably large, or even any universal structure. However, it still carries the possibility of being rescaled so as to eventually offer one, since crossings are more likely than not to reoccur in the supercritical phase. In the infinite case, rescaling is never needed, so the arguments will hold true always. In this sense, the thermodynamic limit offers a guaranteed universal subsystem above p_c.

To see these predictions numerically, we provide estimations for ϱ in *fig. 4.2*. Here, the probability of crossings grow exponentially at the critical threshold before saturating towards unity. Though an identified universal subgraph might not take advantage of all these paths, their presence suffices in confirming that a tracing is possible from p_c.

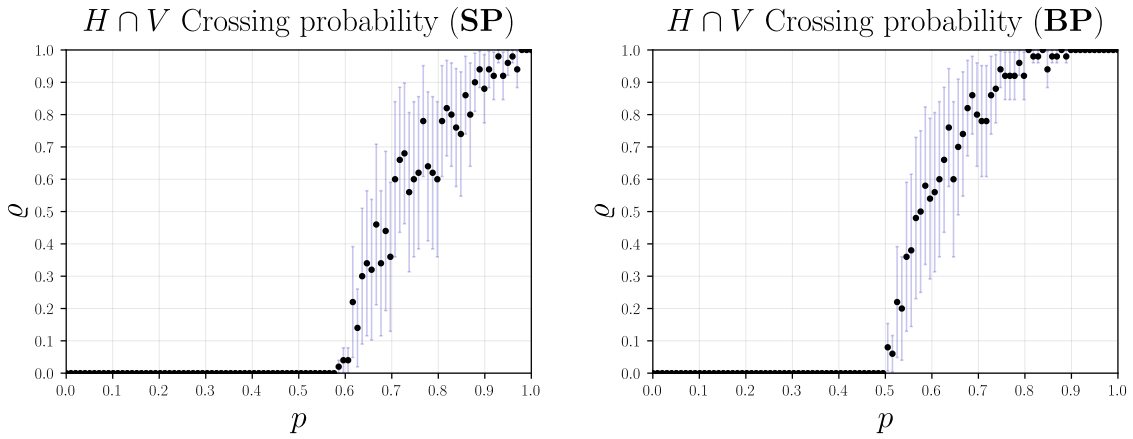


Figure 5.2: Numerical calculation of simultaneous crossing probabilities ϱ on $\mathbb{Z}^2$ for site percolation (left) and bond percolation (right). Computed for $L = 100$ statistically averaged over 50 repeats with standard deviations displayed.

To account for the additional spacing conditions enforced by the SHP algorithm, we further simulate $|S|$ as calculated by our carving algorithm. In [7], a constrained right-handed wall walking (RHWW) algorithm is instead the proposed method of identifying

crossings. We find that the numerical results for our SHP method, see *fig. 4.3*, scales in accordance to the results of RHWW in the original paper. Close to the percolation threshold, SHP only differs in that it offers a linear-like scaling compared to the polynomial growth of RHWW.

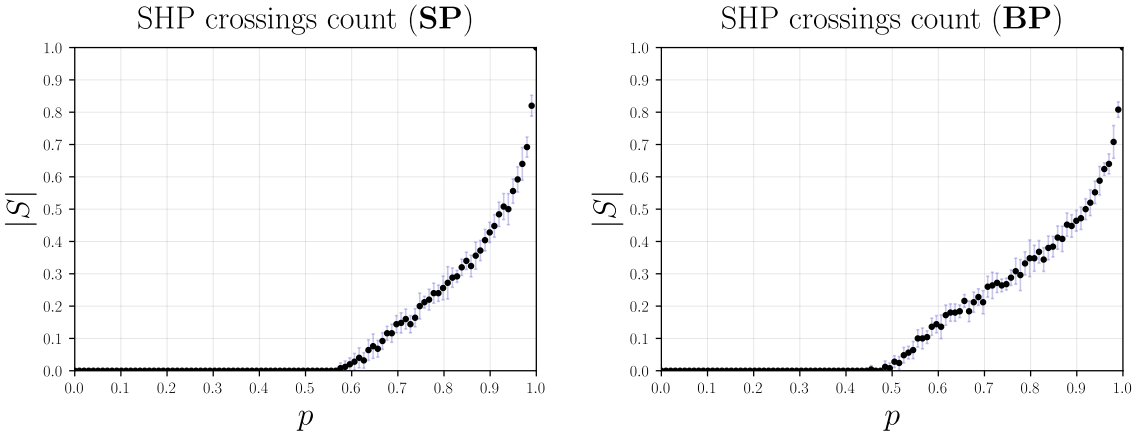


Figure 5.3: Numerical calculation of crossings $|S|$ found using SHP on $\mathbb{Z}^2$ for site percolation (left) and bond percolation (right). Computed for $L = 50$ statistically averaged over 10 repeats with standard deviations displayed.

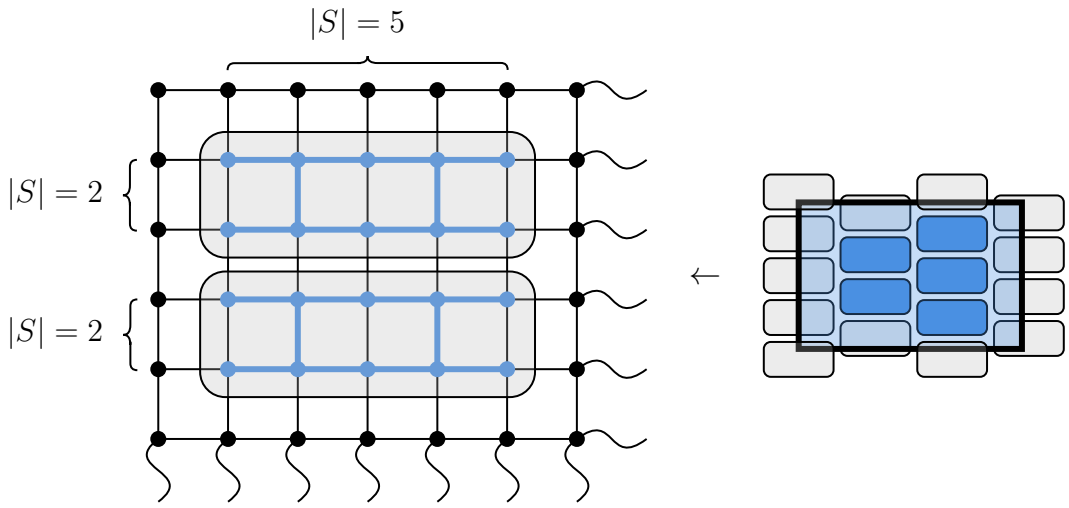
As expected, there are much fewer desirable crossing paths in practice, though they still birth from the classical critical value. These numerical calculations hence promote the argument for supercritcal sufficiency. Whilst we do not provide a size-dependent accuracy of how the sufficient condition for universality holds below the infinite case, 5.1 suggests a good approximation even at smaller scales.

These results can further be used to extract a necessary overhead, given that the number of crossings immediately defines the limits of the universal structures that can be extracted. For this development, instead of seeking $\mathbb{H}^2$ regions within $\mathbb{Z}^2$, we will look directly for a brickwork-compatible subgraph. This builds immediately from the method of tracing out $\mathbb{H}^2$, given the similarities in the two topologies. The component bricks are then promoted as a currency for quantifying computational capacity.

5.5 The method of overcompensation

Calculating a resource overhead to account for errors

The maximal number of discernible bricks $\mathfrak{b}(|S|)$ can be extracted by investigating the relationship between $|S|$ crossings on a $\mathbb{Z}^2$ lattice and its ability to construct the global brickwork structure. We propose an estimation for this quantity by averaging the relative number of crossings necessary to create a single brick. Below we outline the process of how these bricks are quantified from $\mathbb{Z}^2$ crossings, which we represent in a lattice form for clarity.



