Quantum circuit extraction

Resource efficient translation of deterministic measurement patterns to quantum circuits

Marek Grzesiuk

St Edmund Hall

A thesis submitted for the degree of

Master of Science in Advanced Computer Science

Trinity 2023

Abstract

Quantum computing is a new and exciting revolution in the way humans perform computation. It has many possible applications; however, as physical quantum computers have started to appear relatively recently, there are currently many different ways to build a quantum computer and some of them use different computational models. Furthermore due to engineering challenges, currently accessible devices are limited in number and quality of qubits, which are the main computational resource of a quantum computer. As such, only limited-size programs can be executed on currently available machines, which stresses the need for better optimisation methods. Some of the methods currently use a graphical language called *ZX calculus* to apply rewrite rules and optimise circuits that way; however, those rewrite rules are restricted, as extracting a quantum circuit out of an arbitrary *ZX diagram* is computationally hard, and efficient methods are known only for specific families of diagrams.

In this work, we provide a method for extracting quantum circuits out of a family of ZX diagrams, which is significantly larger than the families of diagrams for which efficient extraction methods are known. We do so by providing a new translation method between two computational models called the *quantum circuit* model and *measurement-based quantum computing* which is closely related to ZX diagrams. Furthermore, we provide further insight into resource trade-off in this translation by giving theoretical guarantees on performance of this algorithm for the number of qubits and the number of two-qubit gates in the extracted circuit, as well as evaluate our method empirically.

Contents

1 Introduction

Quantum computing refers to performing computation by transforming quantum systems using rules of quantum mechanics. Its use has been suggested in late 20^{th} century by Feynman as a means to perform physical simulations that would be intractable on classical computers [15]. Further research into quantum computing resulted in famous procedures like Shor's prime factorization algorithm [38] that provides an exponential speed up over currently known classical methods and will revolutionise modern cryptography once sufficiently large quantum computers have been built, or Grover's search [17] which remarkably allows for searching through unstructured list in $O(\sqrt{n})$. The potential of quantum computers has also been noticed by large industrial entities like McKinsey & Company that expect significant added value due to a quantum revolution by as early as mid 2030s [19].

That being said, currently available quantum computers, called *noisy intermediate-scale quantum* (NISQ) era devices are far from reaching those expectations. Some of their main issues are short life span of qubits, the main resources of quantum computation that are analogous to the role of bits in classical computing, relatively high error rates on various operations, specifically, multi-qubit ones, and small number of qubits available in quantum computers.

There are two primary ways to alleviate those issues that are being investigated in parallel, one being, developing better hardware devices. A possible approach of doing so, is investigating different models of computation, for example *measurement-based quantum computers* are theorized to possibly be easier to physically implement [32, 42] than the currently dominant *quantum circuit* model. Another approach is to further improve methods of optimising quantum circuits so that a larger family of them can

be executed on currently available devices. One prominent approach of doing that is using *ZX calculus*, a powerful graph language that allows us to reason about quantum computation in a much simpler way than the "standard" matrix calculus. The primary issue with methods that use ZX calculus for optimization, like that of Duncan et al. [13] is that only very specific rewrite rules, that preserve certain properties of the graph, may be applied. This is because extracting a quantum circuit out of a ZX diagram, which are the diagrams on which ZX calculus computations are performed, is computationally hard in general [4]. As such, current rewrite rules have to preserve various, so called, *flows*, whose existence in graphs allows efficient extraction.

In this work we study translation between a large family of *measurement patterns* (programs in measurement-based quantum computing) and quantum circuits, and due to significant closeness of measurement patterns and ZX diagrams, also extraction methods for a much larger family of ZX diagrams than contemporary flow-based methods allow for. This novel approach also significantly reduces, in some cases by over than 75%, the number of qubits in the extracted circuit as compared to other approaches that do not rely on existence of any type of flow. Moreover, a version of our algorithm that outputs circuits with classically controlled gates significantly reduces the number of two-qubit gates in some cases even outperforming more specialised algorithms that require existence of *generalized flow*. Finally, our work also furthers our understanding of resource trade-off in the measurement-based quantum computing model and the quantum circuit model by analysing theoretical properties of our algorithm and deriving various bounds on the properties of the extracted circuits.

1.1 Contributions

In this work we introduce *partial causal flow*, which extends the concept of *causal flow* and using that extension we provide a novel, more resource efficient extraction method for any pattern that is strongly deterministic, significantly extending the family of patterns for which extraction methods, that do not require using as many qubits as there are in the measurement pattern, are known. Furthermore, we provide theoretical

guarantees on both the number of qubits produced by our method and the number of two-qubit gates in multiple different settings. Finally, we evaluate our method empirically against other algorithms and investigate the quality of the derived bounds, as well as conjecturing further ways of improving methods presented here and of applying them to the problem of quantum circuit optimization.

1.2 Outline

We first provide necessary background to follow this thesis in chapter 2, then we present the bulk of theoretical results in chapter 3. There we discuss our approach and provide theoretical guarantees on the performance of our methods. Later, in chapter 4 we benchmark our approach against another methods and evaluate the quality of the bounds shown in this work. Finally, in chapter 5 we discuss applications of the methods presented here, possible future approaches of extending and improving those methods and reflect on the project as a whole.

2 Background

In this chapter we provide background necessary to follow this thesis and understand its wider context. Firstly, in Section 2.1, we establish some notation which is later used in this thesis to reason about graphs and mention two specific notions from graph theory which play a significant role in deriving some of this thesis' theoretical results. Then in Section 2.2, we give an introduction to quantum computing and present two main models of quantum computation: *quantum circuit model* and *measurement-based quantum computation*. Following that, in Section 2.3, we give an overview of ZX calculus, a powerful graphical language used to reason about quantum computation. Finally, in Section 2.4, we contextualise this thesis by reviewing the related literature.

2.1 Graph theory

Suppose that G is a graph, we denote by $V(G)$ the set of all vertices of G and by $E(G)$ the set of all edges of G. Let $N_G(u)$ (or if G is obvious from the context $N(u)$) be the neighbourhood of vertex $u \in V(G)$. For directed graphs we will denote by $N_{G,in}(u)$ and by $N_{G,out}(u)$ ($N_{in}(u), N_{out}(u)$) the set of nodes from which there is an edge to and from u respectively. We will also denote by $2N_G(u) = \{v \mid dist(u,v) \leq 2\}$ the two-hop neighbourhood of node u (with all the additional notation analogous to neighbourhood), where $dist(u,v)$ is the distance between nodes u and v in G. We denote the odd neighbourhood of graph G as $Odd(A) = \{u \mid |N(u) \cap A| \equiv 1 \mod 2\}$, that is, all nodes that have an odd number of edges connected to set $A \subset V(G)$. Finally, let $deg(u) = |N(u)|$ be the degree of node u, and in-deg(u) $= |N_{in}(u)|$, out-deg(u) $= |N_{out}(u)|$ be the in and out degree of node u respectively.

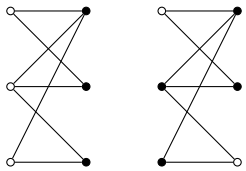


Figure 2.1: On the left: example of a maximum independent set (white nodes are part of the set) and on the right: example of a maximum distance-3 independent set (white notes are part of the set)

2.1.1 Independent sets

We start with defining *independent set* and *maximum independent set* problem, which is a strongly NP-hard problem and it's corresponding decision problem is NP-complete, as defined by Pemmaraju et al. [33].

Definition 2.1.1. *An **Independent set** of a graph G is a subset S of V(G) such that no two nodes in S are neighbours*

Definition 2.1.2. *A **maximum independent set** is the independent set S such that $|S|$ is the largest*

In this work, we will make use of a related concept called *distance-3 independent set*, which is an independent set but rather than nodes not being allowed to be neighbours, they are not allowed to be within distance of two from each other.

Definition 2.1.3. *An **distance-3 independent set** is a independent set S such that $\forall u, v \in S, u = v \vee dist(u, v) > 2$*

Unsurprisingly, computing maximum distance-3 independent set is hard and the related decision problem has also been shown to be NP-complete [20].

2.1.2 Cutwidth

Another NP-complete problem that will appear in this work is the problem of finding the *cutwidth* of a graph. We define a *cut* of size k is a arrangement of nodes in a line, such that a vertical line moved from left to right crosses at most k edges. Then the cutwidth is the minimal size of a cut, or more formally, as stated by Chung [8]:

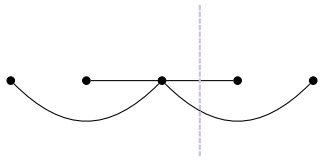


Figure 2.2: Example of minimal cutwidth partition of a star graph G, of order 5, with cutwidth(G)=2

Definition 2.1.4. *Let $\pi : V(G) \to \mathbb{Z}$ be an one-to-one mapping defining some ordering of edges in G, then:*

$$f_\pi(G) = \max_i(|\{(u,v) \in E(G) \mid \pi(u) \leq i < \pi(v)\}|)$$

Then cutwidth is cutwidth$(G) = \min_\pi(f_\pi(G))$

2.2 Quantum computing

The term *quantum computing* refers to performing computation via evolving quantum systems according to the rules of quantum mechanics. While there are many possible ways of implementing quantum systems (and subsequently evolving them) like, for example, using superconductors [43], photons [1] or ion traps [18], one can encapsulate their behaviour into a single mathematical framework called *quantum information*.

2.2.1 Qubits

In general, quantum systems are described as complex Hilbert spaces $\mathcal{H}$, and states of those quantum systems are represented by normalised vectors in $\mathcal{H}$. Specifically, two dimensional quantum system are called *qubits* and their states are described by $|\psi\rangle \in \mathbb{C}^2$, where the symbol $|\psi\rangle$ is called *ket*, and refers to the vector ψ. This symbol is a part of notation that is commonly used in quantum computing called *bra-ket* or Dirac notation. The other part of the name of the notation, *bra* refers to a symbol $\langle\psi|$ which is another way of writing $\psi^\dagger$ (an adjoint of ψ, also called conjugate transpose of ψ) and, when combining the two together (to obtain inner product between two vectors), we use the following notation $\langle\psi|\,|\phi\rangle = \langle\psi|\phi\rangle$.

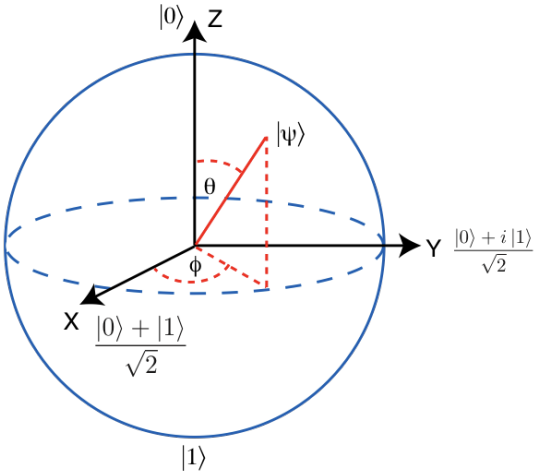


Figure 2.3: Bloch sphere with an arbitrary state $|\psi\rangle = \cos(\frac{\theta}{2})\,|0\rangle + e^{i\phi}\sin(\frac{\theta}{2})\,|1\rangle$ (each state can be written in this form, up to some global phase) plotted. *Source:* Qiskit textbook [34]

As already mentioned, the states of quantum systems are represented by normalised vectors in some Hilbert space $\mathcal{H}$. While any quantum state can be written in such a way, there is no one-to-one relationship between quantum states and vectors. This is because for any two quantum states $|\psi\rangle$ and $|\psi'\rangle$ for which there exist an α such that $|\psi\rangle = e^{i\alpha}\,|\psi'\rangle$, we say that $|\psi\rangle$ and $|\psi'\rangle$ are *equal up to a global phase* and we are unable to distinguish those two states.

In this thesis, we will focus specifically on systems that can be built from qubits, which are the most popular building blocks of quantum computing. States of a qubit have a neat representation as points lying on the *Bloch sphere* which can be seen in figure 2.3. Since states of qubits are represented by normalised vectors in the complex vector space $\mathbb{C}^2$, one can write any state $|\phi\rangle$ as a combination of bases of $\mathbb{C}^2$, for example, using the *Z basis* (also called the *computational basis*)

$$|\phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$$

for some $\alpha, \beta \in \mathbb{C}$ where $\|\alpha\|^2 + \|\beta\|^2 = 1$ and basis states defined as:

$$|0\rangle := \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle := \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

In the Bloch sphere representation, any two states forming an orthonormal basis set (ONBs) are represented as antipodal points on the sphere. When it comes to the Z basis specifically, $|0\rangle$ and $|1\rangle$ also lie on the Z axis. The two other ONBs that lie on the X and Y axis are called the *X basis* defined as $\{|+\rangle, |-\rangle\}$ and the *Y basis* defined as $\{|i\rangle, |-i\rangle\}$ where:

$$|+\rangle := \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad |-\rangle := \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

$$|i\rangle := \tfrac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle) \quad |-i\rangle := \tfrac{1}{\sqrt{2}}(|0\rangle - i\,|1\rangle)$$

2.2.2 Evolution of quantum systems

Computation in quantum information is done via evolving quantum systems according to rules of quantum mechanics, specifically, according to one of the most influential equations of 20^{th} century, the Schrödinger equation [37].

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = \hat{H} |\psi(t)\rangle$$

In the case of quantum information we can assume that the Hamiltonian $\hat{H}$ is constant. This allows to simplify that equation into the following rule: "The evolution of a closed quantum system is described by a unitary transformation" [31]. That is,

$$|\psi(t)\rangle = U |\psi(0)\rangle$$

for some unitary $U \in \mathbb{C}^{d\times d}$ where d is the dimension of the quantum system and $UU^\dagger = U^\dagger U = I$. Furthermore, there is an one-to-one correspondence between the descriptions using Hamlitonians and unitaries which means that any unitary can be used to evolve a quantum system. Moreover, since unitaries represent rotations, evolving quantum states is equivalent to rotating their state in the Hilbert space. In the case of qubits specifically, they relate to rotations around Bloch sphere. For example rotations

by an arbitrary angle α around the X, Y and Z axis, respectively are given by the following transformations:

$$R_x(\alpha) := \begin{pmatrix} \cos(\frac{\alpha}{2}) & -i\sin(\frac{\alpha}{2}) \\ -i\sin(\frac{\alpha}{2}) & \cos(\frac{\alpha}{2}) \end{pmatrix}$$

$$R_y(\alpha) := \begin{pmatrix} \cos(\frac{\alpha}{2}) & -\sin(\frac{\alpha}{2}) \\ \sin(\frac{\alpha}{2}) & \cos(\frac{\alpha}{2}) \end{pmatrix}$$

$$R_z(\alpha) := \begin{pmatrix} e^{-i\frac{\alpha}{2}} & 0 \\ 0 & e^{i\frac{\alpha}{2}} \end{pmatrix}$$

Setting $\alpha = \pi$ gives us three matrices called Pauli matrices (whose eigenvectors form the X, Y and Z basis) which up to a global phase are equal to:

$$X = R_x(\pi) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$Y = R_y(\pi) = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

$$Z = R_z(\alpha) = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

One additional famous gate that is going to be referenced throughout this thesis is called the Hadamard gate H and can be used to swap between X and Z axis - that is, $R_z(\alpha) = HR_x(\alpha)H$. It is a rotation by π around the axis defined by $\frac{\hat{z}+\hat{x}}{\sqrt{2}}$ [31], that is the vector

$$\frac{\hat{z}+\hat{x}}{\sqrt{2}} = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{pmatrix}$$

on the Bloch sphere and is given by the following matrix:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

We finish this overview of single qubit gates by stating a convenient fact which is that, it can be shown that for any unitary and two non parallel axis $\hat{n}, \hat{m}$ there exist real numbers $\alpha, \beta, \gamma, \delta$ such that $U = e^{i\alpha} R_{\hat{n}}(\beta) R_{\hat{m}}(\gamma) R_{\hat{n}}(\delta)$ [31]. Hence, any single qubit transformations can be expressed, up to some global phase, as a composition of rotations around, for example, the Z and X axis. As the name may suggest it, this specific fact plays a significant role in the ZX calculus.

2.2.3 Composite states and entanglement

A qubit, which is also called quantum bit, is the quantum counterpart of a bit in classical computing and much like a bit it is the basic resource of a quantum computer. As such, usually multiple qubits are required to perform interesting computations. A quantum system that is composed of multiple smaller subsystems is called a composite system and is mathematically described as a Kronecker product (also called *tensor product*) of the subsystems.

Mathematically, the tensor product of two vectors $\mathbf{u} \in \mathbb{C}^{d_u}$, $\mathbf{v} \in \mathbb{C}^{d_v}$ is computed as follows:

$$\mathbf{u} \otimes \mathbf{v} = \begin{pmatrix} u_1 \mathbf{v} \\ u_2 \mathbf{v} \\ \vdots \\ u_{d_u} \mathbf{v} \end{pmatrix} = \begin{pmatrix} u_1 v_1 \\ \vdots \\ u_1 v_{d_v} \\ \vdots \\ u_{d_u} v_1 \\ \vdots \\ u_{d_u} v_{d_v} \end{pmatrix}$$

A composite system of two states, represented by the Hilbert spaces $\mathcal{H}_a$ and $\mathcal{H}_b$, is represented by a Hilbert space $\mathcal{H}_a \otimes \mathcal{H}_b$, which contains all vectors $\mathbf{u} \otimes \mathbf{v}$ such that $\mathbf{u} \in \mathcal{H}_a, \mathbf{v} \in \mathcal{H}_b$. Furthermore, given two ONB bases $\{u_i\}, \{v_j\}$ of $\mathcal{H}_a, \mathcal{H}_b$ respectively, then the set $\{u_i \otimes v_j\}$ forms an ONB for $\mathcal{H}_a \otimes \mathcal{H}_b$. Hence, for example the computational basis for a quantum system consisting of two qubits is given by the set $\{|0\rangle \otimes |0\rangle, |1\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |1\rangle\}$.

A natural question to ask is if every vector $u \in \mathbb{C}^{2^n}$ can be decomposed into a tensor product of n other vectors $u_i \in \mathbb{C}^2$. This is not the case and it is easy to check that for example $|\psi\rangle = \frac{|0\rangle\otimes|0\rangle+|1\rangle\otimes|1\rangle}{\sqrt{2}}$ cannot be decomposed into any two vectors $|\psi_a\rangle, |\psi_b\rangle$ such that $|\psi\rangle = |\psi_a\rangle \otimes |\psi_b\rangle$. States like this, which cannot be decomposed into a tensor product of other states, are called *entangled* and states that can be decomposed are called *product states*.