The right-hand diagram demonstrates how only complete bricks will contribute to the computational capacity, meaning that a flooring aspect is required in an estimation for their quantity.

$$\mathfrak{b}(|S|) = \left\lfloor \frac{1}{2} \left\lfloor \frac{|S|}{2} \right\rfloor \left\lfloor \frac{|S|}{5} \right\rfloor \right\rfloor \qquad \mathfrak{b}_0 := \mathfrak{b}\left(\left\lfloor \frac{L}{2} \right\rfloor \right) \qquad \varsigma := \frac{\mathfrak{b}}{\mathfrak{b}_0} \tag{5.7}$$

The maximal amount of bricks available to the original $\mathbb{Z}^2$ is defined as $\mathfrak{b}_0$. The proportion as compared to this original cluster state will then allow us to compute an overhead. We define this quantity to be $\varsigma := \frac{\mathfrak{b}}{\mathfrak{b}_0}$ and then consider our *overhead* as $\Omega := \varsigma^{-1}$.

Qualitatively, this is the value by which we must additionally scale our faulty lattice to almost surely ensure a computational capacity equivalent to an unfaulted implementation of the original $\mathbb{Z}^2$ cluster.

In *fig. 4,* we offer numerical estimations of the overhead. We have numerically implemented $\Gamma : p \longrightarrow |S|$, to instead provide the brick count in terms of success rate p, where $\mathfrak{b}(p) := \mathfrak{b}(\Gamma(p))$. The discontinuity in the *exact* points arises from the quantisation of bricks since they are only present in complete units. By neglecting the flooring aspect, we also superimpose *extended* results, to imitate the tendency expected in the thermodynamic limit.

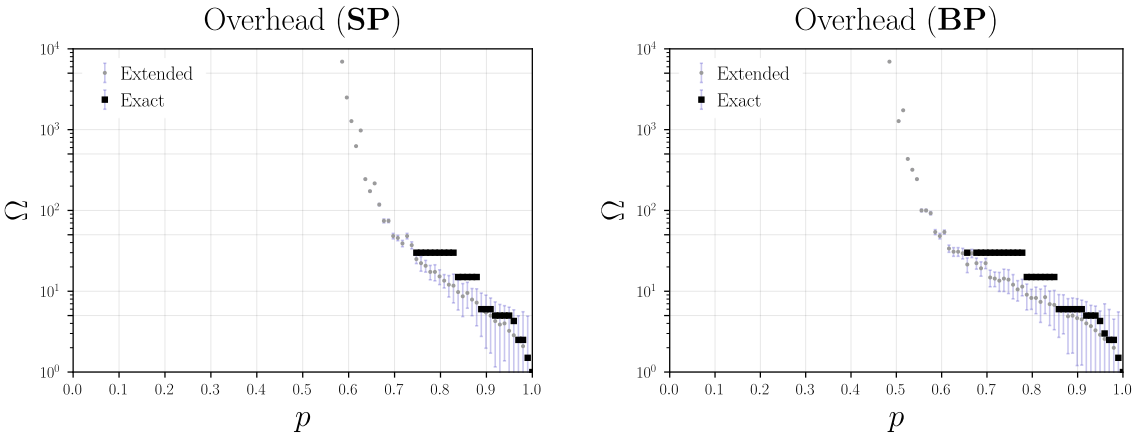


Figure 5.4: Numerical calculation of the overhead Ω on $\mathbb{Z}^2$ for site percolation (left) and bond percolation (right). *Exact* values obey the flooring aspect for quantising bricks, whilst *extended* disregards it so as to represent the general tendency in the thermodynamic limit. Computed for $L = 50$ statistically averaged over 10 repeats with standard deviations displayed.

It is shown that Ω asymptotes vertically at the percolation threshold, phenomena accounting for how only limitless additional resources will take a critically faulty cluster towards a usable state. This was precisely the behaviour predicted beforehand, where we expected possible universality towards p_c, even if they necessitate a large overcompensation in initial resources. In the subcritical regime, no amount of overcompensated scaling can achieve a universal state, so these values are suitably ill-defined.

It is difficult to obtain analytical results from the observed plots, given the fluctuations and dependencies on the crossing algorithm. Still, we can gain insights into the computational complexity.

The relations of 5.7 first imply $\Omega \propto |S|^{-2}$. In [7], the RHWW is found to detect roughly $\mathcal{O}(p)$ crossings away from the percolation threshold. Given its additional numerical proximity to the SHP method, we can say in general that $\Omega \propto \mathcal{O}(p^{\gamma})$ with $\gamma \sim -2$. This means that the overhead scales polynomially in the supercritical regime away from p_c, and so offers an efficient scaling for overcompensation.

What we have then shown is that for regular and equal-sided planar lattices, crossing probabilities directly coincide with the manifestation of an LCC. Given that their very presence is sufficient to develop a universal substructure, one can conclude that the phase transition not only delimits the subcritical regime as simulatable classically but almost surely guarantees the supercritical regime to be universal. To account for these non-critical errors reducing computational capacity, one can compensate with an efficient scaling overhead so as to allow for the universal capacity initially desired in the unfaulted scenario. Nevertheless, our language of efficiency here concerns a computational implementation, as opposed to a physical one. In practice, even polynomial compensation can prove more difficult on an experimental level. Might there exist better ways of growing these lattices in the first place?

6 Faults and fusion

"Have no fear of perfection, you'll never reach it." - *Salvador Dali*

Non-deterministic CZ gates in LOQC can be strategically improved on smaller scales. However, their implementation in larger systems remains constrained, requiring a coarse-grained model for better insights into percolating behaviour. This stems from fusion measurements, which are instead deployed to glue constituent micro-clusters together. The method further allows for three-dimensional resources, which are mandatory for practical fault-tolerant quantum computing. These tools were recently reformulated as a single model proposed in 2021 [3], an architecture that to our knowledge remains unexplored from the perspective of percolation methods and rigorous ZX treatments.

6.1 High-dimensional tolerance

Why fault-tolerance is enabled in 3-dimensions

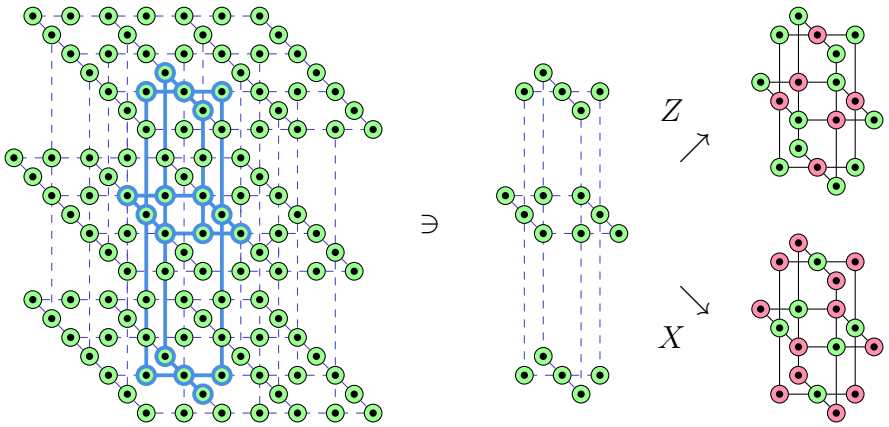
In addition to the errors arising in cluster state preparation, defects can occur during computation. Even the slightest environmental disturbances can lead to unwanted effects on qubits, which can destroy the desired computational process. In classical computation, redundancy is used to overcome these errors by creating multiple copies of the same pieces of information. This approach considers the majority output of clones, rather than relying on a single potentially faulted bit. Unfortunately, the *no-cloning theorem* prevents this scheme entirely from being realised in the quantum setting [13]. This is the statement that it is impossible to construct an identical and independent copy of an arbitrary unknown quantum state.

Instead, we can allocate bits towards a larger entangled state. Here, *k logical qubits* embed their state into a highly entangled collection of *n physical qubits*. How this particular assignment is made defines a particular *code*. One then uses a *syndrome* measure-

ment to detect corrupted qubits in the entangled system. Subsequent operations are then tailored towards restoring the logical piece of information. The general field of quantum error correction (QEC) seeks general methods of detecting and correcting defects without collapsing states. Whist it is not of interest to provide an extensive discussion here, a more developed introduction is available in [29].

Fault tolerance, is the additional requirement for physically viable quantum computing architecture. This is achieved when the physical error rate is reduced below a certain threshold, that through repeated applications of a permitting QEC scheme, can suppress logical errors to arbitrarily low levels [27]. When referring to a MBQC system that enables these properties, we will instead call this *fault-tolerant quantum computing* (FTQC).

Physically, resource states are given an additional dimension for entanglement to most naturally enable fault-tolerance. As part of the error diagnosis, *parity checks* consist of collections of entangled qubits that decipher whether *bit-flip* (undesired X operations) or *phase-flip* (undesired Z operations) have occurred. Inspired by the BCC symmetric FTQC lattice of [6], we illustrate a possible fault-tolerant cluster state below, highlighting the internal structure that plays the role of checks.



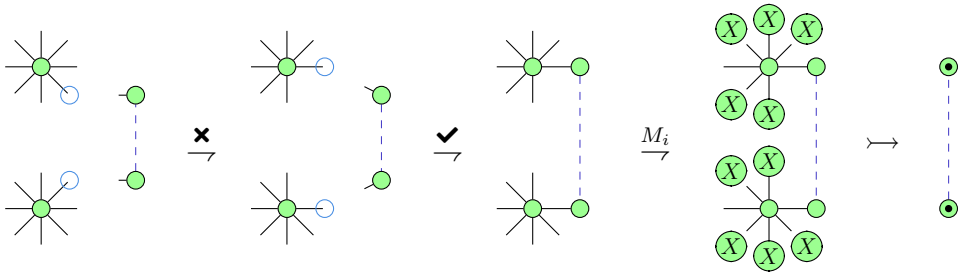
Whilst Z and X parity checks appear to take on the same form, their action differs depending on the layers in which they are embedded. It is suggested in [7] to explore the percolation results in the 3-dimensional case, so one might seek to explore the above topology as a minor of $\mathbb{Z}^3$. However, in a physical setting, these larger clusters are not

immediately implementable. It is therefore not reflective of their physical creation to consider these results with the same methods. LOQC being a more suitable candidate for large-scale FTQC, offers a solution by gluing smaller pieces together.

6.2 Strategies for the optical architect

The realistic approach for building large lattices

Repeat-until-success methods can boost CZ success rates from 50% to well above 99% [19]. To do so, micro-clusters are first created in the form of a central qubit attached to many potentially redundant neighbours, an initial process that can be achieved deterministically [26]. One can then repeatedly apply photonic CZ gates, until successfully entangling two auxiliary qubits. Failed CZ leads to the destruction of both target qubits, so the optimality of this method is entirely determined by the number of neighbouring qubits each. Once complete, any remaining neighbours are measured out, leaving only the central qubits behind. The process is outlined in ZX below.

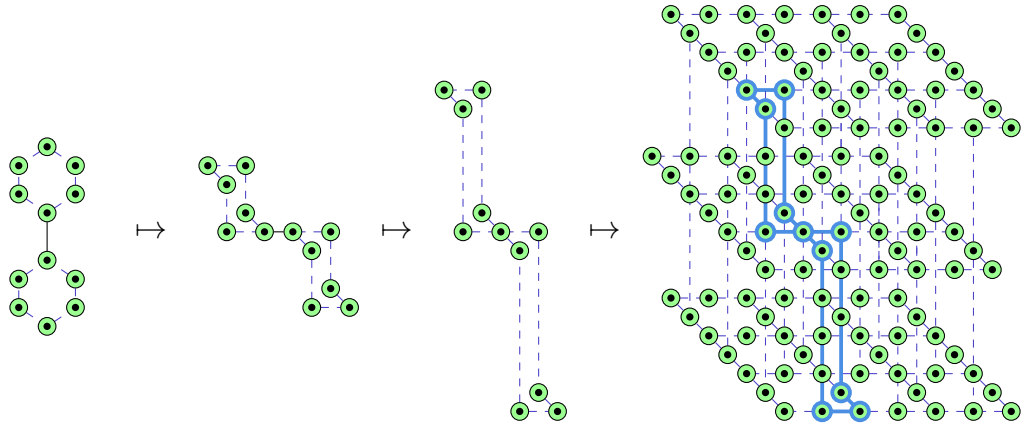


For the above process considering an 8-legged spider, which models 7 potential target qubits, our likelihood of success is already 99.22%. To ensure growth on average, only 2 additional legs are needed on the central qubit [26]. At these scales, bond percolation may seem like a redundant issue, but it is for larger systems where this process becomes unfeasible. This is where 2-qubit measurements known as fusions come into play.

In photonic systems, there exist two primary classes of fusions known as Type-I and Type-II. Both effectively implement Bell measurements, though differing in terms of their

physical realisation. Type-II results are more favourable to cluster state development because they are heralded and can be boosted using auxiliary qubits up to a 75% success rate [14]. It is less a matter of considering the defected CZ gates forming micro-clusters since they can be near-deterministically implemented, but rather a question of how the non-deterministic fusion gates succeed in bridging these smaller components together.

The construction of large-scale fault-tolerant resources using fusion measurements was recently bridged in a set of papers stemming from [3]. Fusion-based quantum computing (FBQC) is here proposed as a method to simultaneously account for the range of issues stated above. By considering micro-clusters structured strategically, their method facilitates the construction of an immediately fault-tolerant resource. For the same BCC architecture as above, the protocol suggested in [6], proposes the creation of 6-ring clusters, innately capable of QEC schemes, before fusing them selectively as shown.



Alternative models also prepare slightly larger micro-clusters of differing structures. The resulting system is then prone to defects mostly at the level of unsuccessful fusions between components, rather than internal to the micro-clusters themselves. However, given the time it takes to implement the above protocols, photonic loss can still occur within the components. Similarly, repeat-until-success methods might be implemented at worse rates, given experimental limitations or requirements for efficiency.

Original percolation models forget to capture these higher-order effects, occurring in 3-dimensional cluster state preparation. For this reason, we consider a renormalisation

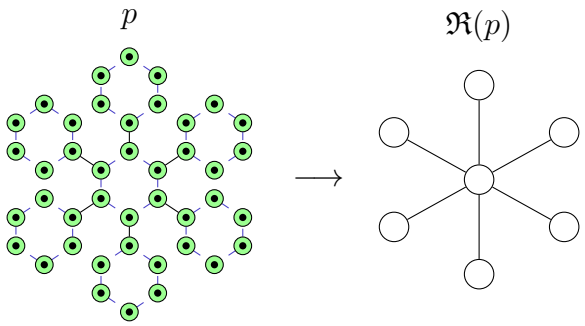
method for a toy-FBQC model. The aim is to demonstrate an alternative approach to the method of finding phases of fault-tolerant universality in FTQC, necessary for the eventual quantification of an overhead.

6.3 Network renormalisation

The strategy of coarse-graining a cluster state

We first distinguish the success rates of micro-cluster percolation and Type-II fusions as p and p_F respectively, since these are not likely to be the same. Fusion measurements always manifest a form of bond percolation at the global scale. On the contrary, depending on the errors that best model our particular setup, we either subject the micro-cluster to photonic loss on sites, or entanglement defects as bonds. We will not consider their simultaneous presence for clarity, but such results are easily found by extension.