Entangling gates are unitaries U such that there exist a product state $|\psi\rangle \otimes |\phi\rangle$ which after being transformed by U is entangled. Examples of such gates are controlled-NOT (or CNOT) and controlled-Z (or CZ) which are defined as:

$$CNOT = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$CZ = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes Z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

These two gates are also examples of controlled gates, that is gates that act on more than one qubit and apply a certain transformation when control qubit is in the state $|1\rangle$, for example $CNOT|11\rangle = |10\rangle$ or $CZ|1\rangle \otimes |+\rangle = CZ\frac{|10\rangle+|11\rangle}{\sqrt{2}} = \frac{|10\rangle-|11\rangle}{\sqrt{2}} = |1\rangle \otimes |-\rangle$.

2.2.4 Measurements

Initially, one can think that since qubits can be in any state, as long as it is normalized, then it is possible to store an infinite amount of information in one qubit. This is not exactly the case: while qubit can be in any normalized state, accessing information about that state is not straightforward. In order to retrieve information about a qubit, one has to *measure* it. A basic measurement in quantum computing is defined by an ONB

$\{|m\rangle\}$, where m is the result of the measurement. Specifically, Z basis measurement of a qubit can result in either 0 or 1 ($|0\rangle$ or $|1\rangle$).

Since qubits can be in the *superposition* of $|0\rangle$ and $|1\rangle$, quantum computing is inherently non deterministic and the probability of obtaining outcome m when measuring a state $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ using ONB $\{\psi_m\}$ is given by the Born rule and is equal to

$$P(m|\,|\psi\rangle) = \|\langle\psi_m|\psi\rangle\|^2 = \langle\psi_m|\psi\rangle\,\langle\psi|\psi_m\rangle$$

For example, measuring state $|\psi\rangle$ in the Z basis would give outcome 0 with probability:

$$Prob(0|\,|\psi\rangle) = \|\langle 0|\psi\rangle\|^2 = \|\alpha\|^2$$

Furthermore, when a qubit is measured and the outcome is m, no matter what the state of that qubit was before the measurement, the qubit after the measurement is in the state $|\psi_m\rangle$. Born rule also provides an explanation for previously stated conditions such as requiring the state vectors to be normalised (so that the probability of each of the outcomes would add up to 1) or vectors being equal up to phase. Indeed, given the vector $|\psi'\rangle = e^{i\theta}\,|\psi\rangle$, if we measure $|\psi'\rangle$ in an ONB with $|\psi\rangle$ (which always exists thanks to the Gram–Schmidt process) then the probability of obtaining measuring $|\psi\rangle$ would be $Prob(|\psi\rangle\,|\,|\psi'\rangle) = \|\langle\psi|\psi'\rangle\|^2 = \|e^{i\theta}\|^2 = 1 = Prob(|\psi\rangle\,|\,|\psi\rangle)$.

2.2.5 Quantum circuits

Similarly to classical computing, there are multiple ways of modeling quantum computation, such as the quantum circuit model which is the most popular one and on which most literature in the field focuses. Furthermore, one of the most prominent approaches in developing quantum computing hardware, superconducting quantum computers use this model, which made quantum circuits even more popular language to represent quantum computations.

Quantum circuits, similarly to classical logic circuits, are read from left to right. They consist of two building blocks: wires, which represent qubits in the circuit and gates, which represent operations on the wires (qubits). Gates are defined by the unitary

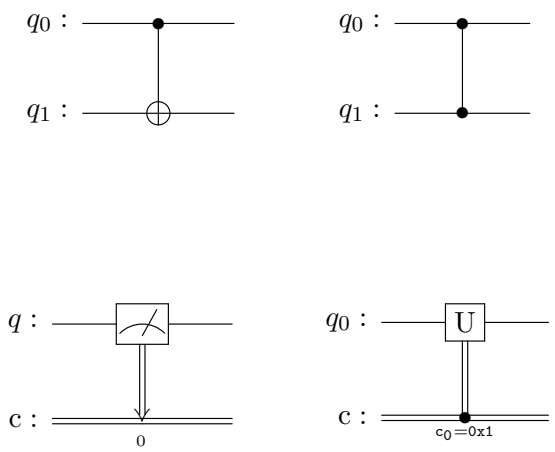


Figure 2.4: Top-left: graphical representation of CNOT (or CX) gate with q_0 being a control qubit and q_1 the target qubit, top-right: CZ gate (this gate is symmetric so either qubit can be considered a control or target), bottom-left: measurement in z basis and bottom-right: a single qubit gate U controlled classically by bit c

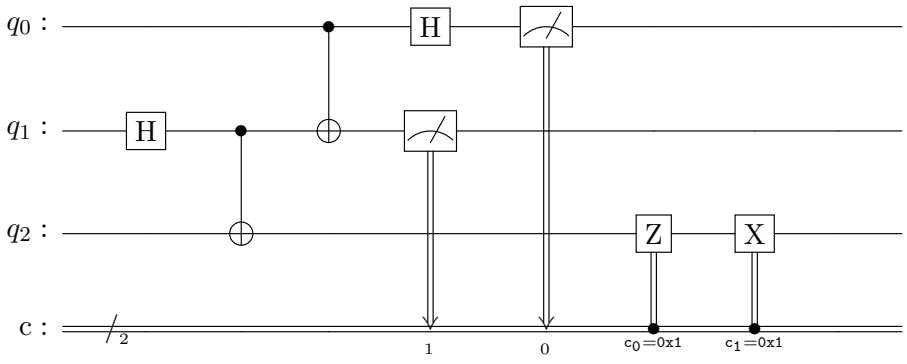


Figure 2.5: Circuit performing quantum teleportation, that is if state of qubit q_0 is $|\psi\rangle$ then after execution of the circuit q_2 is in the state $|\psi\rangle$. This is called teleportation, as the entangling of qubits q_1 and q_2 can be done before the circuit execution and therefore q_0 and q_2 can be arbitrary far apart when the "teleportation" occurs

transformations they represent. Figure 2.4 shows a graphical representation of the most commonly used gates in this thesis, and figure 2.5 shows an example of a circuit used for quantum teleportation. By convention, if the initial state of the qubit is not specified, it is assumed that the qubit is initialised in the state $|0\rangle$.

"Fully quantum" gates, like CZ or CX (where control is a qubit not a bit) are typically faster than classically controlled gates and for some hardware implementations of quantum computers, it may be more desirable to have as little classical control as possible. Luckily, classically controlled gates can be turned into quantum controlled gates by the use of *deferred measurement principle* which states that any measurement can be moved to the very end of the circuit without an effect on the result of the computation. When the measurements are moved, any classically controlled gates are simply changed to quantum controlled gates.

This method however requires slight caution in cases where the measurement is performed using a different basis to the computational basis. Firstly, it is worth noting that the measurement operation does so in the Z basis, as such any measurement in a different basis requires first applying gates that change the targeted basis to the Z basis. For example if one wants to measure in the X basis, one has to first apply a Hadamard gate and only then measure. When one wants to apply the measurement principle in this case, one must first apply change-of-basis, then controlled unitaries, and finally the measurement can be made. Examples of applications of the deferred measurement principle for measurements in the Z basis and the basis X can be seen in figure 2.6 and figure 2.7 respectively.

2.2.6 Measurement-based quantum computation

A vastly different approach to modeling quantum computation than quantum circuits is taken by measurement-based quantum computation (MBQC) models, which are driven by measuring qubits in a highly entangled resource state (multiple qubits entangled together in some specific way). Some examples of concrete measurement-based quantum computing models are teleportation based approaches first introduced by Gottesman et

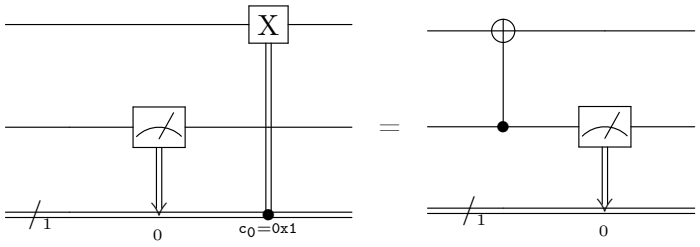


Figure 2.6: Example of deferred measurement principle with z basis measurement

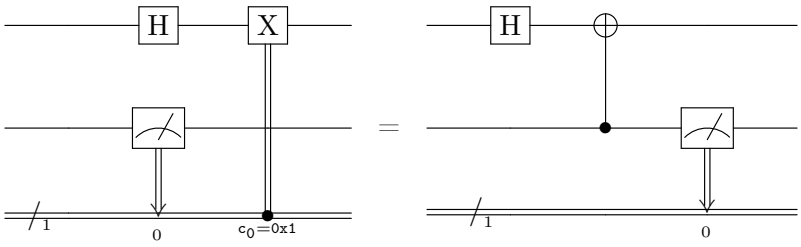


Figure 2.7: example of deferred measurement principle with x basis measurement

al [16] and the one-way model introduced by Raussendorf et al [35]. In this work we will focus on the latter and whenever we mention MBQC we will refer to one-way model.

Computation in the one-way model can be described by *measurement calculus* [12], in which there are 5 commands:

- N_i - prepare qubit i in the state $|+\rangle$
- E_{ij} - apply CZ gate between qubits i and j, this is the entanglement operation
- M_i^α - measure qubit i in the basis $\{|+_\alpha\rangle = \frac{|0\rangle+e^{i\alpha}|1\rangle}{\sqrt{2}}, |-_\alpha\rangle = \frac{|0\rangle-e^{i\alpha}|1\rangle}{\sqrt{2}}\}$
- X_i^s apply Pauli X correction to qubit i if the measurement result of qubit s is 1
- Z_i^s apply Pauli Z correction to qubit i if the measurement result of qubit s is 1

These commands are later used to create *measurement patterns* which describe the instructions that are to be executed on the quantum computer. The general idea is that commands N_i and E_{ij} prepare a cluster state which is later measured in some order by $M_i^{\alpha_i}$ and based on the result of those measurements some corrections (either X or Z) are applied. We provide formal descriptions of measurement pattern and some related concepts as defined by Danos et al. [10]

Definition 2.2.1. *A **measurement pattern** $\mathcal{P}$ is defined by a finite set of qubits V, set of input qubits $I \subset V$, set of output qubits $O \subset V$ (possibly overlapping) and a finite sequence of commands S acting on qubits in V.*

Remark 2.2.1.1. *We will use slight abuse of notation and say that a command C is in the measurement pattern $\mathcal{P}$, or $C \in \mathcal{P}$ when referring to command C being part of the sequence of commands S that is part of the measurement pattern $\mathcal{P}$*

Definition 2.2.2. *A measurement pattern is said to be **runnable** if the following holds:*

- *no command depends on the outcome of measurement that has not yet happened*
- *no command acts on a measured qubit*
- *no command, except of preparation command, acts on non prepared qubits from the set $V \setminus I$*

- *a qubit i is measured if and only if it is not an output ($i \notin O$)*

Remark 2.2.2.1. *In this work we will assume that all patterns are runnable.*

We make this assumption as any pattern that is not runnable has no physical meaning and by definition it is impossible to implement.

Definition 2.2.3. *An **open graph state** (or open graph) of a measurement pattern $\mathcal{P}$ is a tuple (G, I, O) where I and O are input and output sets of the measurement pattern and G is an undirected graph such that $V(G) = V$ and $(u, v) \in E(G) \iff E_{u,v} \in P$. That is each vertex of the graph relates to a qubit in the pattern and each edge in the graph relates to an entangle operation in the measurement pattern.*

In some work measurement patterns are also extended to include measurement planes, that is they include a function $\lambda : O^c \to \{(X, Y), (X, Z), (Y, Z)\}$. Then a qubit i is measured in $\lambda(i)$ plane - that is, it uses the following basis [6]:

- if $\lambda(i) = (X, Y)$, measure using $\{\frac{|0\rangle + e^{i\alpha}|1\rangle}{\sqrt{2}}, \frac{|0\rangle - e^{i\alpha}|1\rangle}{\sqrt{2}}\}$
- if $\lambda(i) = (X, Z)$, measure using $\{\cos(\frac{\alpha}{2})|0\rangle + \sin(\frac{\alpha}{2})|1\rangle, \sin(\frac{\alpha}{2})|0\rangle - \cos(\frac{\alpha}{2})|1\rangle\}$
- if $\lambda(i) = (Y, Z)$, measure using $\{\cos(\frac{\alpha}{2})|0\rangle + i\sin(\frac{\alpha}{2})|1\rangle, \sin(\frac{\alpha}{2})|0\rangle - i\cos(\frac{\alpha}{2})|1\rangle\}$

In this work we will focus only on the original formulation of the measurement pattern in which all measurements are done in (X, Y) basis.

Even though quantum circuit model is quite different from the one-way model, it can be shown that they are equivalent. Raussendorf et al [35] give a method of translating quantum circuits to measurement patterns in which arbitrary single qubit rotations can be implemented with a measurement pattern consisting of five qubits, and CNOT gate can be implemented with four-qubits-large measurement pattern. Since arbitrary single qubit rotations and CNOT gates are sufficient to implement any unitary exactly, one can use this method and compose resulting patterns to produce a measurement pattern that performs desired transformation. An arbitrary measurement pattern can also be implemented by a quantum circuit as shown by Broadbent et al [5]. This is rather unsurprising as any command in the measurement pattern can be implemented in s

quantum circuit, hence given a circuit with number of wires equal to the number of qubits in the pattern one can execute each of the commands on the quantum circuit producing the desired transformation.

While it is possible to transition back and forth between a quantum circuit model and measurement patterns doing so is quite expensive. It can be seen that each gate in the quantum circuit introduces multiple new qubits in the measurement pattern and since translating back to quantum circuit requires using as many qubits as the measurement pattern, one introduces large overhead in the number of qubits. That being said, some optimization methods, like that of Broadbent et al [5] use similar translation methods to reduce the depth of the circuit (which directly correlates with the time required for execution of the computation) in exchange for larger number of qubits needed to execute a circuit.

We end our introduction of MBQC by providing an example of a measurement pattern, let $V = \{1, 2, 3, 4\}$, $I = \{1, 2\}$ and $O = \{1, 4\}$, then the sequence of commands that implements a CNOT is:

$$\mathcal{P} = X_4^3 Z_4^2 Z_1^2 M_3^0 M_2^0 E_{13} E_{23} E_{34} N_3 N_4$$

Patterns are read from right to left, and in this case, firstly, qubits 4 and 3 are prepared (note that qubits 1 and 2 are provided as inputs and as such they do not require initialisation, they are expected to be provided in some state), then pairs of qubits (1, 3), (2, 3) and (3, 4) are entangled together. After that qubits 2 and 3 are measured and then if measurement of qubit 1 resulted in the outcome 1 ($|-\rangle$), the Z gate is applied to qubit 1 and the Z gate is applied to qubit 4, and similarly if measurement of qubit 3 is 1 then the X gate is applied to qubit 4. The open graph of measurement pattern $\mathcal{P}$ can be seen on Figure 2.8.

2.2.7 Determinism in measurement-based quantum computing

As computation in MBQC is driven by measurements, it is inherently non deterministic, that is why,w when discussing measurement patterns one sometimes refers to *branches*

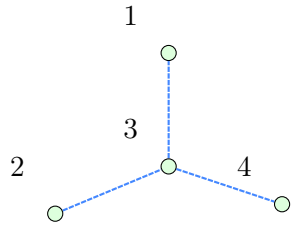


Figure 2.8: Open graph of a measurement pattern implementing a CNOT gate. Nodes represent qubits and edges between nodes represent entangle operations between respective qubits

of computation (each branch relates to a different bit string representing measurement results) and maps that are implemented by those branches which are called *branch maps*. One uses corrections to account for the possible measurement errors (obtaining 1 rather than 0 when measuring a qubit) to obtain a desired transformation.

In the literature, multiple types of determinism are usually distinguished, we give their definitions as stated by Danos et al. [10]:

Definition 2.2.4. *A measurement pattern $\mathcal{P}$ is **deterministic** if it implements a completely positive and trace preserving map that sends pure states to pure states. Or in other words, if maps implemented by different branches of computation are proportional up to a scalar.*

Definition 2.2.5. *A measurement pattern $\mathcal{P}$ is **strongly deterministic** when all branch maps are equal.*

Remark 2.2.5.1. *Strongly deterministic patterns realise unitary transformations.*

Definition 2.2.6. *A measurement pattern $\mathcal{P}$ is **uniformly deterministic** if for all choices of measurement angles for any qubit it is still deterministic.*

Definition 2.2.7. *A measurement pattern $\mathcal{P}$ is **stepwise deterministic** if after performing each measurement and the corrections depending on that measurement the pattern is still deterministic (no matter if measurement results in 0 or 1 the same map is implemented).*

In most of literature, all of this conditions are desired or assumed and as such, they are usually combined into the notion of *robust determinism*, defined as:

Definition 2.2.8. *We say that a measurement pattern $\mathcal{P}$ is **robustly deterministic** if it is uniformly, stepwise, strongly deterministic.*

2.2.8 Flow conditions

Determining if a measurement pattern is deterministic and computing which corrections need to be applied for a pattern to be deterministic are two important tasks that could significantly reduce the difficulty of using MBQC and increase its popularity. While those questions have not been studied very well for general patterns, it has been shown that there exist certain conditions on the open graph of a pattern that guarantee that the pattern is robustly deterministic.

First such condition is existence of causal flow introduced by Danos et al [10], which is sufficient for the measurement pattern to be robustly deterministic. We define causal flow in the following way:

Definition 2.2.9. *An open graph (G, I, O) has **causal flow** if there exist a map $f : V \setminus O \to V \setminus I$ and partial order $\prec$ (also referred to as the time order) such that for all vertices $i \in V \setminus O$:*

- $i \prec f(i)$
- $f(i) \in N(i)$
- $\forall j \in N(f(i)), i \prec j \vee i = j$

It is however known that existence of causal flow is not a necessary condition for robust determinism. A generalization of causal flow has been introduced by Browne et al [6] which is sufficient and necessary for the measurement pattern to be robustly deterministic. We define generalized flow for measurement pattern with only (X, Y) measurements as follow:

Definition 2.2.10. *An open graph (G, I, O) has **generalized flow** (or **gflow**) if there exist a map $g : V \setminus O \to \mathcal{P}^{V \setminus I}$ and partial order $\prec$ (also referred to as the time order) such that for all vertices $i \in V \setminus O$:*

- $\forall j \in g(i), i \prec j$

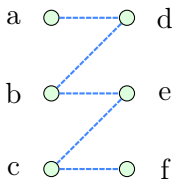


Figure 2.9: Open graph with casual flow $f(a) = d$, $f(b) = e$, $f(c) = f$ and gflow $g(a) = \{d, e, f\}$, $g(b) = \{e, f\}$, $g(c) = \{f\}$

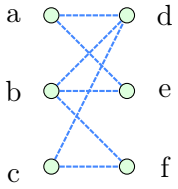


Figure 2.10: Open graph with only gflow: $g(a) = \{d, f\}$, $g(b) = \{d, e, f\}$, $g(c) = \{d, e\}$

- $\forall j \in Odd(g(i)), i \prec j \vee i = j$
- $i \in Odd(g(i))$

In general gflow is not unique, however there always exists *maximally delayed gflow* (which is a gflow with minimal depth) which can be used to determine correction sets. Furthermore, similarly to causal flow, it can be computed in polynomial time by an algorithm presented by Mhalla et al [27]. An example of graph with causal flow (and generalized flow as causal flow gives a valid generalized flow) can be seen in figure 2.9, and an example of graph with only gflow can be seen in figure 2.10.