In this way, we first handle a system subject to $1-p$ errors in either percolation types on the micro clusters but $1-p_F$ on the fusions between. We then *renormalise* the micro-clusters so as to effectively model a global system with a successful site rate of $\mathfrak{R}(p)$ and correctly implement bonds according to p_F.



The translation through the photonic loss model is labelled as $\mathfrak{R}_s$ and for CZ entanglement via $\mathfrak{R}_b$. We then propose a toy construction of 1000 small $\mathbb{Z}^2$ clusters with $L = 20$, modelled with success p for either form of percolation. These components are then subject to fusion measurements so as to relate them via global cubic connections $\mathbb{Z}^3$ of size

$L = 10$, occurring with success p_F. In the renormalised scheme, the entire micro-cluster is modelled as a single node subject to a success rate of $\mathfrak{R}(p)$. If the micro-cluster does not present an innate universal structure, then it is modelled as defective, and so removed from the global structure. That is, we apply a local crossing analysis to each of the components as a means of determining their coarse-grained effect.

For both SP and BP at the component level, our models are expected to display higher-order effects, not predictable by an immediate modelling of a standard cubic lattice. For comparison, *fig. 5.1* presents the standard combined percolated $\mathbb{Z}^3$ LCC growth, where p is the site success and p_F is for bonds.

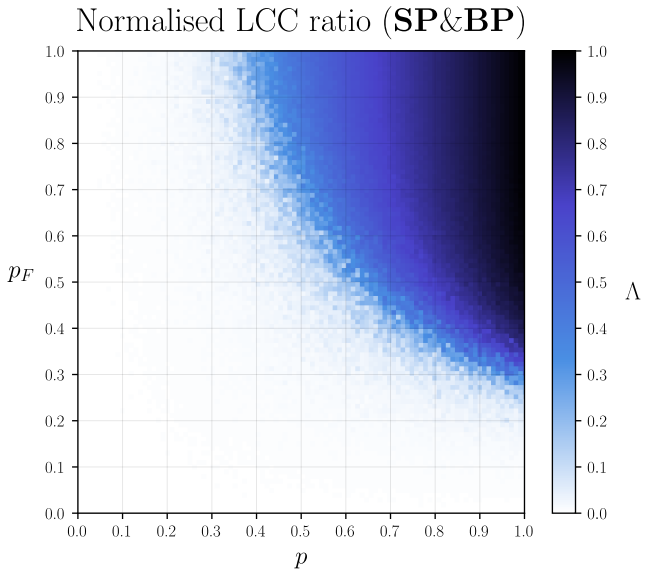


Figure 6.1: Numerical calculation of normalised LCC ratios Λ for the cubic lattice subject to combined site and bond percolation. Computed for $L = 30$ statistically averaged twice.

We again identify the limiting region as a separation from universality, which falls naturally from the arguments of sufficiency before. Indeed, if a unique spanning cluster is identified, given the symmetry of $\mathbb{Z}^3$, crossings are expected from all faces with guaranteed intersections of at least degree 3, since they are part of the same connected component. In this way, one could use an adapted algorithm to trace these out as the backbone for

another underlying universal resource, but will it be fault-tolerant?

Junctions of degree 3, which are guaranteed in both 2-dimensional and 3-dimensional crossing intersections, are sufficient for extracting the universal $\mathbb{H}^2$ or brickwork lattice. However, this is not necessarily enough for current fault-tolerant architectures like the BCC structure illustrated above, instead requiring crossing intersections of at least degree 4. Whilst the likelihood of this event is large in the supercritical regime, it requires a more detailed analysis from the perspective of its degree distribution.

The coarse-grained models sought to better represent the physical FBQC resource preparation are displayed in *fig. 5.2*.

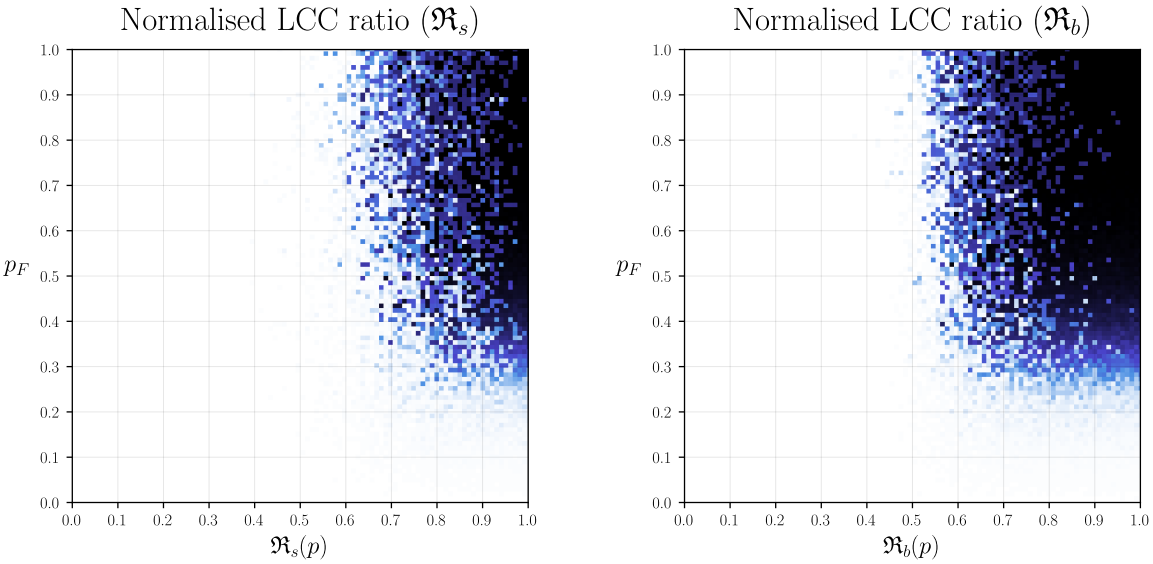


Figure 6.2: Numerical calculation of normalised LCC ratios Λ for a global cubic lattice whose coarse grained nodes represent $\mathbb{Z}^2$ micro-clusters. Fusion bond measurements succeed with p_F whilst micro-clusters are subject either to local site percolation (left) or bond percolation (right) with p. These represent photonic loss via $\mathfrak{R}_s(p)$ or CZ entanglement disruptions as $\mathfrak{R}_b(p)$, affecting the global site percolation accordingly. Computed for $L = 10$ at the cubic level, and $L = 20$ for 1000 micro-clusters with statistical averaging repeated 5 times.

These latter results present a tighter critical boundary, though the supercritical regime offers a more rapidly saturating growth than the non-renormalised case. With $\mathfrak{R}_b(p) =$

99.22% and $p_F = 75\%$, using the optimal case values mentioned before, a universal and very likely fault-tolerant resource safely lands in the supercritical regime. By mapping out the entire region of criticality, limits to which both types of errors must be overcome are then concisely identified. An extended three-dimensional crossing analysis with more sophisticated algorithms could then be used to discern the necessary overhead for such implementations.

6.4 Hypergraph calculus

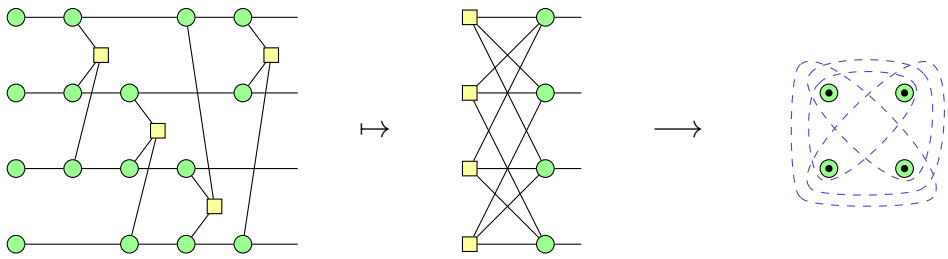
A potential generalisation for more robust states

Erroneous resource preparation can also be studied through the lens of hypergraph states. These are already well defined in the graph state literature as an extension to MBQC resources [30], though their bridge to percolating cluster state preparation and the ZX formalism remains untouched. The *hyperedge* allows for simultaneous connections to many individual nodes. Its practicalities for representing entanglement have already been investigated given that the hyperedge offers a natural description for multipartite systems [20]. Hypergraph states are defined with k indexing the range of qubits to which the multi-controlled CZ gate applies:

$$|H\rangle = \prod_{(v_i,\ldots,v_k)\in E} C^k Z_{v_i,\ldots,v_k} \, |+\rangle^{\otimes |V|} \tag{6.1}$$

Theoretical work has developed message-parsing techniques to discern percolation thresholds in hypergraphs, demonstrating that they exhibit significantly lower critical values than traditional graphs [5]. Independently, hypergraph states have been suggested as deployable resources for *continuous variable* (CV) MBQC [21]. This model of quantum computing instead considers non-discrete Hilbert spaces $\mathfrak{H}$, a framework that can be exploited with specific photonic modes, and displays a variety of advantages over the standard framework [15]. Given their more robust percolation properties, hypergraph resources could prove to be promising candidates against preparation defects.

As an extended representation of the ZH formalism more tailored towards hypergraph states, we suggest the generalised N-cardinality *hyper-hadamard edge*. The role of multi-qubit CZ gates is then directly offered through a hypergraph translation. One can see this more fluently from a bipartite representation, where H-box nodes are taken to represent these hyperedges.



The ZH-calculus carries its own specific rewriting rules [2], whilst simultaneously reflecting the underlying lattice subject to percolation methods. Future work might seek to translate these ZH rules into the above representation, a transparency that would promote its relationship towards percolation on hypergraph states. Rewriting rules could also be deployed within the faulty lattices, to help convert regions into more easily identifiable universal structures.

Beyond MBQC, hypergraph structures are omnipresent in multipartite entangled systems. It would be of interest to see if the use of percolation methods and graph robustness provide options for quantifying entanglement, non-locality or decoherence in open quantum systems.

7 Conclusion

> "Beauty is the first test; there is no permanent place in the world for ugly mathematics."
>
> - *Godfrey Harold Hardy*

By mediating the benefits of ZX-Calculus and percolation theory, we have shown that a number of new insights regarding faulty cluster state preparation are obtainable. The ZX formalism was shown initially to allow for an organic transition towards erroneous clusters. Here, universal properties devised from the diagrammatic language were exhibited though phase transitions in percolation theory.

We argued that the supercritical regime is sufficient for a universal subsystem to appear on the $\mathbb{Z}^2$ symmetric lattice at the thermodynamic limit. The argument extends naturally to structures obeying similar requirements, allowing for a richer interpretation of error regimes. Numerical results are shown to agree with these predictions, though it is stressed that fluctuations in accuracy are expected for finite systems. These methods, and their relation to the brickwork lattice explored through ZX, further enabled an estimation for the physical overhead required for physical cluster state resources.

Whilst [7] motivated investigations into 3-dimensional cases, we found that although this model is readily treated with similar methodologies as before - though needing additional degree distribution analysis to guarantee fault tolerance, it is not reflective of modern cluster state preparation techniques. Instead, we consider fusion-based models as the most realistic preparation process. Here, coarse gaining was applied to the global structure of the cluster states, an approach that was demonstrated on a toy model. Subsequent numerical simulations reveal higher-order effects which shift the original critical thresholds from standard 3-dimensional percolation. Hypergraph states were then promoted as a more robust candidate against percolating errors, a path that is left for further work.

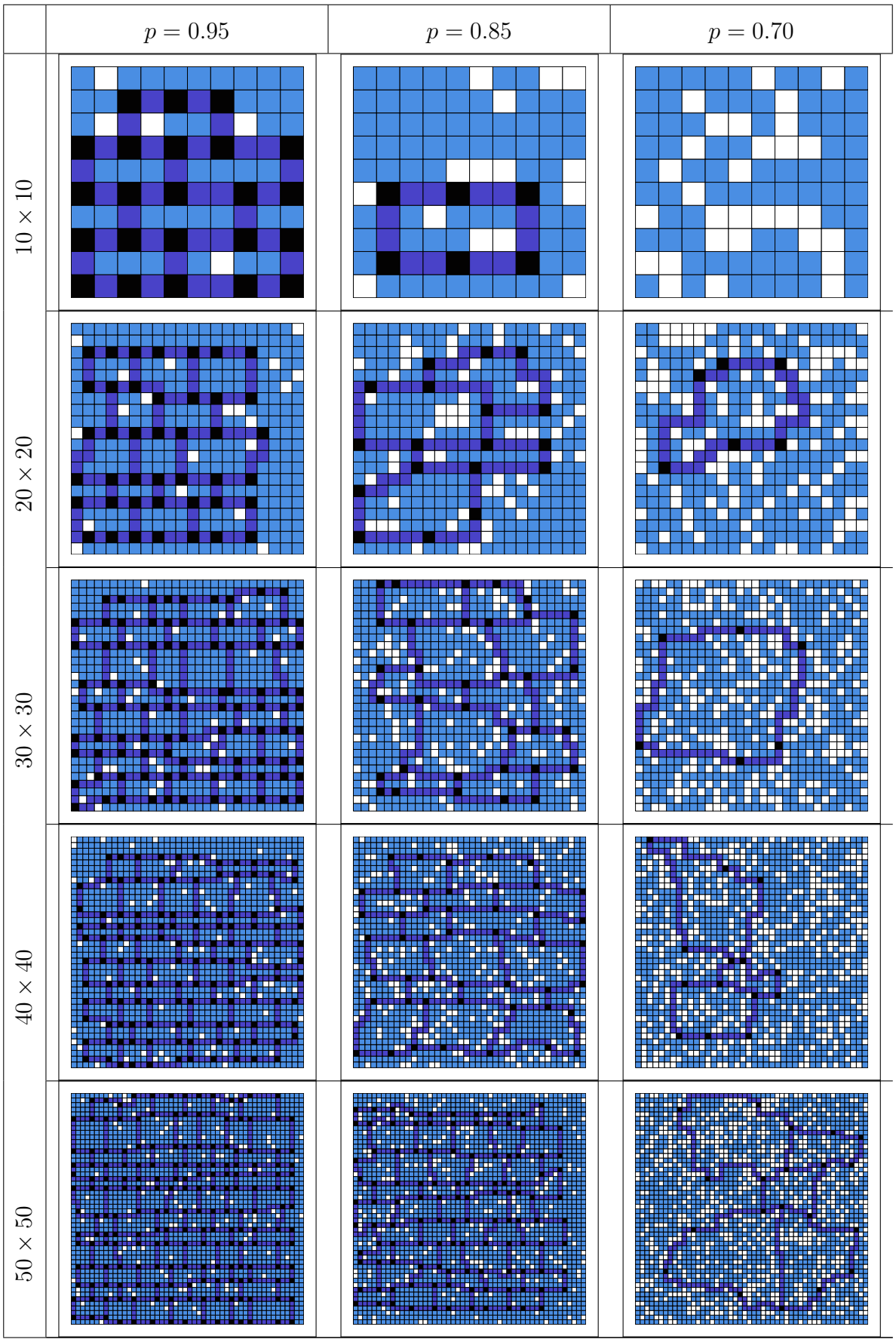
The duality of the ZX-calculus and percolation theory, alongside the methods devised throughout our discussions, may not be limited to applications in faulty measurement-based quantum computing. As suggested in the hypergraph discussion, large entangled structures entail a wide range of quantum-phenomena, and it would be of interest to promote the combined methods across new areas of quantum research. Alternatively, we could exercise these strategies for other percolated tensor networks, given that these structures are frequently prevalent in other disciplines.