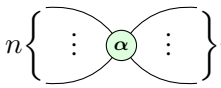
2.3 ZX calculus

ZX calculus is a rigorous graphical language used to reason about quantum computation, introduced by Coeck et al [9]. It is very well suited to reasoning about MBQC and has been applied in many areas like optimization of quantum circuits [13, 22], simulation of quantum circuits [7, 26] and compilation to circuits abiding by hardware constraints [24]. It can be shown that any equality in quantum information can be derived by a sequence of rewrite rules in ZX calculus, that is, ZX calculus is complete [29, 30]. This fact makes it a powerful and convenient tool to reason about quantum computing and MBQC in particular.

2.3.1 ZX DIAGRAMS

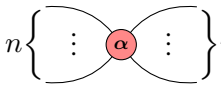
At the heart of ZX calculus lie diagrams on which the graphical rewrites are performed. Remarkably, given all of its power, ZX diagrams are built from few very simple building blocks, namely Z and X spiders, Hadamard gates and wires which are used to connect spiders and Hadamard gates together.

Z spider with n inputs and m outputs is a family of linear maps that takes in $\mathbb{C}^{2^m \times 2^n}$ and are defined as:



$$:= \otimes_{i=1}^{m} |0\rangle \otimes_{i=1}^{n} \langle 0| + e^{i\alpha} \otimes_{i=1}^{m} |1\rangle \otimes_{i=1}^{n} \langle 1|$$

Where $\otimes_{i=0}^{m} |\psi_i\rangle = |\psi_0\rangle \otimes \cdots \otimes |\psi_m\rangle$ (so, for example $\otimes_{i=3}^{3} |0\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle$) and analogously for $\otimes_{i=0}^{m} \langle\psi_i|$. Similarly, X spider is defined in the similar manner, but for the X basis:



$$:= \otimes_{i=1}^{m} |+\rangle \otimes_{i=1}^{n} \langle +| + e^{i\alpha} \otimes_{i=1}^{m} |-\rangle \otimes_{i=1}^{n} \langle -|$$

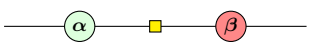
Finally, the Hadamard gate and wire are defined as the Hadamard unitary and the identity (since the wire's purpose is to "pass information") respectively. We sometimes refer to wires with Hadamard gates on them as *Hadamard edges*.

$$:= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$:= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

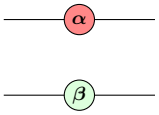
These building blocks can be composed together by sequential and parallel composition to build larger ZX diagrams which relate to multiplication and tensor product respectively. Examples of both types of composition can be seen below:

Sequential composition:



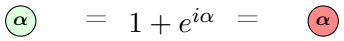
$$= (|+\rangle \langle +| + e^{i\alpha} |-\rangle) H (|0\rangle \langle 0| + e^{i\beta} |1\rangle \langle 1|)$$

Parallel composition:

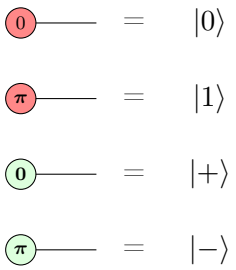


$$= (|+\rangle \langle +| + e^{i\alpha} |-\rangle \langle -|) \otimes (|0\rangle \langle 0| + e^{i\beta} |1\rangle \langle 1|)$$

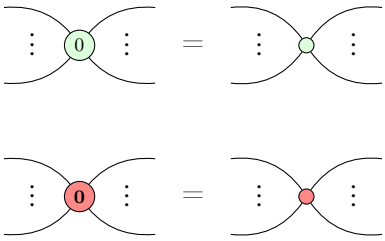
All the examples provided so far represented Z and X spiders as maps, however they can also represent states and scalars. For example:



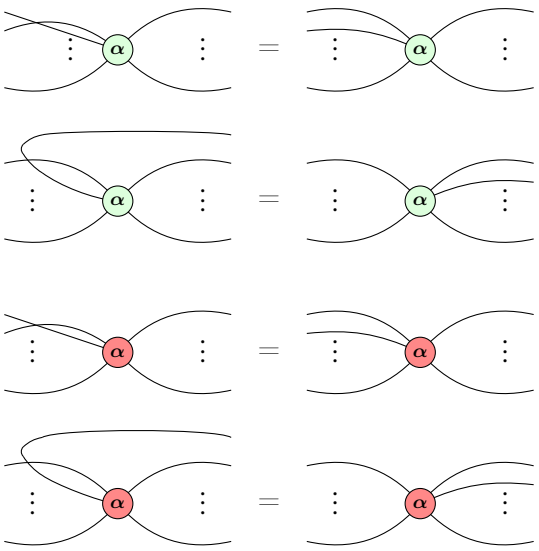
Z and X spiders can also represent, up to a non-zero scalar, $|+\rangle, |-\rangle$ and $|0\rangle, |1\rangle$ states:



It is noted that the equal signs above is equal up to a non-zero scalar, in fact there is a $\sqrt{2}$ factor missing however since in this thesis only equality up to a non-zero scalar is relevant; we will use standard equal sign to denote it. Furthermore, as a convenience we omit the phase when it is equal to 0, that is:



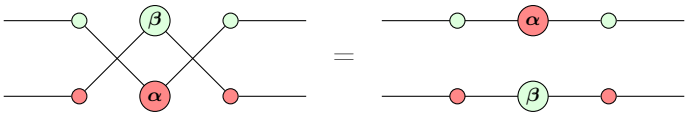
We have defined spiders in terms of inputs and outputs, however, it turns out that such distinction is not needed. Spiders are invariant under swapping some of the wires, even when swapping inputs with outputs etc. [44].



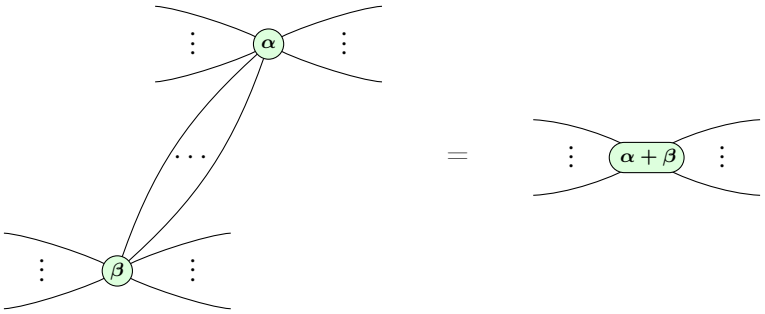
2.3.2 Identities

ZX diagrams provide us with the ability to represent different stages of quantum computation, however its the rewrite rules that allow us to reason about quantum computation and show various equities.

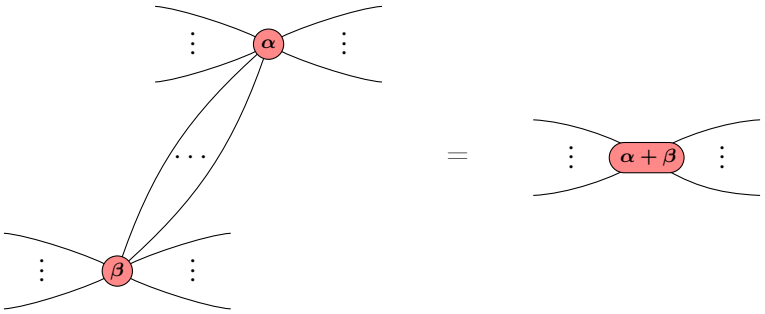
A particularly useful rule is *only connectivity matters* (ocm), which states that any two diagrams that have the same spiders with the same phases on them, connected by the same edges (with Hadamard gates on the same wires) with the same order of inputs and outputs are equal. In other words, ZX diagrams can be deformed by bending wires, moving spiders and Hadamard gates around without change to the underlying map, if the order of inputs and outputs is preserved. For example:



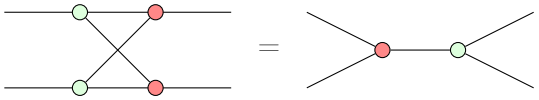
Deriving most of ZX calculus rules can be done with the following four base rules [23]. Completeness results mentioned before require some additional rules however as they are not relevant to the work done in this thesis we will omit them.



(2.1)



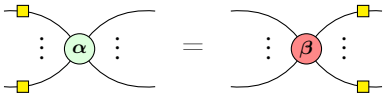
(2.2)



(2.3)

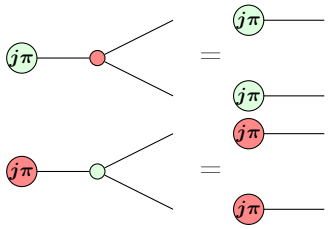
(2.4)

The rule represented by equations 2.1 and 2.2 is called spider fusion, the rule in equation 2.3 is called strong complementarity and the rule in equation 2.4 is called the Y-rule, which can also be replaced by the much more popular color change rule [23] shown in equation 2.5:

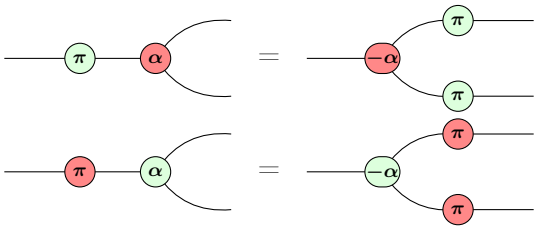


(2.5)

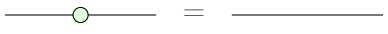
From those rules one can derive some useful identities like the copy laws shown in equation 2.6, the π-commute law shown in equation 2.7, the identity laws shown in equation 2.8 and the hh law shown in equation 2.9 [23].



for $j \in \{0, 1\}$ (2.6)



(2.7)

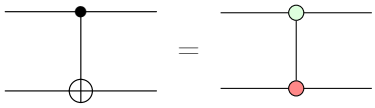


(2.8)

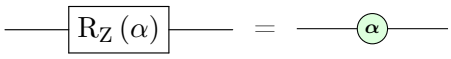
(2.9)

2.3.3 Quantum circuits in ZX calculus

One can represent quantum circuits as ZX diagrams in a straight forward manner. Firstly, one can implement $CNOT$ gate as follows:



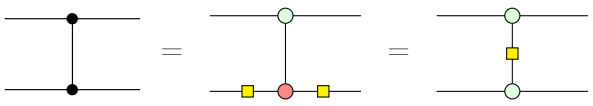
Furthermore, it is easy to see that one can implement R_z as a Z-spider with one input and one output:



Similarly, one can implement R_x gate as an X-spider:

Since, as already stated, any single qubit unitary can be implemented by composition of rotations around two different axis by some angle, we can implement any single qubit unitary with the mentioned R_z and R_x gates. Moreover, since as already seen, we can represent a CNOT gate in a ZX diagram, hence we can represent arbitrary unitaries and therefore arbitrary quantum circuits [13]. As such, to represent any quantum circuit as a ZX diagram, it suffices to decompose it into CNOT gates, R_z and R_x gates and then change those gates into their equivalent ZX diagrams.

We mention some additional gates implemented in ZX diagram that will be useful throughout this work, firstly, Controlled-Z gate:



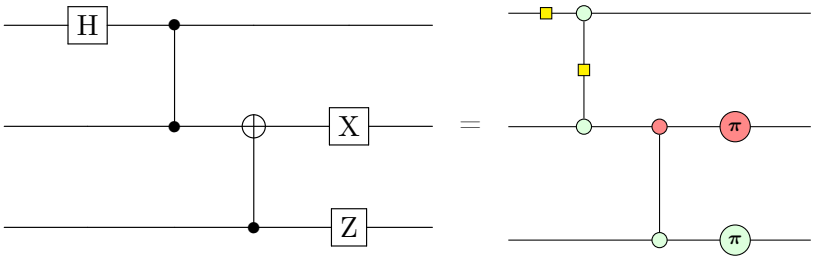
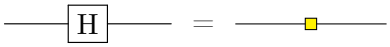


Figure 2.11: Example of quantum circuit and related ZX diagram

And the already mentioned Hadamard gate:



As it can be seen in figure 2.11, translating quantum circuits to ZX diagrams is incredibly simple, the converse however is not true as shown by de Beaudrap et al [4]. They prove that extracting a quantum circuit that is equivalent to an arbitrary ZX diagram can be #P-hard, additionally, they provide hardness results for approximate extraction. That being said, there exist families of ZX-diagrams, like those that admit generalized flow, from which a quantum circuit can be extracted in a resource efficient way [2].

2.3.4 Measurement-based quantum computing in ZX calculus

ZX diagrams can also encode measurement patterns as first shown by Duncan et al [14]. In that representation each qubit in the pattern $\mathcal{P}$ is a Z-spider, each entangle operation is an wire with a Hadamard gate on it connecting two Z-spiders (CZ gate between those two qubits), input qubits have incoming wires not connected to anything and, analogously, output qubits have outgoing wires. Finally, measurements are encoded as shown in figure 2.12.

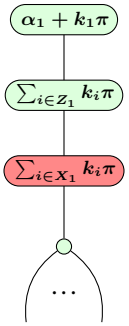


Figure 2.12: Encoding measurement in ZX-calculus, k_1 is the result of the measurement of qubit 1, Z_1 is a set of measurement results of qubits that qubit 1 Z corrections depend on, analogously X_1 for x corrections. It is noted that order of Z and X corrections is arbitrary as Z-spiders and X-spiders with π phase commute past each other (technically there is a minus added onto the phase but all phases are angles so $-\pi \mod 2\pi \equiv \pi \mod 2\pi$)

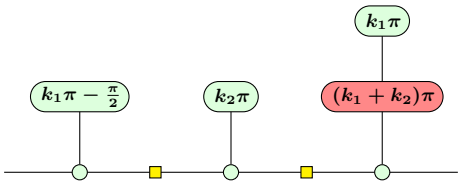


Figure 2.13: Example of encoding measurement pattern $\mathcal{P} = X_3^1 X_3^2 Z_3^1 M_2^0 M_1^{-\frac{\pi}{2}} E_{23} E_{12} N_2$ with $V = \{1, 2, 3\}$, $I = \{1\}$ and $O = \{3\}$

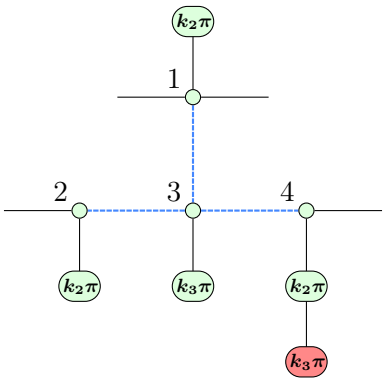


Figure 2.14: Example of encoding measurement pattern implementing CNOT, $\mathcal{P} = X_4^3 Z_4^2 Z_1^2 M_3^0 M_2^0 E_{13} E_{23} E_{34} N_3 N_4$ with $V = \{1, 2, 3, 4\}$, $I = \{1, 2\}$ and $O = \{1, 4\}$. We represent edges with an Hadamard gate on them (Hadamard edges) by dashed blue edges

Interestingly, any ZX-diagram can be represented as a measurement pattern without corrections - that is, it represents the branch map where all measurements are 0 and no corrections are necessary. This is because any ZX-diagram can be put into a *graph-like* form as shown and defined by Duncan et al [13]:

Definition 2.3.1. *A ZX-diagram is* ***graph-like*** *if:*

- *All spiders are Z-spiders.*
- *Z-spiders are connected by Hadamard edges (edges with a Hadamard wire on it).*
- *There are no parallel Hadamard edges or self loops.*
- *Every input or output is connected to a Z-spider and every Z-spider is connected to at most one input or output.*

It can be seen that if we use spider fusion to "unfuse" Z spiders with non zero phase, the resulting diagram will look like a diagram resulting from the previously mentioned protocol to encode measurement patters, as such one can read out a pattern from it and hence every ZX-diagram corresponds to a measurement pattern.

2.4 Related Work

Work in translation of one-way computations to quantum circuits has mostly focused on measurement patterns whose open graphs have some specific properties. That being said, a general purpose method for translation of arbitrary measurement patterns has been developed Broadbent et al [5]. In that work an additional method has been shows that reduces the depth of computation of the circuit (length of the critical path) to $O(d \log(|V \setminus O|))$, however it requires $O(I + |V \setminus O|^3)$ qubits, where V is the set of the corresponding pattern, I and O are sets of inputs and outputs and d is depth of the pattern (which is defined as longest sequence of commands in the pattern such that each command depends on the previous one).

Most of work in the field focused on extraction techniques that produce circuits without ancillary qubits, which can be done for some specific families of patterns. Besides

the main two conditions, that is existence of flow and gflow, other flow conditions have also been investigated in the literature, for example Simmons provides a method of extracting circuits from measurement patterns with *Pauli flow* [41] and de Beaudrap with *modified flow* [3].

The main method, that this work has been inspired by and improves on, has been introduced by Danos et al [10] and is called star pattern translation. It can be used to translate measurement patterns, which underlying open graph admits causal flow, to circuits consisting of controlled-Z gates and family of parameterized gates introduced by Danos et al [11]. Straightforward application of this method to measurement patterns with generalized flow results in acausal gates (multiqubit gates that act on different qubits at different time steps) which can be studied in the context of closed timelike curves [39]. Multiple methods of alleviating this issue have been proposed. Duncan et al [14] proposed use of ZX calculus to rewrite graphs with gflow to graphs with flow. Miyazaki et al [28] provide a direct extension of the original star pattern translation by finding a specific path cover and eliminating acausal CZ gates by inserting additional two-qubit gates. Dias da Silva et al [39] take a different approach and provide a procedure that first applies the general purpose method producing a circuit with $|V|$ qubits and later simplifies the circuit using certain rewrite rules that remove ancillary qubits, however they leave certain aspects of the algorithm as open problems and do not prove that this approach works for an arbitrary pattern with gflow. These problems are later fixed in their future work where they provide an algorithm that rewrites the extended circuit, which is the result of general purpose algorithm, and produces a circuit with all ancillary qubits removed [40].

Other translation methods that require existence of gflow focus on extraction of quantum circuits out of ZX diagrams, which, due to close connections between ZX diagrams and measurement patterns, is closely connected to the problem of extracting quantum circuits out of measurement patterns. Duncan et al [13] provide a method of extraction for ZX diagrams with gflow and Backens et al [2] extend that method to diagrams that admit gflow with three measurement planes.