7.1 Further Work

Instead of relating computational units to the brickwork lattice, one could attempt to quantify efficiency from a more theoretical approach, to better adapt this measure to non-regular lattice structures. In this way, overhead predictions could be offered for arbitrary systems, an achievement that would be extremely practical to real-life cluster state preparation.

It would also be of interest to harness the methods of renormalisation beyond toy models and see their effects on realistic fusion-based systems. Likewise, a 3-dimensional tracing algorithm could be devised to then identify the universal subsections within. Whilst it is likely that this subgraph is also fault-tolerant, it is suspected that a deeper investigation into degree distributions on percolation models would be required to locate the sufficient region for extracting fault-tolerant universal states.

Finally, the list of methods proposed could be generalised to hypergraph states, as motivated by their natural representation in the ZX-formalism as well as their robustness against percolating errors. This could further motivate rewriting methods specific to assisting the identification of universal substructures, a strategy that might first begin with more standard graph states.


$p = 0.95$
$p = 0.85$
$p = 0.70$
10×10
20×20
30×30
40×40
50×50

A SHP algorithm

```
def graph_to_matrix(graph, size):
    """
    Converts a graph into a matrix representation.
    Inputs:
        - graph: the graph
        - size: size of the matrix
    Outputs:
        - matrix: matrix representation of the graph
    """
    matrix = np.zeros((size, size))
    for (i, j) in graph.nodes:
        matrix[i][j] = 1
    return matrix

def count_neighbour_types(neighbours, neighbour_types):
    """
    Counts the number of neighbours of a certain type.
    Inputs:
        - neighbours: list of neighbour nodes
        - neighbour_types: types of neighbours to count
    Outputs:
        - total: count of neighbours of specified types
    """
    total = 0
    for neighbour in neighbour_types:
        total += neighbours.count(neighbour)
    return total

def get_square_neighbours(i, j, matrix):
    """
    Identifies a square region around a point in the matrix.
    Inputs:
        - i: row index
        - j: column index
        - matrix: the matrix
    Outputs:
        - square_neighbours: list of square members
    """
    size = len(matrix)
    shifts = [(0, 0), (0, -1), (-1, 0), (-1, -1)]
    square_positions = [(0, 0), (0, 1), (1, 0), (1, 1)]
    for sx, sy in shifts:
        square_neighbours = []
        for dx, dy in square_positions:
            nx = i + dx + sx
            ny = j + dy + sy
            if 0 <= nx < size and 0 <= ny < size:
```

```
                if matrix[nx][ny] == 2 or matrix[nx][ny] == 3:
                    square_neighbours.append(matrix[nx][ny])
        if len(square_neighbours) == 4:
            break
    return square_neighbours

def is_flower_structure(i, j, matrix):
    """
    Checks if a point is part of a flower structure in the matrix.
    Inputs:
        - i: row index
        - j: column index
        - matrix: the matrix
    Outputs:
        - is_flower: boolean indicating if the point is part of a
          flower structure
    """
    size = len(matrix)
    directions = [(1, 0), (0, 1), (-1, 0), (0, -1)]
    direction_indices = [0, 1, 2, 3]
    values = np.zeros(4)
    for direction_index in direction_indices:
        rotated_directions = np.roll(directions, -direction_index, axis
            =0)
        shifts = rotated_directions[0:2]
        for dx, dy in shifts:
            nx = i + dx
            ny = j + dy
            if 0 <= nx < size and 0 <= ny < size:
                if matrix[nx][ny] == 4 or matrix[nx][ny] == 3:
                    values[direction_index] += 1
    return tuple(values).count(2) == 1

def von_neumann_neighbourhood(matrix, neighbour_types):
    """
    Computes the Von Neumann neighbourhood for a given matrix and
       neighbour cell types.
    Inputs:
        - matrix: the matrix
        - neighbour_types: types of neighbours to count
    Outputs:
        - VNN_matrix: matrix of neighbour counts
    """
    VNN_matrix = np.zeros((len(matrix), len(matrix)))
    for i in range(len(matrix)):
        for j in range(len(matrix)):
            neighbour_count = count_neighbour_types(
                get_von_neumann_neighbours(i, j, matrix),
                neighbour_types)
            VNN_matrix[i][j] = neighbour_count
    return VNN_matrix

def branch_shortest_path(graph, matrix, direction):
    """
```

```
    Branches the shortest path horizontally or vertically in the matrix.
    Inputs:
        - graph: the graph
        - matrix: the matrix
        - direction: 'H' for horizontal, 'V' for vertical
    Outputs:
        - matrix_with_paths: matrix with paths
        - all_paths: list of all paths found
    """
    if direction == 'H':
        flipper = 1
    elif direction == 'V':
        flipper = -1
    else:
        return "ERROR"

    matrix_with_paths = matrix.copy()
    all_paths = []

    for shift in range(len(matrix)):
        VNN_matrix = von_neumann_neighbourhood(matrix_with_paths, [2])
        path_candidates = []

        source = (shift, 0)[::flipper]
        target = (shift, len(matrix) - 1)[::flipper]

        if source in graph.nodes and target in graph.nodes:
            overlap = False
            try:
                path = nx.shortest_path(graph, source, target)
            except Exception:
                continue
            for (i, j) in path:
                if VNN_matrix[i][j] != 0:
                    overlap = True
                    break
            if not overlap:
                path_candidates.append(path)

        if path_candidates:
            best_path = min(path_candidates, key=len)
            all_paths.append(best_path)
            for (i, j) in best_path:
                matrix_with_paths[i][j] = 2

    return matrix_with_paths, all_paths

def combine_paths(graph, matrix):
    """
    Combines horizontal and vertical paths in the matrix.
    Inputs:
        - graph: the graph
        - matrix: the matrix
```

```
    Outputs:
        - combined_matrix: matrix with combined paths
        - vertical_paths: list of vertical paths
    """
    horizontal_paths, _ = branch_shortest_path(graph, matrix, 'H')
    vertical_paths, vpaths = branch_shortest_path(graph, matrix, 'V')
    combined_matrix = horizontal_paths + vertical_paths
    combined_matrix[combined_matrix == 2] = 1
    combined_matrix[combined_matrix == 3] = 2
    combined_matrix[combined_matrix == 4] = 3
    return combined_matrix, vpaths

def trim_isolated_elements(matrix):
    """
    Trims isolated elements in the matrix based on their neighbourhood.
    Inputs:
        - matrix: the matrix
    Outputs:
        - trimmed_matrix: trimmed matrix
    """
    size = len(matrix)
    trimmed_matrix = matrix.copy()
    while True:
        VNN_matrix = von_neumann_neighbourhood(matrix, [2, 3])
        for i in range(size):
            for j in range(size):
                if VNN_matrix[i, j] == 1:
                    matrix[i][j] = 1
        if np.array_equal(trimmed_matrix, matrix):
            break
        else:
            trimmed_matrix = matrix.copy()
    return matrix

def trim_bridges(matrix):
    """
    Trims bridges in the matrix.
    Inputs:
        - matrix: the matrix
    Outputs:
        - trimmed_matrix: trimmed matrix
    """
    size = len(matrix)
    trimmed_matrix = matrix.copy()
    while True:
        VNN_matrix = von_neumann_neighbourhood(matrix, [2, 3])
        for i in range(size):
            for j in range(size):
                if matrix[i][j] == 2 and VNN_matrix[i][j] == 1:
                    matrix[i][j] = 1
        if np.array_equal(trimmed_matrix, matrix):
            break
        else:
            trimmed_matrix = matrix.copy()
```

```
    return matrix

def remove_bridges(graph, matrix):
    """
    Removes bridges from the matrix after crossing paths.
    Inputs:
        - graph: the graph
        - matrix: the matrix
    Outputs:
        - result_matrix: matrix with bridges removed
    """
    crossed_matrix, vertical_paths = combine_paths(graph, matrix)
    trimmed_matrix = trim_isolated_elements(crossed_matrix)
    size = len(crossed_matrix)
    new_paths = np.zeros((size, size))

    switch = 1
    for path in vertical_paths:
        switch *= -1
        for step in path[:-1]:
            stamp = 2 if switch == 1 else 0
            if trimmed_matrix[step[0]][step[1]] == 3:
                next_step = path[path.index(step) + 1]
                if trimmed_matrix[next_step[0]][next_step[1]] != 3:
                    stamp = 0
            new_paths[step[0], step[1]] = stamp

    for i in range(size):
        for j in range(size):
            if new_paths[i][j] == 2 and crossed_matrix[i][j] != 2:
                crossed_matrix[i][j] = 0
    result_matrix = crossed_matrix
    return result_matrix

def remove_errors(matrix):
    """
    Removes erroneous paths based on square patterns in the matrix.
    Inputs:
        - matrix: the matrix
    Outputs:
        - trimmed_matrix: trimmed matrix
    """
    trimmed_matrix = matrix.copy()
    while True:
        for i in range(len(matrix)):
            for j in range(len(matrix)):
                if matrix[i][j] == 2:
                    square = get_square_neighbours(i, j, matrix)
                    if tuple(square).count(2) == 1:
                        matrix[i][j] = 1
        if np.array_equal(trimmed_matrix, matrix):
            break
        else:
            trimmed_matrix = matrix.copy()
```

```
    return matrix

def highlight_intersections(matrix):
    """
    Colors intersections in the matrix.
    Inputs:
        - matrix: the matrix
    Outputs:
        - highlighted_matrix: matrix with colored intersections
    """
    for i in range(len(matrix)):
        for j in range(len(matrix)):
            if matrix[i][j] == 2:
                neighbours = get_von_neumann_neighbours(i, j, matrix)
                if neighbours.count(2) > 2:
                    matrix[i][j] = 3
    return matrix

def spread_contamination(matrix):
    """
    Spreads contamination from intersections in the matrix.
    Inputs:
        - matrix: the matrix
    Outputs:
        - contaminated_matrix: contaminated matrix
    """
    size = len(matrix)
    contaminated_matrix = matrix.copy()
    while True:
        for i in range(size):
            for j in range(size):
                if matrix[i][j] == 2:
                    if get_von_neumann_neighbours(i, j, matrix).count
                        (3) == 1:
                        contaminated_matrix[i][j] = 3
        if np.array_equal(contaminated_matrix, matrix):
            break
        else:
            matrix = contaminated_matrix.copy()
    return matrix

def fix_contaminated_intersections(matrix):
    """
    Fixes contaminated intersections in the matrix.
    Inputs:
        - matrix: the matrix
    Outputs:
        - fixed_matrix: fixed matrix
    """
    size = len(matrix)
    fixed_matrix = matrix.copy()
    while True:
        for i in range(size):
            for j in range(size):
```

```
                if matrix[i][j] == 3:
                    square = get_square_neighbours(i, j, matrix)
                    if tuple(square).count(3) == 1:
                        fixed_matrix[i][j] = 2
        if np.array_equal(fixed_matrix, matrix):
            break
        else:
            matrix = fixed_matrix.copy()
    return matrix

def fix_outer_intersections(matrix):
    """
    Fixes outer intersections in the matrix.
    Inputs:
        - matrix: the matrix
    Outputs:
        - fixed_matrix: fixed matrix
    """
    size = len(matrix)
    fixed_matrix = matrix.copy()
    while True:
        for i in range(size):
            for j in range(size):
                if matrix[i][j] == 2:
                    if is_flower_structure(i, j, matrix):
                        fixed_matrix[i][j] = 3
        if np.array_equal(fixed_matrix, matrix):
            break
        else:
            matrix = fixed_matrix.copy()
    return matrix

def simplify_matrix(matrix):
    """
    Chains multiple reduction steps to simplify the matrix.
    Inputs:
        - matrix: the matrix
    Outputs:
        - simplified_matrix: reduced matrix
    """
    matrix = fix_outer_intersections(matrix)
    matrix = fix_contaminated_intersections(matrix)
    matrix = spread_contamination(matrix)
    return matrix

def iterative_trim(matrix):
    """
    Repeatedly trims the matrix using chain reduction.
    Inputs:
        - matrix: the matrix
    Outputs:
        - trimmed_matrix: trimmed matrix
    """
    while True:
```

```
365         previous_matrix = matrix.copy()
366         matrix = simplify_matrix(matrix)
367         matrix = remove_errors(matrix)
368         matrix = trim_bridges(matrix)
369         if np.array_equal(previous_matrix, matrix):
370             break
371     return matrix
372 
373 def mark_paths(graph, matrix):
374     """
375     Sets fire (marks) to specific paths in the matrix based on
        intersections.
376     Inputs:
377         - graph: the graph
378         - matrix: the matrix
379     Outputs:
380         - marked_matrix: matrix with paths marked
381     """
382     result_matrix = remove_bridges(graph, matrix)
383     result_matrix = highlight_intersections(result_matrix)
384     result_matrix = spread_contamination(result_matrix)
385     result_matrix = iterative_trim(result_matrix)
386     return result_matrix
387 
388 def cement_paths(matrix):
389     """
390     Cements (fixes) the paths in the matrix after marking.
391     Inputs:
392         - matrix: the matrix
393     Outputs:
394         - cemented_matrix: cemented matrix
395     """
396     cemented_matrix = matrix.copy()
397     while True:
398         previous_matrix = matrix.copy()
399         for i in range(len(matrix)):
400             for j in range(len(matrix)):
401                 if matrix[i][j] == 3:
402                     if get_von_neumann_neighbours(i, j, matrix).count
                        (3) == 1:
403                         cemented_matrix[i][j] = 2
404         if np.array_equal(previous_matrix, cemented_matrix):
405             break
406         else:
407             matrix = cemented_matrix.copy()
408     return cemented_matrix
409 
410 def highlight_squares(matrix):
411     """
412     Colors squares in the matrix.
413     Inputs:
414         - matrix: the matrix
415     Outputs:
416         - highlighted_matrix: matrix with colored squares
```

```
    """
    size = len(matrix)
    highlighted_matrix = matrix.copy()
    for i in range(size):
        for j in range(size):
            if matrix[i][j] == 2:
                square = get_square_neighbours(i, j, matrix)
                if tuple(square).count(2) == 1:
                    highlighted_matrix[i][j] = 1
    return highlighted_matrix

def full_path_extraction(graph, matrix):
    """
    Fully extracts the essential paths and features from the matrix.
    Inputs:
        - graph: the graph
        - matrix: the matrix
    Outputs:
        - extracted_matrix: extracted matrix
    """
    matrix = cement_paths(matrix)
    matrix = mark_paths(graph, matrix)
    matrix = highlight_squares(matrix)
    matrix = iterative_trim(matrix)
    return matrix
```

Bibliography

[1] Samson Abramsky and Bob Coecke. 'A categorical semantics of quantum protocols'. In: (2004). DOI: 10.48550/ARXIV.QUANT-PH/0402130. URL: https://arxiv.org/abs/quant-ph/0402130.

[2] Miriam Backens and Aleks Kissinger. 'ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-linearity'. In: (2018). DOI: 10.48550/ARXIV.1805.02175. URL: https://arxiv.org/abs/1805.02175.

[3] Sara Bartolucci et al. 'Fusion-based quantum computation'. In: (2021). DOI: 10.48550/ARXIV.2101.09310. URL: https://arxiv.org/abs/2101.09310.