Our method, as opposed to, various "flow requiring" methods (like star pattern translation, Dias da Silva et al method etc), does not aim to produce a full time ordering of qubits. We introduce a notion of *partial causal flow*, which is an extension of causal flow that always exists in any "sensible" measurement pattern. The main idea is that we add qubits to the partial causal flow only if they are compatible with current time ordering and the time ordering imposed by corrections in the pattern. We then use that partial flow to put some of the qubits in the measurement pattern on the same wire of the quantum circuit in a similar way to star pattern translation and all the qubits outside of the time ordering are then implemented in a similar manner to the general purpose method proposed by Broadbent et al [5]. This approach, as it will later become apparent, allows to reduce number of qubits and two-qubit gates while also not being restricted by existence of any flow.

Finally, this work also further advances our understanding of spatial resources required for implementing arbitrary measurement patterns. Such questions have been previously studied, for example Houshmand et al. [21] investigate the minimal number of qubits required for implementing patterns with flow and gflow.

3 Circuit extraction

Current methods of translating measurement patterns to quantum circuits without ancillary qubits restrict themselves to only patterns, whose open graphs satisfy some specific conditions such as the existence of causal flow or generalized flow. Such conditions are quite restrictive and do not use additional information provided in the pattern, like corrections.

On the other hand, general purpose methods that translate arbitrary measurement patterns to quantum circuits are inefficient in terms of spacial resources and hence they are often infeasible when considering current state of physical quantum computers.

This project aimed to use additional information in the measurement pattern, beyond just the open graph, to relax some of the conditions required for extraction of quantum circuits without introducing a significantly number of ancillary qubits. Originally, the idea was to remove the uniform determinism requirement on patterns, which was present in most extraction methods due to requirements such as existence of causal flow or generalized flow or only slightly relaxed in case of extractions using Pauli flow. However, throughout this project, this approach has been shifted into finding path covers on the open graph, that would abide by the time ordering imposed by the measurement pattern and its corrections, inspired by how Miyazaki et al [28] extended star pattern translation to patterns with gflow. Later we apply "star pattern translation like" method to build a circuit that implements the measurement pattern, based on the path cover, with each path becoming a wire in a circuit.

Results presented in this Chapter are results of those considerations. Firstly, in section 3.1 we introduce the concept of partial causal flow which allows us to find a partial path cover and is the foundation for "star pattern like" part of the extraction

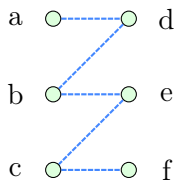


Figure 3.1: Open graph with casual flow $f(a) = d$, $f(b) = e$, $f(c) = f$

algorithm. In that section we also discuss some possible algorithms of finding said partial causal flow and their advantages and drawbacks. Then in section 3.2 we present the algorithm that, given partial causal flow on a measurement pattern and the pattern, extracts a quantum circuit out of that pattern. Furthermore, we provide theoretical guarantees on the performance of that algorithm. Finally, in section 3.3 we compare the algorithm introduced in this work with other methods of extracting quantum circuits out of measurement patterns.

3.1 Partial causal flow

We first present the concept of *partial causal flow* which is an extension of causal flow, that allows for some of the qubits to not be the part of the flow. Later, in the extraction algorithm those "outside" qubits will be turned into ancillary qubits in the resulting circuit. We define partial causal flow as follows:

Definition 3.1.1. *An open graph (G, I, O) has* ***partial causal flow*** *if there exist a* ***partial*** *function $f_{par} : V \setminus O \to V \setminus I$ and partial order $\prec$ such that for all vertices i in the domain of f_{par} we have:*

- $i \prec f_{par}(i)$
- $f_{par}(i) \in N(i)$
- $\forall j \in N(f_{par}(i)), i \prec j \vee i = j$

The first thing to note is that partial causal flow is not unique and its quality may differ drastically. For example, the causal flow of the graph presented in figure 3.1 is also a valid partial causal flow, however, so is, a partial function $f_{par}(a) = d$ (with only

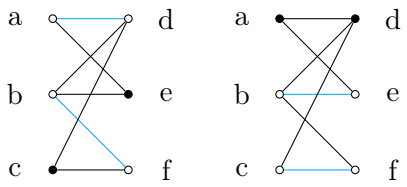


Figure 3.2: Example of two different largest partial causal flows in the same graph (with $I = \{a, b, c\}$, $O = \{d, e, f\}$), cyan lines indicate flow, that is on the left we have $f_{par}(a) = d$, $f_{par}(b) = f$ and on the left $f_{par}(b) = e$, $f_{par}(c) = f$

a in domain). Naturally, we would prefer the "full" causal flow function rather than just a subset of it, and in the case of graphs where there is no causal flow, we would prefer the largest possible causal flow. This is because when using the extraction method proposed in this work, the larger the partial causal flow, the smaller the amount of qubits and two-qubit gates in the extracted circuit (as it is stated in lemma 3.2.1 and remark 3.2.1). We can define the largest partial causal flow formally as:

Definition 3.1.2. *We call a partial causal flow f_{par} the* ***largest partial causal flow*** *if domain of f_{par} is the largest, that is, if there is no partial causal flow f'_{par} such that $|domain(f'_{par})| > |domain(f_{par})|$. We say $|domain(f_{par})|$ is the* ***size of the partial causal flow***

Concept of largest partial causal flow does not result in uniqueness, as can be seen in figure 3.2, however it ensures that if the graph has flow then largest partial causal flow is also the causal flow (there cannot be larger partial causal flow since if the graph has flow then all possible nodes are part of the domain of that flow function, hence causal flow is the largest partial causal flow).

Before turning to the question of computing partial causal flow, we must first ensure that the computed partial causal flow is consistent with corrections that are in the pattern (which impose some conditions on the time ordering $\prec$). As such, we will extend the definition of partial causal flow to measurement patterns rather than just open graphs. We do so by adding conditions imposed by corrections from the pattern to the partial order and result in the following final version of the definition.

Definition 3.1.3. *A measurement pattern $\mathcal{P}$ has **partial causal flow** if there exist a **partial** function $f_{par} : V \setminus O \to V \setminus I$ and partial order $\prec$ such that for all vertices i in the domain of f_{par} we have:*

- $i \prec f_{par}(i)$
- $f_{par}(i) \in N(i)$
- $\forall j \in N(f_{par}(i)), i \prec j \vee i = j$

and for all corrections $C_t^s \in \mathcal{P}$ (where $C = \{X, Z\}$) we have $s \prec t$.

Theorem 3.1.1. *Any runnable pattern $\mathcal{P}$ with at least two connected qubits, and at least one of them is not an output, then there exist a partial causal flow of size at least 1.*

Proof. Take nodes $o, u \in V$ such that $u \notin O$, u is minimal element of $N(o)$ with respect to order $\prec$ imposed by corrections of $\mathcal{P}$ and there is no correction C_u^o (if o is an output then this cannot be the case, if it is not, then both u and o are not an output and hence we simply find a minimal element of $N(u)$ that is not an output and take it instead of o) where $\prec$ is the order imposed by corrections in $\mathcal{P}$. There has to be an minimal element, since $\prec$ contains only conditions from corrections, if there were not one then there would be chain of corrections that create a loop requiring for some correction C_s^t to have $s \prec t$ and $t \prec s$ but then t would have to be measured before and after s, making the pattern not runnable. Setting $f(u) = o$ gives conditions $u \prec o$ and $u \prec t, \forall t \in N(o)$ (with the exception of $t = u$). The first condition is true since by construction there is no C_u^o, so $\neg(o \prec u)$ when considering only conditions imposed by corrections, the second condition is true because u is minimal. As such, we can always find partial causal flow $(f, \prec)$ which has size of at least 1 □

The most straightforward approach that one can take to compute partial causal flow for a given pattern is a greedy approach presented in algorithm 1. Although, if one selects which edges (i, f(i)) are added to the flow in the correct order, this approach does find the largest partial causal flow; however, in practice it is not obvious how to find that correct order. As such, we take a different approach to computing partial

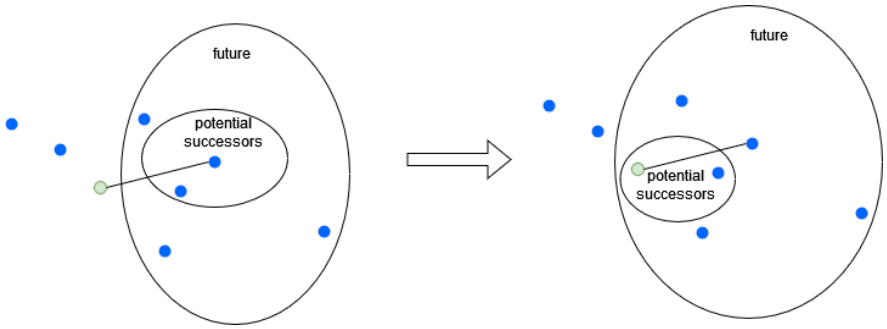


Figure 3.3: Example of adding a point u (marked green) to the flow (setting f(u)=v, where v is the indicated neighbour of u)

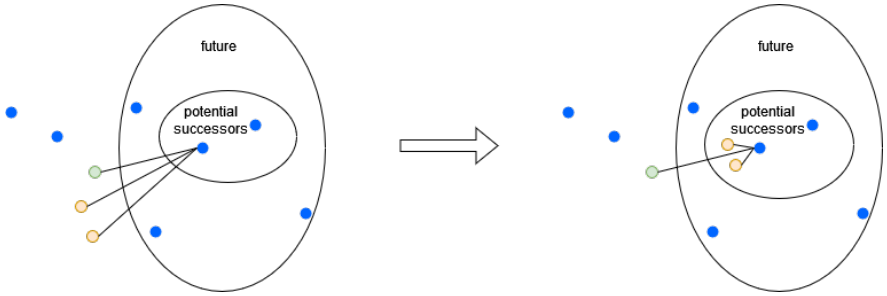


Figure 3.4: Example of the "ancillary" step where nodes (marked orange) are added to the future and used as ancillary qubits in the extraction, allowing for the node (marked green) to be added to the flow at the next iteration, we assume that both orange points are later in the time order imposed by corrections than the green node

causal flow and extend a causal flow finding algorithm provided by Mhalla et al. [27]. In that algorithm, flow is computed from the "back", that is we start with set of potential successors equal to set of outputs minus set of inputs $O \setminus I$ and set of nodes in the future equal to the set of outputs O. Then the algorithm, checks if any nodes v in the potential successor set have only one neighbour u not in the future, and if so, set $f(u) = v$ and updates the set of potential successors and the "future" set. The algorithm continues to do these operations until no new nodes are added, example of one such iteration can be seen in figure 3.3. We extend this algorithm by introducing an additional step, if no nodes with only one neighbour in the past are found, we select the node which neighbourhood, that is not in the future, is the smallest and add all but one of nodes in that neighbourhood to the future and potential successors set making the originally selected node suitable to be added to the flow. Example of this can be seen in figure 3.4. This added nodes will turn into ancillary qubits that are "outside" of the time ordering. This procedure continues until no further changes are done and each node is

either part of the partial flow or moved to the future as an ancillary qubit. An extra attention needs to be paid to make sure the time ordering obeys conditions specified by the corrections in the pattern, as such pair (u, v) is only added to the partial flow if the current ordering does not imply $v \prec u$ and only nodes that are maximal with respect to nodes that are not in the future in the ordering (node s is maximal with respect to set A if there is no node $t \in A$ such that $s \prec t$) are added as ancillary qubits. Algorithm 2 provides a pseudo code for this procedure.

Algorithm 1 Greedy algorithm for finding partial causal flow

Input: measurement pattern $\mathcal{P}$
Output: partial causal flow and the imposed ordering $(f_{par}, \prec)$

$G \leftarrow$ open graph of $\mathcal{P}$
partial function $f_{par} f$, with no argument-value pairs added
empty partial order $\prec$
for correction $C_t^s \in \mathcal{P}$ **do**
    add $s \prec t$ to $\prec$
end for
for $(u, v) \in E(G)$ **do**
    if adding $u \prec v$ and $\forall t \in N(v), u \prec t$ to $\prec$ does not break transitivity of $\prec$ **then**
        $f_{par}(u) = v$
        add $u \prec v$ to $\prec$
        for $t \in N(v) \setminus \{u\}$ **do**
            add $u \prec t$ to $\prec$
        end for
    end if
end for
return $f_{par}, \prec$

While, sadly, algorithm 2 does not always find largest partial causal flow (as it can be seen in figure 3.5), we guarantee another desirable property that is, if the flow exists, the partial causal flow returned by algorithm 2 is that flow (under some technical assumptions).

Remark 4.1. *If the open graph of a measurement pattern $\mathcal{P}$ admits causal flow $(f, \prec)$ found by the algorithm by Mhalla et al [27] and for all corrections $C_t^s \in \mathcal{P}$ we have $s \prec t$, then algorithm 2 finds $(f, \prec)$*

Proof. If the open graph G of the measurement pattern $\mathcal{P}$ admits causal flow, by the proof of the original algorithm by Mhalla et al [27], at each iteration until termination there is $v \in potential\_successors$ that satisfies the condition on line 2 of subroutine 3

Algorithm 2 Partial causal flow finding algorithm

Input: measurement pattern $\mathcal{P}$ with G being open graph, I set of inputs and O set of outputs

Output: partial causal flow and the imposed ordering $(f_{par}, \prec)$

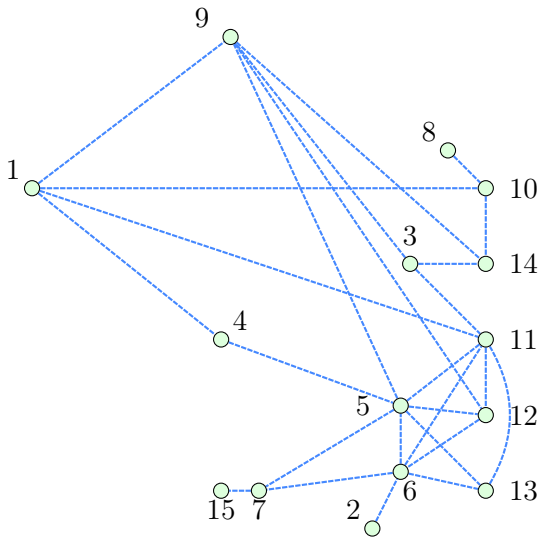
partial function f_{par}, with no argument-value pairs added
empty partial order $\prec$
for correction $C_t^s \in \mathcal{P}$ **do**
    add $s \prec t$ to $\prec$
end for
$potential\_successors \leftarrow O \setminus I$
$future \leftarrow O$
while future is changed **do**
    $successor\_removals \leftarrow \emptyset$
    $future\_additions \leftarrow \emptyset$
    $past \leftarrow V(G) \setminus future$
    ▷ part of the original algorithm - "flow finding" step
    flow-step() ▷ defined in algorithm 3
    $potential\_successors \leftarrow (potential\_successors \setminus successor\_removals)$
    $potential\_successors \leftarrow potential\_successors \cup (future\_additions \cap V) \setminus I$
    $future \leftarrow future \cup future\_additions$
    ▷ "ancillary qubits" step
    ancillary-step() ▷ defined in algorithm 4
end while
return $f_{par}, \prec$

Algorithm 3 flow-step

for $v \in potential\_successors$ **do**
    if $N(v) \cap past = \{u\}$ **then**
        if conditions $u \prec v$ and $u \prec t, t \in N(v) \setminus \{u\}$ are consistent with $\prec$ **then**
            $f_{par}(u) \leftarrow v$
            $future\_additions \leftarrow future\_additions \cup \{u\}$
            $successor\_removals \leftarrow successor\_removals \cup \{v\}$
            add $u \prec v$ to $\prec$
            for $t \in N(v) \setminus \{u\}$ **do**
                add $u \prec t$ to $\prec$
            end for
        end if
    end if
end for

Algorithm 4 ancillary-step

```
1: if successor_removals is empty then
2:     min_v ← None
3:     min_num_past_neigh ← ∞
4:     for v ∈ potential_successors do
5:         if v is maximal with respect to V(G) \ future then
6:             if |N(v) ∩ past| < min_num_past_neigh then
7:                 min_v ← v
8:                 min_num_past_neigh ← |N(v) ∩ past|
9:             end if
10:        end if
11:    end for
12:    if min_v ≠ None then
13:        added ← 0          ▷ Sorting with respect to ≺, starting from the largest node
14:        for n ∈ sorted(N(min_v) ∩ past, ≺) do
15:            if added < |N(min_v ∩ past)| − 1 then
16:                future ← future ∪ {n}
17:                if n ∉ I then
18:                    potential_successors ← potential_successors ∪ {n}
19:                end if
20:                added ← added + 1
21:            end if
22:        end for
23:    end if
24: end if
```



$$X^2_{12}X^2_{11}X^4_{12}X^4_{11}X^5_{12}X^5_9X^5_{11}X^6_9X^6_{11}X^{12}_7X^{11}_7X^{15}_{12}X^{15}_{11}$$
$$X^8_{12}X^8_{10}X^8_{11}X^8_{14}X^9_{12}X^9_{13}X^1_{12}X^1_{11}X^1_{14}X^3_{12}X^3_{13}X^3_{14}$$

Figure 3.5: Example of a graph and corrections with inputs $I = \{1, 2, 3, 4, 8\}$ and outputs $O = \{10, 11, 13, 12, 14\}$, for it for which algorithm 2 finds the partial causal flow $f_{par}(3) = 11, f_{par}(5) = 13, f_{par}(8) = 10$. Another,larger partial causal flow can be defined as $g_{par}(2) = 6$, $g_{par}(4) = 5$, $g_{par}(8) = 10$, $g_{par}(15) = 7$

and it is the same v that would be found by the original algorithm. Furthermore, since order generated by the flow agrees with order generated by the corrections, condition on the line 3 of subroutine 3 is also satisfied. As such the algorithm performs exactly the same operation as the original flow finding algorithm, and since as long as there are nodes left in the graph that were not added, flow-step will add some node to the flow, causing the ancillary-step to not execute. The ancillary-step is only executed when all nodes are added to the flow, but then it performs no operation since there is no min_v. this means that our algorithm perfectly recovers the original algorithm and returns $(f, \prec)$ □

Finally, we show that the ordering corresponding to the partial causal flow that is the result of algorithm 2 is valid and does not conflict with the ordering imposed by the corrections of the pattern.

Theorem 3.1.2. *Algorithm 2 returns a valid partial causal flow on the measurement pattern $\mathcal{P}$ given as input.*

Proof. First we show that the algorithm terminates, indeed since the stopping condition for the loop is the fact that future set is not updated. Since we only add nodes to future and there is finite amount of nodes $n \in V(G)$ then the stopping condition will be reached.

Now, it is sufficient to show that the resulting partial order $\prec$ is valid, that is, there is no $u, v \in V(G)$ such that $u \prec v$ and $v \prec u$. However, clearly by condition on the line 3 of subroutine 3, only nodes that do not break $\prec$ are added to the flow, as such the resulting flow is valid. □

We conclude our discussions of computing partial causal flow by providing complexity of algorithm 2.