[4] Jacob D. Biamonte, Stephen R. Clark and Dieter Jaksch. 'Categorical Tensor Network States'. In: (2010). DOI: 10.48550/ARXIV.1012.0531. URL: https://arxiv.org/abs/1012.0531.

[5] Ginestra Bianconi and Sergey N. Dorogovtsev. 'The theory of percolation on hypergraphs'. In: (2023). DOI: 10.48550/ARXIV.2305.12297. URL: https://arxiv.org/abs/2305.12297.

[6] Hector Bombin et al. *Unifying flavors of fault tolerance with the ZX calculus*. 2023. DOI: 10.48550/ARXIV.2303.08829. URL: https://arxiv.org/abs/2303.08829.

[7] Daniel E. Browne et al. 'Phase transition of computational power in the resource states for one-way quantum computation'. In: (2007). DOI: 10.48550/ARXIV.0709.1729. URL: https://arxiv.org/abs/0709.1729.

[8] John Cardy. 'Critical Percolation in Finite Geometries'. In: (1991). DOI: 10.48550/ARXIV.HEP-TH/9111026. URL: https://arxiv.org/abs/hep-th/9111026.

[9] Kim Christensen. *Percolation Theory*. Imperial College London, Oct. 2002. URL: https://web.mit.edu/ceder/publications/Percolation.pdf.

[10] Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, Mar. 2017. ISBN: 9781316219317. DOI: 10.1017/9781316219317. URL: http://dx.doi.org/10.1017/9781316219317.

[11] Christopher M. Dawson and Michael A. Nielsen. 'The Solovay-Kitaev algorithm'. In: (2005). DOI: 10.48550/ARXIV.QUANT-PH/0505030. URL: https://arxiv.org/abs/quant-ph/0505030.

[12] Artur K. Ekert. 'Quantum cryptography based on Bell's theorem'. In: *Physical Review Letters* 67.6 (Aug. 1991), pp. 661–663. ISSN: 0031-9007. DOI: 10.1103/physrevlett.67.661. URL: http://dx.doi.org/10.1103/PhysRevLett.67.661.

[13] Heng Fan et al. 'Quantum Cloning Machines and the Applications'. In: (2013). DOI: 10.48550/ARXIV.1301.2956. URL: https://arxiv.org/abs/1301.2956.

[14] Mercedes Gimeno-Segovia et al. 'From three-photon GHZ states to ballistic universal quantum computation'. In: (2014). DOI: 10.48550/ARXIV.1410.3720. URL: https://arxiv.org/abs/1410.3720.

[15] Mile Gu et al. 'Quantum Computing with Continuous-Variable Clusters'. In: (2009). DOI: 10.48550/ARXIV.0903.3233. URL: https://arxiv.org/abs/0903.3233.

[16] K. Kieling, T. Rudolph and J. Eisert. 'Percolation, renormalization, and quantum computing with non-deterministic gates'. In: (2006). DOI: 10.48550/ARXIV.QUANT-PH/0611140. URL: https://arxiv.org/abs/quant-ph/0611140.

[17] E. Knill, R. Laflamme and G. J. Milburn. 'A scheme for efficient quantum computation with linear optics'. In: *Nature* 409.6816 (Jan. 2001), pp. 46–52. ISSN: 1476-4687. DOI: 10.1038/35051009. URL: http://dx.doi.org/10.1038/35051009.

[18] Laurin Köhler-Schindler and Vincent Tassion. 'Crossing probabilities for planar percolation'. In: (2020). DOI: 10.48550/ARXIV.2011.04618. URL: https://arxiv.org/abs/2011.04618.

[19] Yuan Liang Lim, Almut Beige and Leong Chuan Kwek. 'Repeat-Until-Success Quantum Computing'. In: (2004). DOI: 10.48550/ARXIV.QUANT-PH/0408043. URL: https://arxiv.org/abs/quant-ph/0408043.

[20] Xiangyi Meng et al. 'Percolation Theories for Quantum Networks'. In: (2023). DOI: 10.48550/ARXIV.2310.18420. URL: https://arxiv.org/abs/2310.18420.

[21] Darren W. Moore. 'Quantum Hypergraph States in Continuous Variables'. In: (2019). DOI: 10.48550/ARXIV.1909.03871. URL: https://arxiv.org/abs/1909.03871.

[22] M. Van den Nest. 'Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond'. In: (2008). DOI: 10.48550/ARXIV.0811.0898. URL: https://arxiv.org/abs/0811.0898.

[23] M. Van den Nest et al. 'Fundamentals of universality in one-way quantum computation'. In: (2007). DOI: 10.48550/ARXIV.QUANT-PH/0702116. URL: https://arxiv.org/abs/quant-ph/0702116.

[24] Maarten Van den Nest et al. 'Universal resources for measurement-based quantum computation'. In: (2006). DOI: 10.48550/ARXIV.QUANT-PH/0604010. URL: https://arxiv.org/abs/quant-ph/0604010.

[25] Michael A. Nielsen. 'Optical quantum computation using cluster states'. In: (2004). DOI: 10.48550/ARXIV.QUANT-PH/0402005. URL: https://arxiv.org/abs/quant-ph/0402005.

[26] Michael A. Nielsen. 'Optical quantum computation using cluster states'. In: (2004). DOI: 10.48550/ARXIV.QUANT-PH/0402005. URL: https://arxiv.org/abs/quant-ph/0402005.

[27] Alexandru Paler and Simon J. Devitt. 'An introduction to Fault-tolerant Quantum Computing'. In: (2015). DOI: 10.48550/ARXIV.1508.03695. URL: https://arxiv.org/abs/1508.03695.

[28] Miklós Rédei and Stephen Jeffrey Summers. 'Quantum probability theory'. In: *Studies in History and Philosophy of Science Part B: Studies in History and Philosophy of Modern Physics* 38.2 (June 2007), pp. 390–417. ISSN: 1355-2198. DOI: 10.1016/j.shpsb.2006.05.006. URL: http://dx.doi.org/10.1016/j.shpsb.2006.05.006.

[29] Joschka Roffe. 'Quantum Error Correction: An Introductory Guide'. In: (2019). DOI: 10.48550/ARXIV.1907.11157. URL: https://arxiv.org/abs/1907.11157.

[30] M. Rossi et al. 'Quantum Hypergraph States'. In: (2012). DOI: 10.48550/ARXIV.1211.5554. URL: https://arxiv.org/abs/1211.5554.

[31] Dietrich Stauffer and Ammon Aharony. *Introduction To Percolation Theory*. Taylor Francis, Dec. 2018. ISBN: 9781482272376. DOI: 10.1201/9781315274386. URL: http://dx.doi.org/10.1201/9781315274386.

[32] Guifre Vidal. 'Entanglement monotones'. In: (1998). DOI: 10.48550/ARXIV.QUANT-PH/9807077. URL: https://arxiv.org/abs/quant-ph/9807077.

[33] Quanlong Wang. 'Completeness of the ZX-calculus'. In: (2022). DOI: 10.48550/ARXIV.2209.14894. URL: https://arxiv.org/abs/2209.14894.

[34] Tzu-Chieh Wei. 'Quantum spin models for measurement-based quantum computation'. In: *Advances in Physics: X* 3.1 (Jan. 2018), p. 1461026. ISSN: 2374-6149.

DOI: 10.1080/23746149.2018.1461026. URL: http://dx.doi.org/10.1080/23746149.2018.1461026.

[35] John van de Wetering. 'ZX-calculus for the working quantum computer scientist'. In: (2020). DOI: 10.48550/ARXIV.2012.13966. URL: https://arxiv.org/abs/2012.13966.

[36] Sabine Wölk and Otfried Gühne. 'Characterizing the width of entanglement'. In: (2015). DOI: 10.48550/ARXIV.1507.07226. URL: https://arxiv.org/abs/1507.07226.