Theorem 3.1.3. *Algorithm 2 has the time complexity of* $O(|V|^4)$

Proof. Since the main loop (line 8 of algorithm 2) runs as long as at least one node is added to the future set, then it can run for at most $|V|$ steps.

flow-step iterates through set of *potential_successors* which is of size at most $|V|$, getting an intersection of neighbourhood and past set can be done in at most $O(|V|)$ with appropriate data structures used. Then creating set of conditions takes at most $O(|V|^2)$ and detection of cycles in $\prec$ can take $O(|V|+|V|^2)$ (cycle detection in a graph with $|V|^2$ edges, which is an upper bound on number of edges in a directed graph representing $\prec$), finally adding node to flow will take another $|V|^2$ because of conditions needing to be added to $\prec$. Summarizing flow-step takes $O(|V|(|V| + |V|^2 + |V| + |V|^2 + |V|^2)) = O(|V|^3)$.

If the ancillary-step is executed then the for loop at line 4 of algorithm 4 does at most $|V|$ iterations, checking if v is maximal can be done in $O(|V|)$ and size of the intersection of neighbourhood and past can be also computed in $O(|V|)$. Sorting the list of the neighbours (line 14) is done in $O(|V| \log |V|)$ and iterating through that list takes at most $|V|$ iterations. Putting this all together gives the time complexity of ancillary-step equal to $O(|V|(|V| + |V| + |V| \log |V| + |V|)) = O(|V|^2 \log |V|)$

As such, the complexity of the entire algorithm is $O(|V|(|V|^3 + |V|^2 \log |V|)) = O(|V|^4)$. □

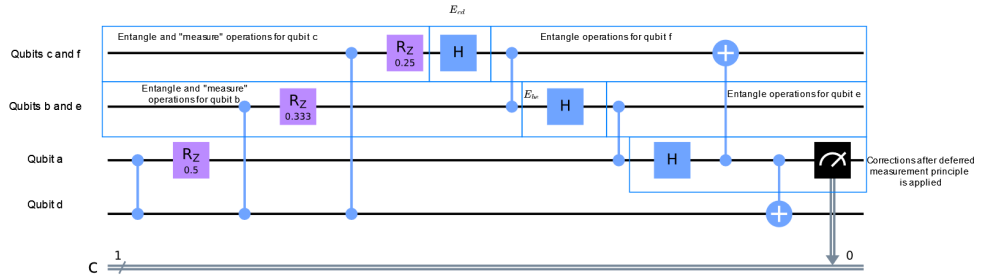


Figure 3.6: Result of algorithm 5 when applied to measurement pattern $\mathcal{P} = X_d^a X_f^a X_d^b X_e^b X_f^b X_d^c X_e^c M_a^{\frac{1}{2}} M_b^{\frac{1}{3}} M_c^{\frac{1}{4}} E_{ad} E_{ae} E_{bd} E_{be} E_{bf} E_{cd} E_{cf} N_f N_e N_d$, with $V = \{a, b, c, d, e, f\}$, $I = \{a, b, c\}$, $O = \{d, e, f\}$ (note lack of flow in the underlying open graph)

3.2 Extraction algorithm

Algorithm 5

Input: measurement pattern $\mathcal{P}$
Output: quantum circuit implementing $\mathcal{P}$

edge-graph ← empty graph
$(f, \prec)$ ← partial causal flow on $\mathcal{P}$
qubits-wire-mapping ← qubit-wire-mapping() ▷ subroutine defined in algorithm 6
ordered-gates ← order-gates() ▷ subroutine defined in algorithm 7
qc ← empty circuit
for wire in qc **do**
    if first qubit in $\prec$ that wire is mapped to is not an input **then**
        add a Hadamard gate to that wire (wire assumed to be initialized in $|0\rangle$)
    end if
end for
for gate in ordered-gates **do**
    add gate to qc using qubit-wire-mapping to determine which wires gate acts on
end for
return qc

We now turn to the question of extracting a quantum circuit given the partial causal flow. Our method connects the star pattern translation introduced by Danos et al [10] and the general purpose method show by Broadbent et al [5]. The method presented here applies those two methods to different types of nodes, the nodes that are part of the partial causal flow (are either in domain or image of f) and the nodes that are not part of the flow. For the nodes that are part of the flow, "star pattern translation like" method is applied and the remaining nodes, that were not part of the flow, are

Algorithm 6 qubit-wire-mapping()

Input: measurement pattern $\mathcal{P}$, partial causal flow $(f, \prec)$
Output: mapping between $V(G)$ (qubits in the pattern $\mathcal{P}$) and non negative integers indicating wire indices

```
qubit-wire-mapping ← empty mapping
wire-idx ← 0
for v ∈ ordered(V(G), ≺) do    ▷ order nodes in accordance to ≺ so that maximal elements are at the end
    if v ∈ image(f) then
        u ← f^{-1}(v)                                   ▷ f(u) = v
        qubit-wire-mapping[v] ← qubit-wire-mapping[u]
    else
        qubit-wire-mapping[v] ← wire-idx
        wire-idx ← wire-idx+1
    end if
end for
for u ∈ V(G) do
    if u ∉ qubit-wire-mapping then
        qubit-wire-mapping[u] ← wire-idx
        wire-idx ← wire-idx+1
    end if
end for
return qubit-wire-mapping
```

added as ancillary qubits, that is similarly to the general purpose method, they are mapped to a separate wires and commands from the pattern are executed on those wires. The example of applying this procedure can be seen in figure 3.6 and pseudo code is presented in algorithm 5.

This long and rather monotone pseudo code performs the following operations. First it computes the partial causal flow, then it creates a mapping of qubits onto wires ensuring that any pair of qubits $(u, f(u))$ is mapped to the same wire and qubits u, t are mapped to the same wire if and only if there exist n such that $u = f^{(n)}(t)$ or $t = f^{(n)}(u)$ where $f^{(n)}$ is f applied n times (for example $f^{(2)}(x) = f(f(x))$). After that, a directed graph of operations is created that ensures that operations are performed in the right order, figure 3.7 gives the overview of the order. Finally, the gates are added to the circuit according to their position in the topological ordering of the operations graph and according to the qubit-wire mapping.

Algorithm 5 in its current form applies the principle of deferred measurement to change all of the corrections into two-qubit gates, however, one can also implement the

Algorithm 7 order-gates()

Input: measurement pattern $\mathcal{P}$, partial causal flow $(f, \prec)$
Output: ordered list of operations to execute

operations-graph ← Empty directed graph
for $u \in V(G)$ **do**
    operations ← empty list
    if $u \in image(f)$ **then**
        start-operation ← Hadamard gate on u with id 0
    else
        start-operation ← Identity on u with id 0
    end if
    entangle-corrections-separator ← Identity on u with id 1
    for $E_{ui} \in \mathcal{P}$ **do**
        if $i \neq f(u) \vee i \neq f^{-1}(u)$ **then**
            operation-graph.add-edge(start-operation, E_{ui})
            operation-graph.add-edge(E_{ui}, entangle-corrections-separator)
        end if
    end for
    if u is measured and $u \notin domain(f)$ **then**
        qubit-end-operation ← Hadamard on u with id 1
        measurement ← Measurement operation on u with appropriate angle
        operation-graph.add-edge(qubit-end-operation, measurement)
        for corrections $C_t^u \in \mathcal{P}$ **do**
            operation-graph.add-edge(qubit-end-operation, C_t^u)
            operation-graph.add-edge(C_t^u, measurement)
        end for
    else if $u \in domain(f)$ **then**
        qubit-end-operation ← Hadamard on f(u) with id 0
    else
        qubit-end-operation ← Identity on u with id 2
    end if
    if $u \notin O$ **then**
        z-phase ← Z rotation gate on u with measurement angle on u
    else
        z-phase ← Identity on u with id 3
    end if
    operation-graph.add-edge(entangle-corrections-separator, qubit-end-operation)
    operation-graph.add-edge(entangle-corrections-separator, z-phase)
    order-corrections() ▷ subroutine defined in algorithm 8
end for
return topological-sort(operations-graph)

Algorithm 8 order-corrections()

Input: measurement pattern $\mathcal{P}$, partial causal flow $(f, \prec)$

```
x-corrections ← empty list
for correction C_u^s ∈ P do
    if s ∉ domain(f) then
        if C = X then
            x-corrections.append(C_u^s)
        end if
        operation-graph.add-edge(entangle-corrections-separator, C_t^u id 0)
        operation-graph.add-edge(C_t^u id 0, z-phase)
    end if
end for
operation-graph.add-edge(z-phase, qubit-end-operation)
if u ∉ O then
    for X_u^s ∈ x-corrections do
        operation-graph.add-edge(z-phase, X_t^u id 1)
        operation-graph.add-edge(X_t^u id 1, qubit-end-operation)
    end for
end if
```

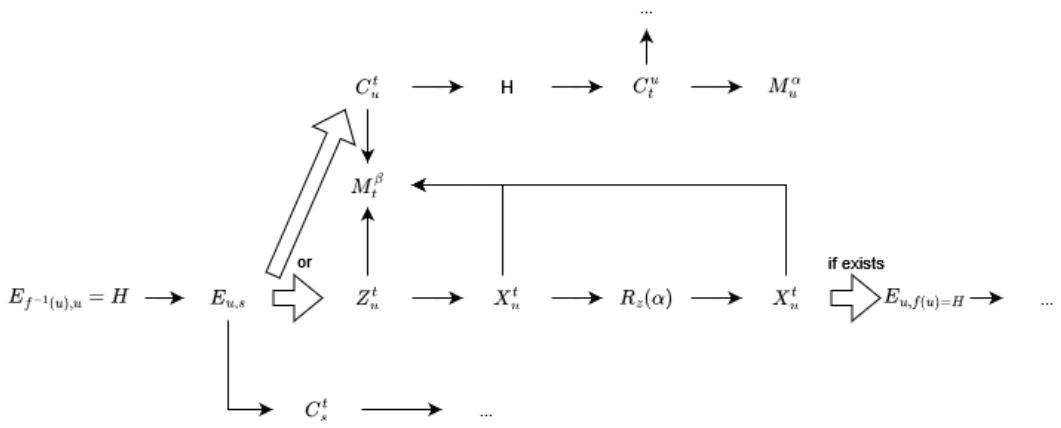


Figure 3.7: Edges in the operations graph added for node u. If one of the operations does not exist, add edge to the next one in the order. Large arrows symbolise possible options, if a qubit is measured or not and if there is a successor of the qubit in f. Note that by adding edges for other nodes u some additional edges may appear between these nodes and others, that were not specified in this graph.

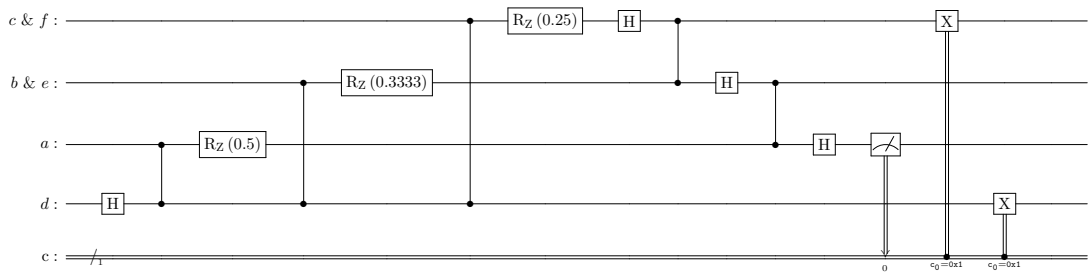


Figure 3.8: Result of our algorithm applied to the same case as in the figure 3.6 when classical control is allowed

same circuits using **classically controlled gates**. In that case, each of the controlled-Z or controlled-X gates used to implement corrections from the inputted measurement pattern is replaced with a classically controlled Z gates or classically controlled X gates respectively, and the measurement is moved before any of the classically controlled gates. This approach may be interesting as using classically controlled gates instead of the quantum ones results in better theoretical bounds, as can be seen in the next sections of this work. Furthermore, using classically controlled gates may alleviate the issue of short qubit life span as some qubits will be measured earlier than in the quantum control version of the algorithm. An example of a extracted circuit with classical control can be seen in figure 3.8 and the modification necessary to our order-gates subroutine can be seen in algorithm 9, with the change bolded.

3.2.1 Correctness

We now prove that algorithm 5 behaves in the expected manner. Firstly, we prove that it terminates:

Theorem 3.2.1. *Algorithm 5 successfully terminates and returns an output.*

Proof. In this case, it is sufficient to show that the operation graph created in subroutine 7 has no cycles, so that one can perform topological sort on it. Lets assume, by contradiction, that there is a cycle in the operations graph.

Algorithm 9 order-gates-cc()

Input: measurement pattern $\mathcal{P}$, partial causal flow $(f, \prec)$
Output: ordered list of operations to execute

operations-graph ← Empty directed graph
for $u \in V(G)$ **do**
    operations ← empty list
    if $u \in image(f)$ **then**
        start-operation ← Hadamard gate on u with id 0
    else
        start-operation ← Identity on u with id 0
    end if
    entangle-corrections-separator ← Identity on u with id 1
    for $E_{ui} \in \mathcal{P}$ **do**
        if $i \neq f(u) \vee i \neq f^{-1}(u)$ **then**
            operation-graph.add-edge(start-operation, E_{ui})
            operation-graph.add-edge(E_{ui}, entangle-corrections-separator)
        end if
    end for
    if u is measured and $u \notin domain(f)$ **then**
        qubit-end-operation ← Hadamard on u with id 1
        measurement ← Measurement operation on u with appropriate angle
        operation-graph.add-edge(qubit-end-operation, measurement)
        for corrections $C_t^u \in \mathcal{P}$ **do**
            operation-graph.add-edge(measurement, C_t^u)
        end for
    else if $u \in domain(f)$ **then**
        qubit-end-operation ← Hadamard on f(u) with id 0
    else
        qubit-end-operation ← Identity on u with id 2
    end if
    if $u \notin O$ **then**
        z-phase ← Z rotation gate on u with measurement angle on u
    else
        z-phase ← Identity on u with id 3
    end if
    operation-graph.add-edge(entangle-corrections-separator, qubit-end-operation)
    operation-graph.add-edge(entangle-corrections-separator, z-phase)
    order-corrections() ▷ subroutine defined in algorithm 8
end for
return topological-sort(operations-graph)

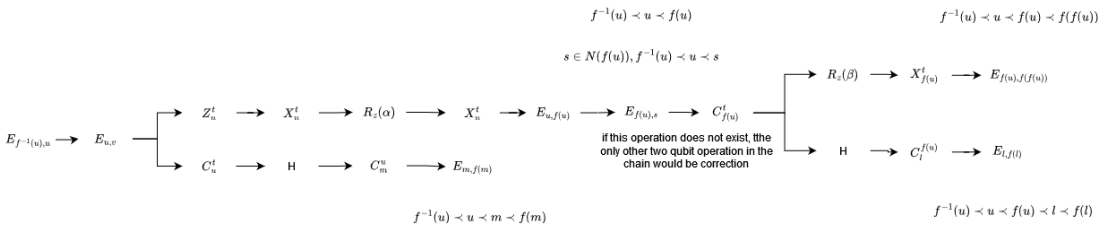


Figure 3.9: Enumeration of all possible paths to reach an entangle operation node from an entangle operation node in the operation graph, paths from $E_{u,v}$ to operations with qubit v not listed since they are analogous by symmetry. If some of the operation do not exist (no corrections etc) then there are edges introduced for operations on the same qubit and if the operations are on different qubit the path does not exist

First we note that this cycle has to have an $E_{u,v}$ for some qubits u, v in it (in this case we will also call the Hadamard gate implementing $E_{u,g(u)}$ as $E_{u,g(u)}$). This is because, any edges outgoing from a correction Z_u^v, for $u \in domian(f)$ (as can be seen in figure 3.7) has an edge to X_u^v which has an edge to $R_z(\alpha)$ which has an edge to a new X_u^v (different operation from the original X_u^v) which has an edge either to measurement or an entangle operation node. Since measurement has no outgoing edges, only possible cycle requires going through an $E_{u,f(u)}$. For $u \notin domain(f)$ edges from corrections C_u^v have edge to a Hadamard operation gate from which there are only edges to corrections of form C_s^u, note that $v \prec u \prec s$, so taking this path will result in being in a correction with both control and target larger in $\prec$, as such path, without entangle operations on it, will reach only other corrections which control and target qubits are either equal or larger in $\prec$, making it impossible to form a cycle of non entangled edges. Similar thing can be show for one qubit operation that do not replace $E_{u,f(u)}$ for some u, (as only corrections are reachable from them).

Now that we established that any potential cycle needs to have an node representing entangle operation on it, let it be node corresponding to $E_{u,v}$. Note that for an edge $E_{u,v}$, if $u \neq f(v)$ and $v \neq f(u)$, that is $E_{u,v}$ does not represent a Hadamard operation, the only incoming edges to $E_{u,v}$ are from $E_{f^{-1}(u),u}$ and $E_{f^{-1}(v),v}$, so either of those has to be in the cycle. As such, we know that there has to be a node corresponding to an operation $E_{f^{-1}(u),u}$, for some u, in any possible cycle (either $E_{u,v}$ is already in that form, or we take it predecessor). We show by enumeration that any reachable operation of form $E_{t,f(t)}$ from $E_{f^{-1}(u),u}$ satisfies $f^{-1}(u) \prec t$, this is done by enumeration as shown

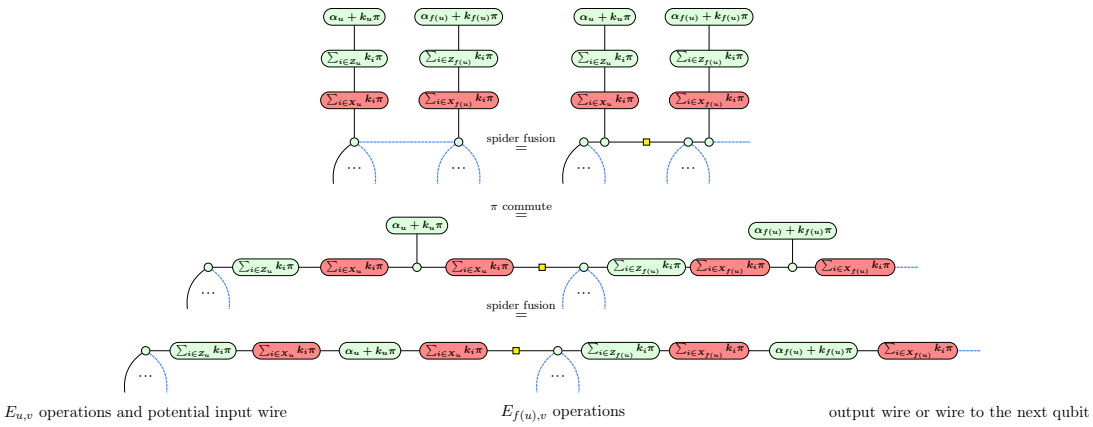
in figure 3.9. As such it is impossible to reach $E_{f^{-1}(u),u}$ by any path from $E_{f^{-1}(u),u}$, hence no cycle is possible and the algorithm terminates. □

And now we prove that the result of the algorithm implements a deterministic measurement pattern.

Theorem 3.2.2. *Given a strongly deterministic measurement pattern $\mathcal{P}$, algorithm 5 returns a circuit which implements $\mathcal{P}$*

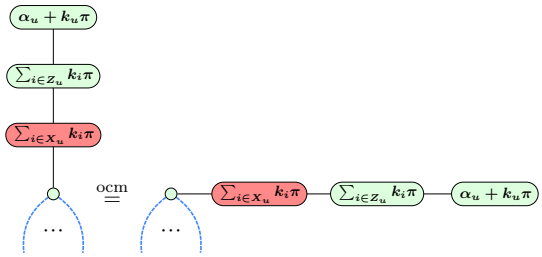
Proof. In order to prove the equality we will use ZX representation of $\mathcal{P}$. Firstly, since $\mathcal{P}$ is strongly deterministic, then we can propagate all measurements $k_u\pi$ from qubits $u \in domain(f)$ where f is the partial causal flow to its corresponding corrections $k_u\pi$ and cancel them out.

Then it can be easily seen that a sequence of applications of spider fusion, π copy and only connectivity matters rules allows us to create wires with appropriate qubits on them, as can be seen below:

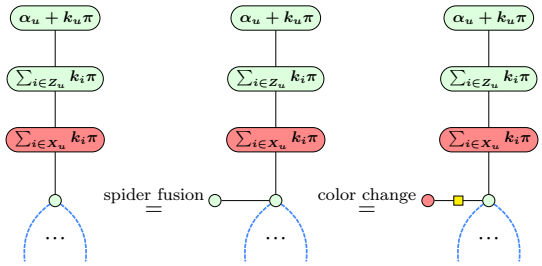


Thanks to the spider fusion rule, we can apply $E_{u,v}$ operations in any order, and specifically in the order implied by the topological sort of the operation graph (note that edges in the operation graph force corrections to be after entangle operations for both qubits and the result of applying topological sort on that operation graph gives exact time to execute each of the operations, so there is no acausal gates).

In case of qubits that are measured and have no successor in f, they can be transformed in the following way to "look" like the circuit:



Finally, any non input qubit that is first on the wire can be represented as a wire initialized to 0 with a Hadamard gate applied to it and then standard transformation, mentioned earlier, can be applied.



As such, any measurement pattern represented as ZX-diagram can be transformed into the circuit returned by algorithm 5 according to ZX-calculus rules. Given correctness of ZX-calculus, this implies that the circuit returned by algorithm 5 implements the inputted, deterministic, measurement pattern $\mathcal{P}$ □

3.2.2 Computational complexity

We now show that algorithm 5 can be executed in polynomial time.

Theorem 3.2.3. *Time complexity of the algorithm 5 is $O(P + |V|^2)$ where P is the time complexity of finding partial causal flow (which in the case of algorithm presented before, $p = O(|V|^4)$)*

Proof. It can be seen that creating a qubit-wire mapping takes $O(|V|^2)$ time, that is topological sort to get ordering according to $\prec$ takes $O(|V| + |V|^2)$ (since $|V|^2$ bounds number of edges in the graph) and then we iterate twice through list of vertices which takes $O(|V|)$.

Procedure order-gates() also takes $O(|V|^2)$ amount of time as we first iterate $V(G)$ at line 2 of subroutine 7, and then in that loop iterate over entanglements for that qubit and corrections at lines 10 and 20 respectively, both can have at most $O(|V|)$ elements as each node can correct at most all other nodes but cannot correct the same node twice (or entangle with the same node twice), order-corrections() also takes $O(|V|)$.

Finally topological sort on the operation graph takes $O(|V|^2 + |V|^2)$ as there is at most $O(|V|^2)$ operations and in subroutine 7 we add at most $O(|V|)$ edges per node, giving us $O(|V|^2)$ edges in the graph.

Finally, adding Hadamard gates to start of the wire can be done in $O(|V|)$ and creating all the gates can be done in $O(|V|^2)$. All of which together takes $O(P + |V|^2)$ time (since we also find partial causal flow in time $O(P)$). □

3.2.3 Number of twoqubit gates

An important property of the algorithm 5 is that it provides a straightforward way of bounding number of two-qubit gates in the resulting circuit. In general, circuits with lower number of two-qubit gates are more desirable as their execution time on real quantum computers is much longer than single-qubit gates and it is not possible to apply two-qubit gates between any arbitrary two-qubits in a physical device which requires using SWAP operations (which are implemented by 3 controlled-NOT gates) to make circuits executable on real hardware. As such lower number of two-qubit gates directly correlates with circuits taking less time to execute and less overhead resulting from mapping quantum circuits onto architecture of real quantum device.

Our bound exploits the fact that throughout the execution of the algorithm two-qubit gates are only added if there is a correction or if there is an entanglement operation. This fact gives rise to the following remarks:

Remark 3.2.1. *Let a circuit C with classical control be an output of algorithm 5 given a measurement pattern $\mathcal{P}$ with open graph G, then C has at most $|E(G)|$ two-qubit gates, and if the algorithm is executed with partial causal flow f, then C has exactly $|E(G)| - |image(f)|$ two qubit gates.*

Proof. In the case of classical control we assume that all corrections are implemented classically, as such the only two-qubit gates that are added, are the ones corresponding to an $E_{u,v}$ operation in $\mathcal{P}$, which corresponds directly to the number of edges of G. As such the number of two-qubit gates is at most $|E(G)|$.

Furthermore, any qubit $v \in image(f)$, has an u for which edge $E_{u,v}$ is implemented as a Hadamard gate, and that is the only case in which an edge in G is not mapped to a controlled-Z gate so C has exactly $|E(G)| - |image(f)|$ two qubit gates. □

Remark 3.2.2. *Let a circuit C, without classical control allowed, be an output of the algorithm 5 given a measurement pattern $\mathcal{P}$ with open graph G, then C has at most $|E(G)| + Z + 2X$ two qubit gates, where Z is the number of Z corrections and X is the number of X corrections in $\mathcal{P}$*

Proof. Firstly, we clearly add as many gates as in the case with classical control to implement all the entanglements, by algorithm 5, we also add one controlled-Z gate for some Z correction and two controlled-X gate for some X correction (we add those gates only for qubits that are ancillary, since for any qubits with a successor in the flow, we don't implement any corrections controlled by them) as such there is at most $|E(G)|+Z$ controlled-Z gates and $2X$ controlled-X gates in C, and no other two-qubit gates. □

3.2.4 Number of ancillary qubits

Given that an aim of this algorithm is to provide a more resource efficient way of implementing deterministic measurement patterns, it is natural to ask if one can provide a theoretical guarantee as to the number of qubits required for implementing the resulting circuit outputted by the algorithm 5 We show that this indeed is possible by providing a lower bound for the size of the largest partial flow on any given pattern. This is done by creating a specific directed graph that encodes information about the pattern and finding distance-3 independent set as defined in the definition 2.1.3 on that pattern, which, as we show encodes a partial causal flow.

We first show that given the size of a partial causal flow on a pattern, one can say exactly how many qubits will be required to implement the circuit which results from our translation method, using that partial causal flow.

Lemma 3.2.1. *Given a partial causal flow of size k and a strongly deterministic measurement pattern with n qubits, then the circuit outputted by algorithm 5 has n-k wires.*

Proof. First we note that by the subroutine 6, for any $u \in domian(f)$ and $v = f(u)$, both u and v will be put on the same wire, as we create a wire for all qubits in the pattern but each element of image of f (since they will be put on already existing wires, which is mapped to its predecessor), which creates $n - |image(f)|$ wires overall. As such, it is sufficient to show that f is injective, so size of the image is equal to size of the domain and equal to k.

By contradiction, let there be $u_1, u_2 \in domain(f)$ such that $f(u_1) = f(u_2)$, but then since u_1 and $f(u_1)$ need to be neighbours and similarly u_2 and $f(u_2)$, by the definition of partial causal flow we have $u_1 \prec u_2$ (since we have u_1 smaller than any neighbour of $f(u_1)$) and similarly $u_2 \prec u_1$ so f is not a valid partial causal flow and hence contradiction. □

We now define the directed graph G' later used for lower bounding the size of partial causal flow.

Definition 3.2.1. *A directed graph G' of a measurement pattern $\mathcal{P}$ with qubit set V and set of inputs and outputs, I and O respectively, is a graph created in the following way.*

1. *create a node for each qubit in the pattern, $V(G') = V$*
2. *initially add two edges (one per direction) for each of the entangle operation, that is, $\forall E_{ij} \in \mathcal{P}$, $(i,j),(j,i) \in E(G')$*
3. *for all corrections C_t^s ($C = \{X, Z\}$) if $dist_G(t,s) < 3$ add a new node $a_{t,s}$ and edge $(t, a_{t,s})$, furthermore $\forall n \in V$ add edge $(a_{t,s}, n)$*

Theorem 3.2.4. *Let S be an distance 3 independent set such that $|S| > 1$, then there exist a partial causal flow function f on the original graph G (open graph of $\mathcal{P}$) such that $domain(f) = S \setminus (O \cup ON)$, where O is set of outputs for pattern $\mathcal{P}$ and $ON = \{u \mid \forall v \in N_G(u), v \in I\}$ is the set of nodes that neighbour only with inputs (note that this also includes all isolated vertices if they exist).*

Proof. The partial causal flow function f can be created by taking for each node of $S \setminus (O \cup ON)$ any neighbour of that node, that is $\forall u \in S \setminus (O \cup ON), f(u) = v$ where $v \in N_{G',out}(v) \setminus I$.

Note that $\forall u \in S \setminus (O \cup ON), N_G(u) \subset N_{G',out}(u)$, furthermore the only nodes $u \notin N_G(u)$ and $u \in N_{G',out}(u)$ are the ancillary nodes created in step 3 of definition 3.2.1, and it is impossible for $N_{G',out}(u)$ to contain only such nodes (node cannot have only edge to $a_{t,s}$ as that would imply no other outgoing edge and $dist_G(t, s) = \infty$ hence $a_{t,s}$ would not exist). Since we also subtract from S all nodes that do not have other neighbours than inputs, there is always a node $f(v) \in N_G(u) \setminus I$ that can be picked. This construction results in $f : V \setminus O \to V \setminus I$ being a valid partial function.

It is now sufficient to show that this construction of f produces a valid partial causal flow. Lets assume by contradiction that the resulting ordering $\prec$ is not valid, that is there are qubits u, v such that $u \prec v$ and $v \prec u$. This can happen in one of the following cases:

1. $v = f(u)$ and $u = f(v)$
2. $v = f(u)$ and $u \in N_G(f(v))$ (or symmetric condition, but without loss of generality we will only consider this case)
3. $v = f(u)$ and $C_u^v \in \mathcal{P}$
4. $v \in N_G(f(u))$ and $C_u^v \in \mathcal{P}$

In the case 1, since both $u, v \in S \setminus (O \cup ON)$ then as already stated $N_G(u) \subset N_{G',out}(u)$ and since $u = f(v)$ then $u \notin I$, so $u \in N_{G',out}(v)$ hence $dist_{G'}(v, u) = 1$ so both u and v cannot be in distance 3 independent set S, contradiction.

In the case 2, we analogously have $v \in N_{G',out}(f(u))$, and $f(u) \in N_{G',out}(u)$ so $dist(u,v) = 2$ and hence both u and v cannot be in distance 3 independent set S, contradiction.

In the case 3, it can be clearly seen that $dist_G(u,v) = 1 < 3$ so $a_{u,v} \in V(G')$, as such, $\forall t \in V(G'), dist_{G'}(u,t) < 3$ hence if $u \in S$ then $|S| = 1$ and hence by condition $|S| > 1$ we arrive at a contradiction.

In the case 4, analogously to the case 3, it can be seen that $dist_G(u,v) < 3$ (since $v \in N_G(f(u))$ and $f(u) \in N_G(u)$) so $a_{u,v} \in V(G')$, as such, $\forall t \in V(G'), dist_{G'}(u,t) < 3$ hence if $u \in S$ then $|S| = 1$ and hence by condition $|S| > 1$ we arrive at a contradiction.

As all cases lead to contradiction, there is no such u, v for which we have both $u \prec v$ and $v \prec u$ so f is a valid partial causal flow on G. □

Combining lemma 3.2.1 and theorem 3.2.4 can give us a desired bound on number of wires of the resulting circuit. Indeed, if there is a distance 3 independent set S satisfying conditions of theorem 3.2.4, then we know that there exist flow of at least the size of $|S|$, that is we know that the k in lemma 3.2.1 satisfies the condition $k > |S|$, so $n - k < n - |S|$ as such we arrive at an upper bound of the number of wires in the resulting quantum circuit. If there is no set S that satisfies the conditions of theorem 3.2.4, then we can still bound the size of partial cause using theorem 3.1.1 (which requirements are reasonable assumptions to be made about a pattern in which there is any information flow) and get $k \geq 1$ which provides the bound of $n - 1$. It is important to mention however, that the bound derived here, specifically the upper bound $n - |S|$ is a guarantee for our extraction method when largest partial causal flow is found, and as already stated, our method of computing partial causal flow does not always returns the largest partial causal flow. It is unknown how to efficiently compute largest partial causal flow.

Furthermore, given the fact that the ancillary qubits in the circuit outputted by algorithm 5 are all measured at some point in time and then no further operations are performed on them, one could ask how many qubits are required to implement a circuit if one can reuse qubits. That is, rather than creating a new wire for another ancillary qubit, reuse a wire that has already been measured, by resetting it (which can

be accomplished on a quantum computer, although it is a relatively time consuming process) back to the state $|+\rangle$ and treating it as a new wire.

In that case, given a partial causal flow $(f, \prec)$ one can bound number of qubits with the help of the parameter called cutwidth mentioned in section 2.1.2. We present two versions of the bound for the circuit with classical control and for the circuit where all corrections are implemented by quantum controlled gates.

Theorem 3.2.5. *Given a measurement pattern $\mathcal{P}$ with partial causal flow $(f, \prec)$ and open graph G, if one allows for reusing qubits, the circuit outputted by algorithm 5, with classical control allowed, can be implemented using $2cutwidth_{\prec}(G)$, where $cutwidth_{\prec}(G)$ is cutwidth of the graph G as defined in section 2.1.2, however with the ordering of edges π constrained to the partial order $\prec$, that is, for any $u \prec v$ we have $\pi(u) < \pi(v)$.*

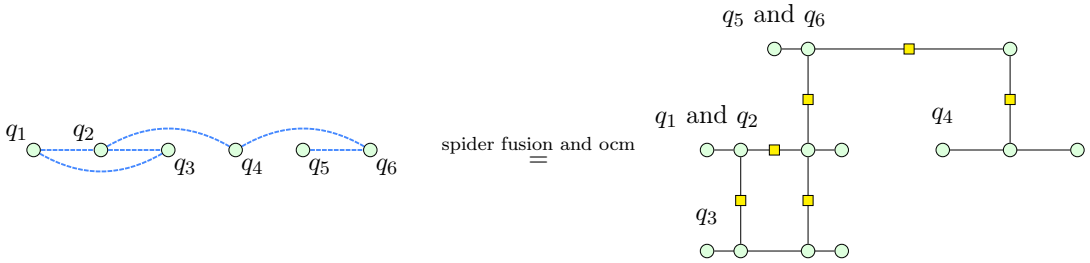
Proof. The ordering π corresponds to an linear arrangement of qubits in the measurement pattern, these can be transformed according to the method shown in the proof of theorem 3.2.2. Note that cutwidth for that ordering (without changing the ordering of nodes), that is $cutwidth_{\prec}(G)$ corresponds to number of edges going from wires in the past to wires in the future, as such if the gates are made causal (by extending the wires and using only connectivity matters to make controlled-Z gates execute at the same time) it corresponds to half of number of wires needed at any point (this is an upper bound as some of the entangle operations represented by edges may be implemented as Hadamard operations, hence not requiring two wires to be implemented). Once all multi-qubit operations are implemented on wires, only single qubit operations are left, which can be implemented "in a single time frame" (since all other wires can just be made to "wait" for the single-qubit operations to end) and the wire is free to be reused by another qubit.

Furthermore, since π produced ordering that abides by $\prec$, then $f(u)$ is after u, and so are all of its neighbours, allowing to put u and $f(u)$ on the same wire.

As such $2cutwith_{\prec}(G)$ is an upper bound on how many wires are necessary at any given time, and any qubits that do not need to be "present" in that time are either already measured or do not need to be initialized yet so their wires either do not yet exist or can be reused.

□

An example of this transformation, for reusing wires based on linear arrangement, can be seen below:



Theorem 3.2.6. *Given a measurement pattern $\mathcal{P}$ with partial causal flow $(f, \prec)$ and open graph G, if one allows for reusing qubits, the circuit outputted by algorithm 5 can be implemented using $2cutwidth_{\prec}(G_c)$, where $cutwidth_{\prec}(G_c)$ is cutwidth of the graph G_c as defined in section 2.1.2, however with the ordering of edges π constrained to the partial order $\prec$, that is, for any $u \prec v$ we have $\pi(u) < \pi(v)$ and G_c is a graph with $V(G_c) = V(G)$, $E(G_c) = E(G) \cup \{(u, v) \mid C_u^v \in \mathcal{P}, C = X \vee C = Z\}$, that is graph with additional edges added for each correction (if the edge is not already there).*

Proof. The proof in this case is the same as of the bound in theorem 3.2.5, with the additional note that since we are implementing control using quantum gates, both qubits have to also exist during correction. As such we amend the open graph by adding edges (s, t) for any correction C_t^s (if the edge does not already exist), in that case we ensure that cutwidth finds the point in which the most qubits are needed at the same time and then the argument from the theorem 3.2.5 applies. □

3.3 Evaluation

We conclude by giving some general remarks about the algorithm and its theoretical properties presented in this chapter.

Firstly, it is worth mentioning that this algorithm combines the original star pattern translation method presented by Danos et al [10] and the general purpose method presented by Broadbent et al [5]. Moreover, based on the size of the partial causal flow, our

method resembles either of these methods. That is, if there is flow in the open graph of input measurement pattern and partial flow finding algorithm finds that flow (which is always the case using algorithm 2) then the resulting circuit outputted by algorithm 5 results in the same circuit as star pattern translation would (and in a sense performs the same operations). And, if the partial causal flow that was found is relatively small, the outputted circuit will resemble more the output of the general purpose method.

The method presented here has two important advantages over methods like star pattern translation [10], star pattern translation for graphs with gflow [28], circuit extraction from ZX diagrams [2] etc. Firstly, this method works for a much broader class of patterns, correctness proof works for all strongly deterministic patterns, not just robust deterministic patterns as is the case other mentioned methods. Furthermore, while there is no proof as to behaviour of this algorithm for non strongly deterministic patterns, we know that it terminates and outputs a circuit and there are some hopes that for a "sensible" non deterministic patterns the circuits may be correct. Sadly, due to lack of time and proper definition of "sensible" this topic is not further examined in this work. Secondly, our method provides a theoretical bound on number of two qubit gates, something that most methods for extracting circuits for patterns with gflow, like the ones by Miyazaki et al and Backens et al do not provide [2, 28].

4 Experiments

4.1 Experimental setup

In order to evaluate our methods, we compare their performance to other algorithms presented in the literature by evaluating number of qubits and two-qubit gates in circuits extracted from randomly generated measurement patterns. However, computing corrections for an arbitrary measurement pattern, to our knowledge, is not a solved problem and most likely is computationally hard. As such, we generate our test dataset by generating a random quantum circuit consisting of Clifford and T gates (which form an universal gateset) using a method provided by python library PyZX [25]. We then apply optimization method introduced by Duncan et al [13], which modifies the circuit into a ZX diagram that possibly does not have causal flow, but guarantees existence of generalized flow. We then compute delayed gflow on the resulting ZX diagram to determine corrections for the measurement pattern represented by the resulting ZX diagram.

Thanks to this approach we can generate random measurement patterns with known corrections and evaluate our methods on it. Furthermore, since the open graphs of measurement patterns have generalized flow, we can compare our method not only against general purpose methods but also against extraction methods that work only on patterns with gflow.

The downside of this approach is that we do not have direct control over the size of the pattern which will depend on the random circuit generated by PyZX. We do however have control of number of qubits in the random circuit and its depth, as such, we vary those parameters in the experiments presented here to present a more holistic picture. As it will be later seen, we make the depth of the circuit dependent on number

of qubits, this makes it more likely that an "interesting" circuit (with two-qubit gates) is generated.

4.2 Empirical results

4.2.1 Comparison of partial flow finding algorithms

Firstly, we compare the performance of algorithms 1 (with an arbitrary edge ordering) and 2 by comparing the size of the resulting partial causal flow. We generate our test dataset for circuits with 5, 10, 20, 30, 40 and 50 qubits and their depth ranging from $10 + n$, where n is the number of qubits in the circuit, to $300 + n$, and for each size generating 10 circuits. The results can be seen in figure 4.1 As it can be seen, there does not seem to be a significant difference between the two algorithms. However, it does seem like there is slightly more variance in size of partial causal flow returned algorithm 2 then the size of partial causal flows returned by greedy approach. Due to this, for the following experiments we will present results for both approaches of computing partial causal flow, referring to the algorithm 1 as greedy and to algorithm 3 as delayed (referring to the fact that it has been inspired by algorithm that finds maximally delayed causal flow).

4.2.2 Number of qubits in the circuit

We now evaluate number of qubits in the extracted circuit. We compare our approach against a general purpose method introduced by Broadbent et al [5] that can be applied to any pattern, without any restrictions. Using the fact that our dataset has gflow, we also compare our method to domain specific approach, in this case we use PyZX extract circuit method, which is a more optimized version of an ZX circuit extraction approach presented by Backens et al [2].

We present our results for dataset generated out of circuits with 5, 10, 20, 30, 40 and 50 qubits and depth ranging from $10 + n$ to $300 + n$ where n is the number of qubits, and for each size of the circuit we generate 5 circuits.

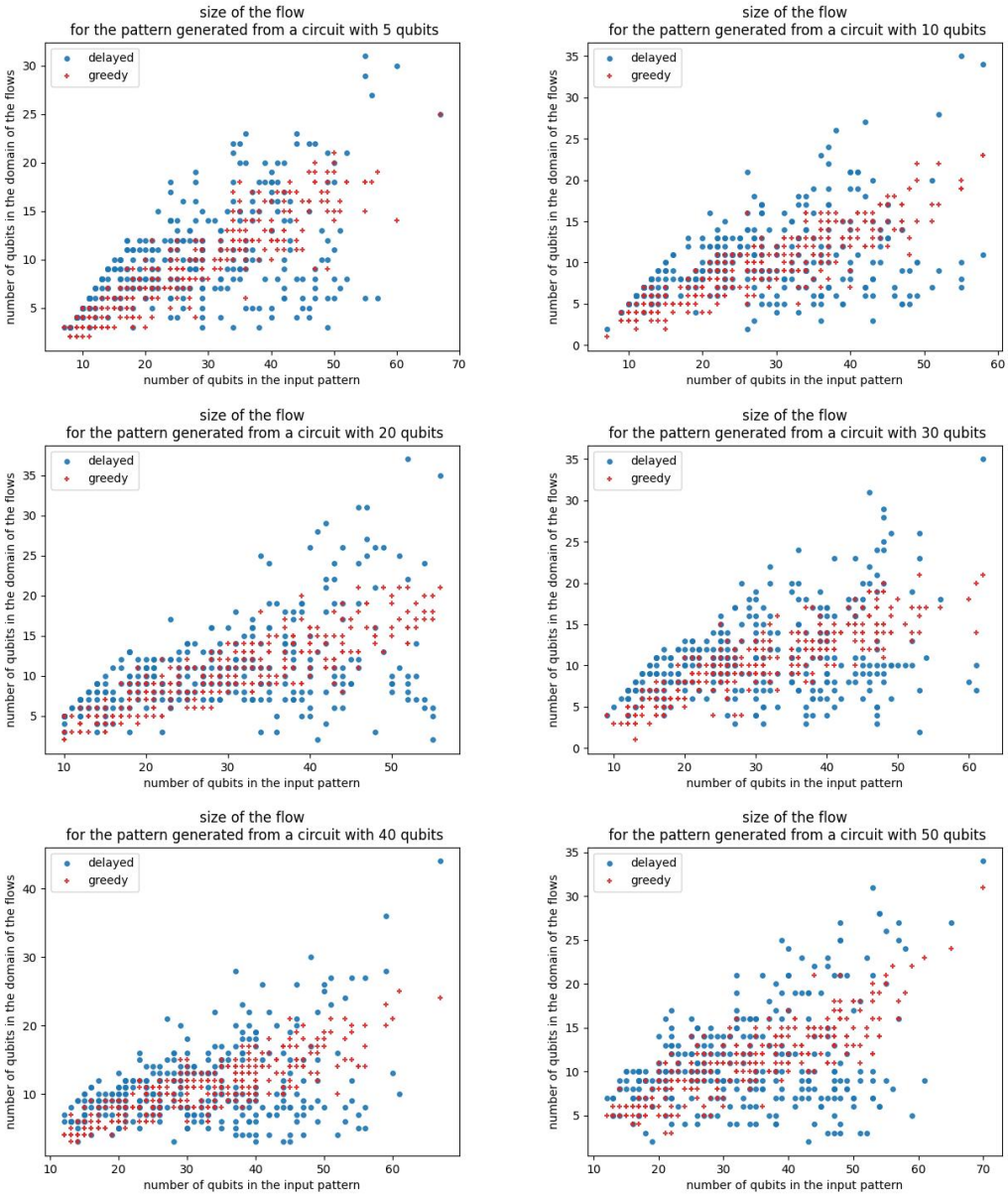


Figure 4.1: Comparison of sizes of partial causal flows (number of qubits in domain) produced by algorithms 1 (referred to as greedy) and 2 (referred to as delayed)

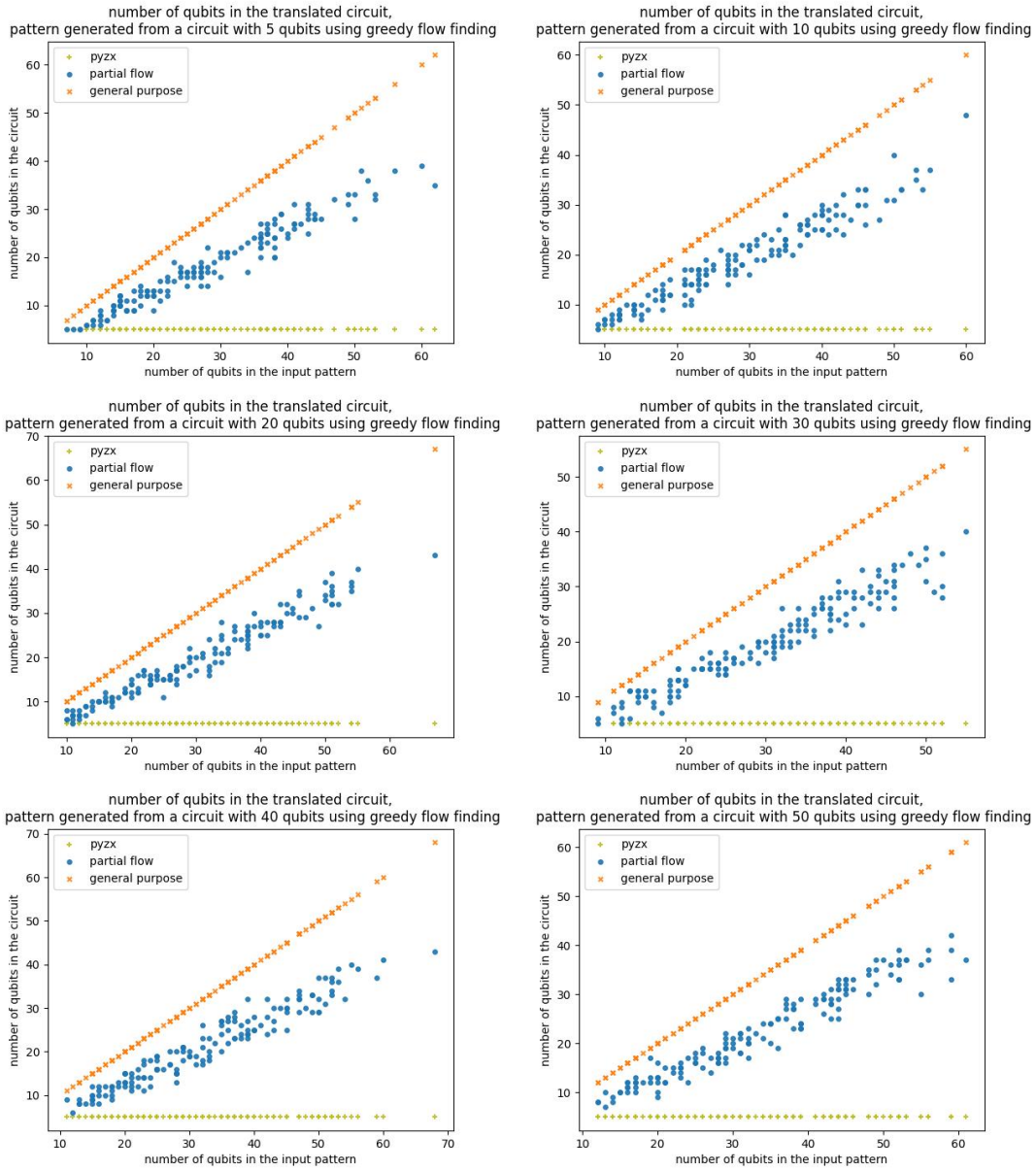


Figure 4.2: Number of qubits in the translated circuit for general purpose method [5], pyzx extract circuit method [25] and our method, using greedy flow finding

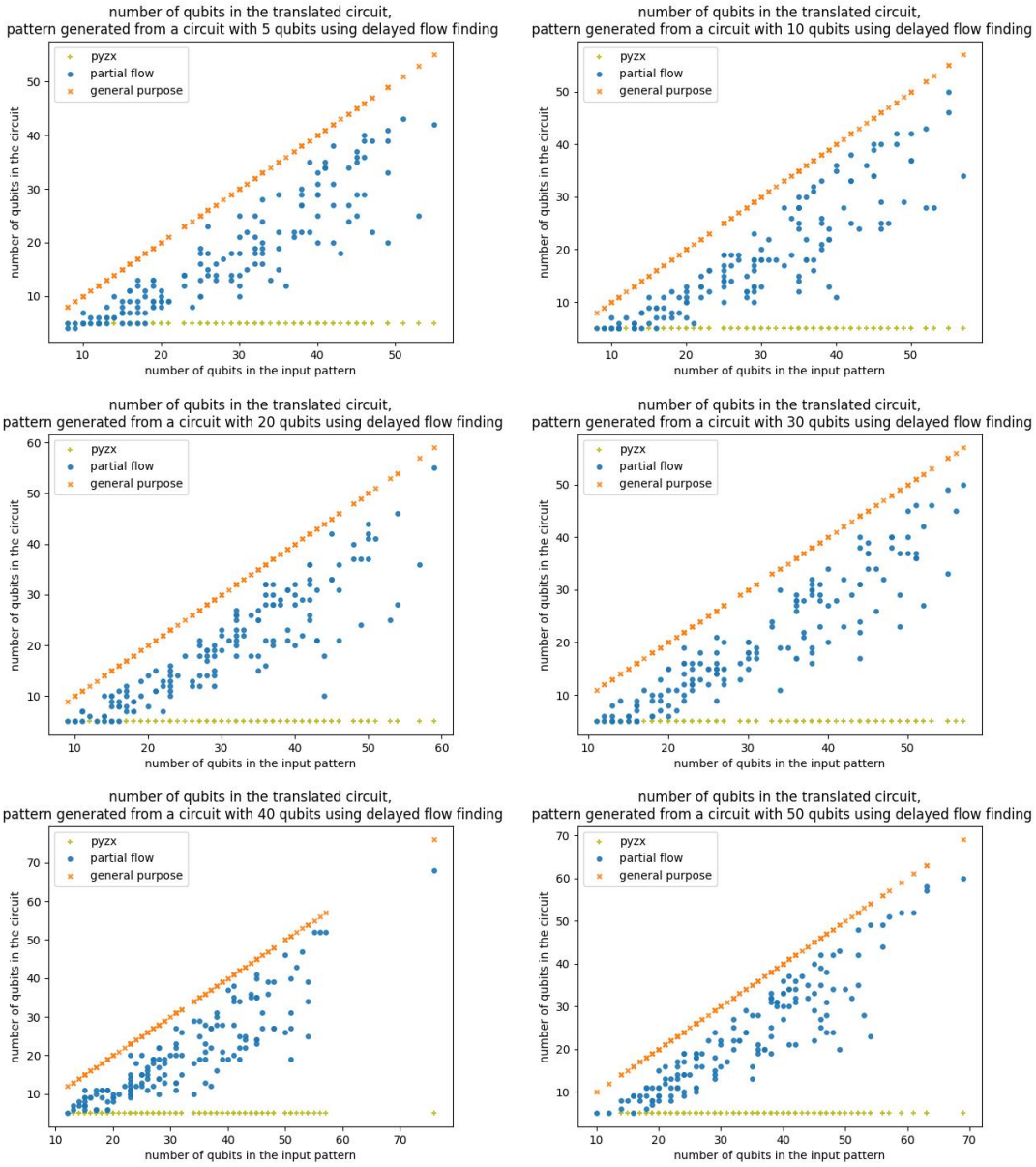


Figure 4.3: Analogous experiment to the one in figure 4.2 but with partial causal flow computed by delayed flow finding method

Results of our experiments can be seen in figure 4.2 for the greedy approach of finding partial causal flow and in figure 4.3 for the delayed approach of finding partial causal flow. Unsurprisingly, our method results in circuits larger than the domain specific efficient extraction methods for patterns with generalized flow. However, it reduces the number of qubits in the circuit compared to the general purpose method in some cases by over 50%.

An interesting fact is that our method can find circuits that are slightly smaller than pyzx extraction method. This can be a result of our implementation, pyzx keeps boundary nodes that serve as output nodes while in our method we remap those boundary points onto their Z spider neighbours, this sometimes (given that an optimization routine is run to remove flow from the graph) may decrease the number of inputs and hence make it possible for our method to find circuit with less wires.

4.2.3 Tightness of the bound

We now turn to the evaluation of the estimation method presented in section 3.2.4. Due to computational expensiveness of that method, we reduce the depth of the circuit to ranging from $n + 10$ to $n + 150$, where n is the number of qubits which is either 5, 10, 15 or 20 and for each size of the circuit we generate 10 circuits. Here we present the results as proportional overhead over our extraction method, that is, for estimation (and general purpose method provided as reference) we take the result of estimation, subtract the result produced by our approach and then divide that result by result produced by our approach.

Our results can be seen in figure 4.4 for the greedy flow finding approach and in figure 4.5 for the delayed flow finding approach (which is used to compute the baseline circuit, that is our method).

These experiments further confirm that our method of finding partial causal flow does not find the largest partial causal flow as can be seen from some of the negative values in the plots. It has been verified that the independent sets returned in that case produce valid partial causal flow. As such, the bound provided shows that there exist an algorithm which can accomplish these results, but it is not the partial flow finding

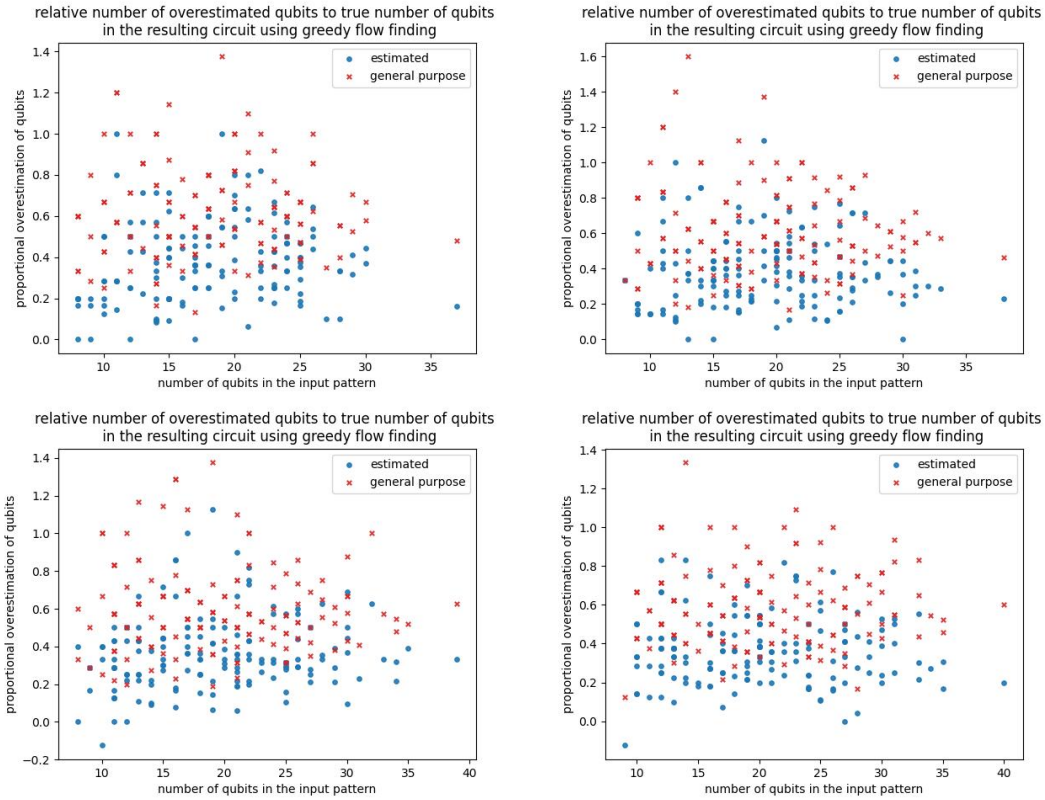


Figure 4.4: Relative number of qubits estimated and in the general pattern to the number of qubits of the circuit returned by our method, calculated as $\frac{\text{number of qubits estimated} - \text{number of qubits returned by our method}}{\text{number of qubits returned by our method}}$ and analogously for general purpose method $\frac{\text{number of qubits returned by general purpose method} - \text{number of qubits returned by our method}}{\text{number of qubits returned by our method}}$, calculated using greedy flow finding approach

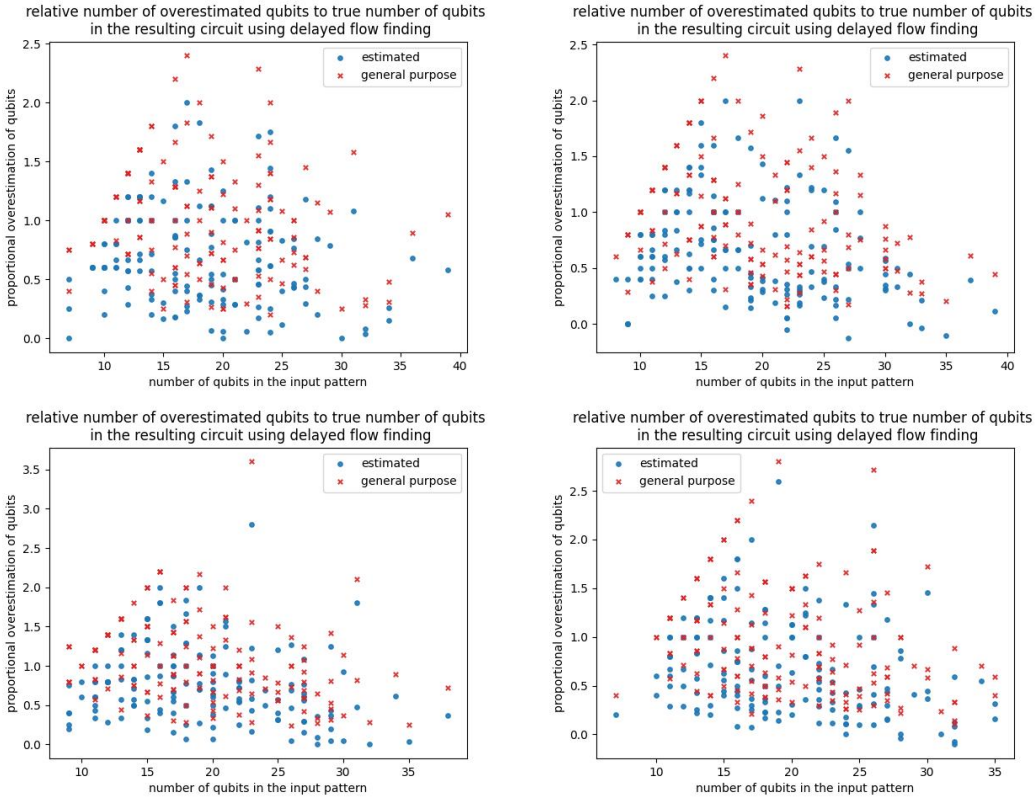


Figure 4.5: Analogous experiment to the one in figure 4.4 but with partial causal flow computed by delayed flow finding method

algorithm provided in this project. Furthermore, it is unknown if a polynomial method to compute largest partial causal flow exists.

4.2.4 Number of two-qubit gates in the circuit

Another important aspect of circuit extraction is the number of two-qubit gates in the translated circuit. Figures 4.6 and 4.7 display the relative proportional amount of two-qubit gates as compared to the baseline approach, that is the general purpose method presented by Broadbent et al [5]. Our experimental setup in this case is analogous to the experimental setup for examining number of qubits in the translated circuit with the exception that we present number of two-qubit gates in circuits extracted by our method with and without classical control, by PyZX extract circuit method and by extension of the general purpose approach to allow for classical control. We include classical control only in experiments that examine number of two-qubit gates as that is the only parameter that changes in that approach (since classical control allows us to implement some of two-qubit gates as classically controlled gates).

Firstly, it can be seen that our approach based on partial causal flow in most cases out performs the baseline approach while also generating circuits with less qubits. That being said, classical control approaches seem to perform incredibly well by reducing the number of two-qubit gates by over 40% with our partial flow method with classical control outperforming general purpose method with classical control and even in some cases the domain specific PyZX circuit extraction method (when using delayed flow finding method), by quite a significant margin.

4.2.5 Trade-off between number of qubits and number of two-qubit gates

In this work the main two qualities of the translated circuits that are compared are number of two-qubit gates and number of qubits in the circuit. In order to provide the full picture of how the two interact with each other, we present experiment that for each extracted circuit plots relative proportion of two-qubits gates against proportion of qubits, in both cases compared to the circuit extracted by general purpose method.

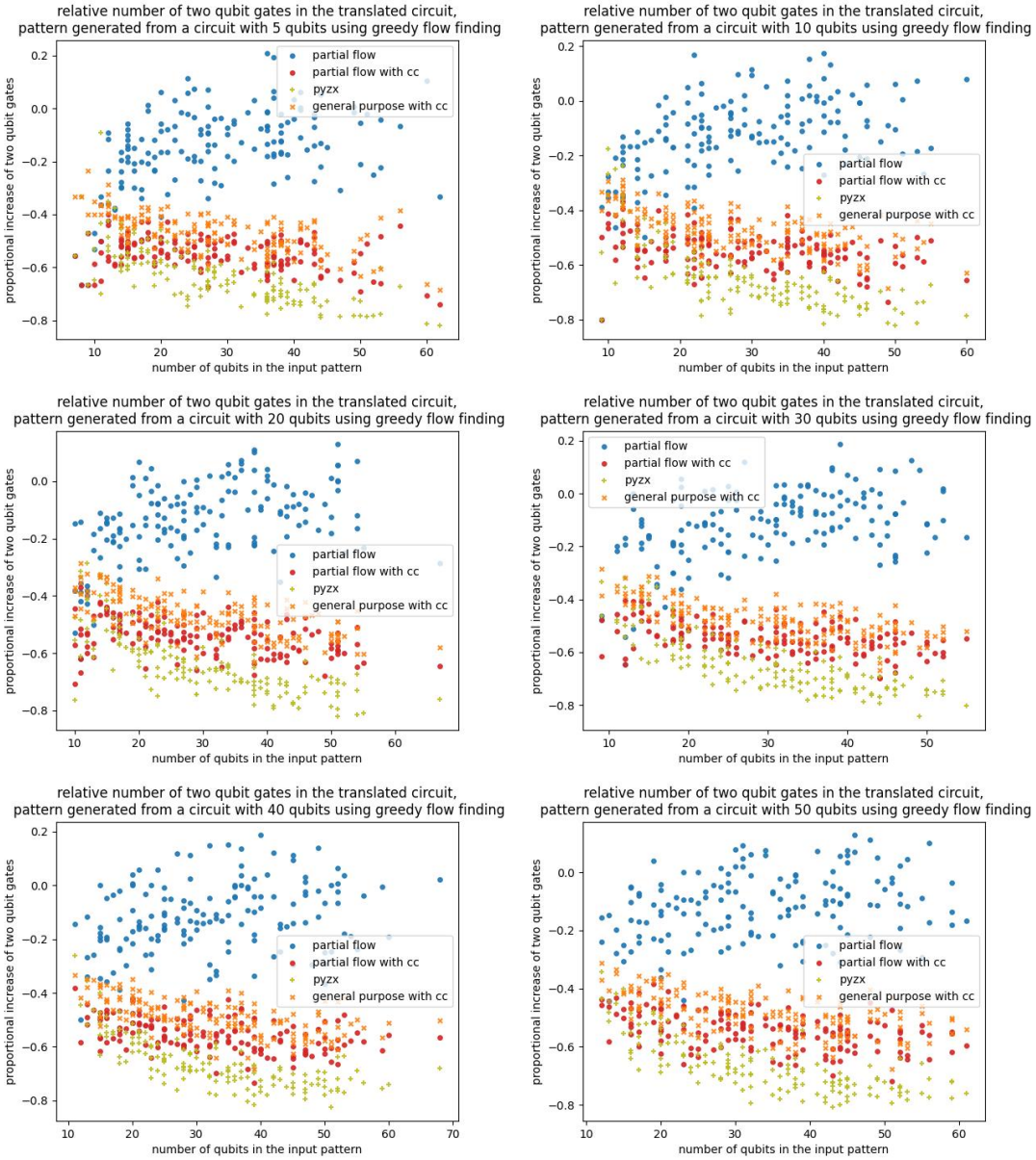


Figure 4.6: Relative number of two qubit gates in circuits outputted by PyZX extraction method, our method, classical control extension of it and classical control extension of general purpose method (both classical control extensions denoted as "with cc"), calculated as $\frac{\text{num two qubit gates returned by algorithm} - \text{num two qubit gates returned by general purpose method}}{\text{number of two qubit gates returned by general purpose method}}$. Calculated using greedy flow finding approach

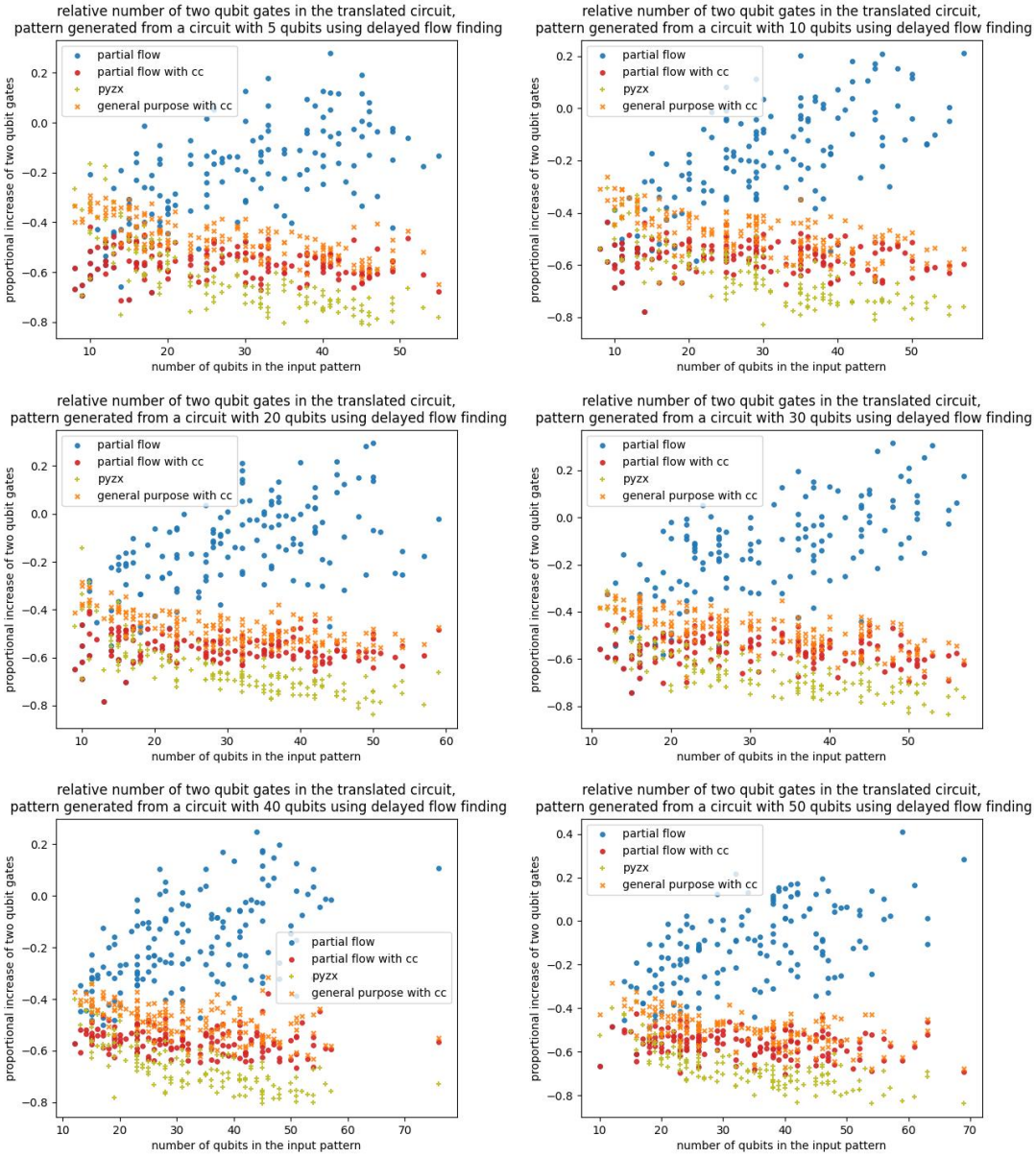


Figure 4.7: Analogous experiment to the one in figure 4.6 but with partial causal flow computed by delayed flow finding method

We use the same experimental setup as in the case of number of qubits and number of two-qubit gates experiments.

The results of our experiment can be seen in figures 4.8 and 4.9. This data further confirms that our method with classical control significantly outperforms the baseline and when delayed flow finding approach is used, can even, at times, perform better or on par with domain specific approaches.

4.2.6 Conclusion

To summarize, empirical evaluation of our method shows that it provides improvement over the general purpose method when it comes to number of qubits and in most cases also number of two-qubit gates. When allowing for classical control, our method significantly improves number of two qubit gates over the baseline and in some cases performs on par or even outperforms method designed specifically with graphs that admit gflow in mind. This is highly encouraging as our method works on much larger family of measurement patterns than the ones with underlying gflow.

Sadly, it can be seen that our approach to computing partial casual flow does not significantly outperform greedy approach and in some cases is unable to find flows larger than our estimation method, which should not, in theory give a very tight bound.

Overall, our method performs really well, especially given the generality of it and gives high hopes for future applications of this method with better flow finding approaches (as size of the domain of the partial flow has inverse correlation with number of qubits and number of two-qubit gates).

4.3 Pattern without generalized flow

We conclude the evaluation of our algorithm by providing an example of a measurement pattern that does not admit generalized flow (or even Pauli flow) yet is strongly deterministic and implements a SWAP operation. This pattern has been shown by Raussendorf et al [36] and then later mentioned by Browne et al [6]. Sadly, neither of them have provided corrections that need to be applied, as such we derived them by

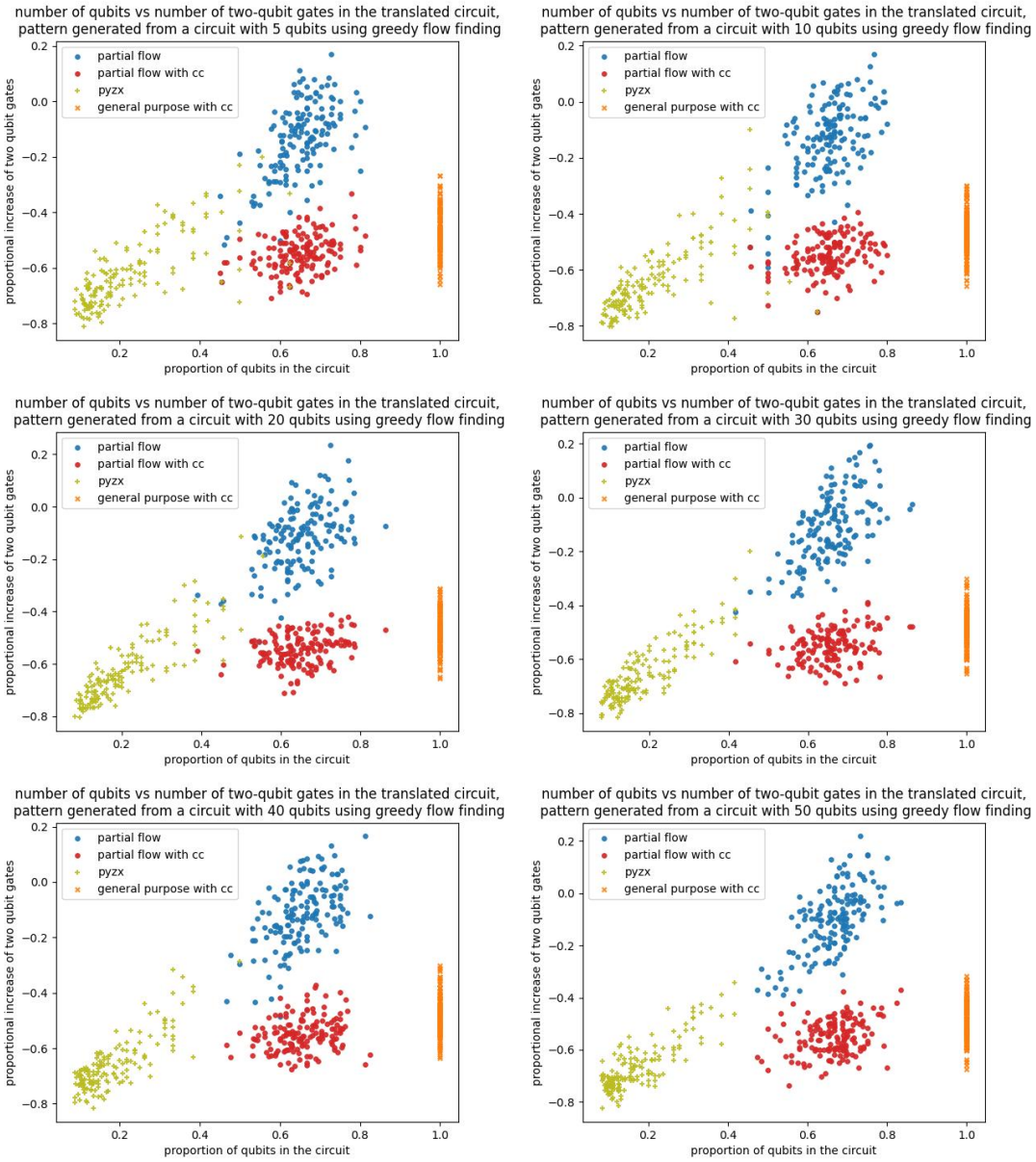


Figure 4.8: Proportional increase in number of two-qubit gates (calculated similarly as in the case of figure 4.6) plotted against proportion of number of qubits in the extracted circuit by given method to number of qubits in the extracted circuit by general purpose method (which is equal to the size of the pattern). Calculated using greedy flow finding method

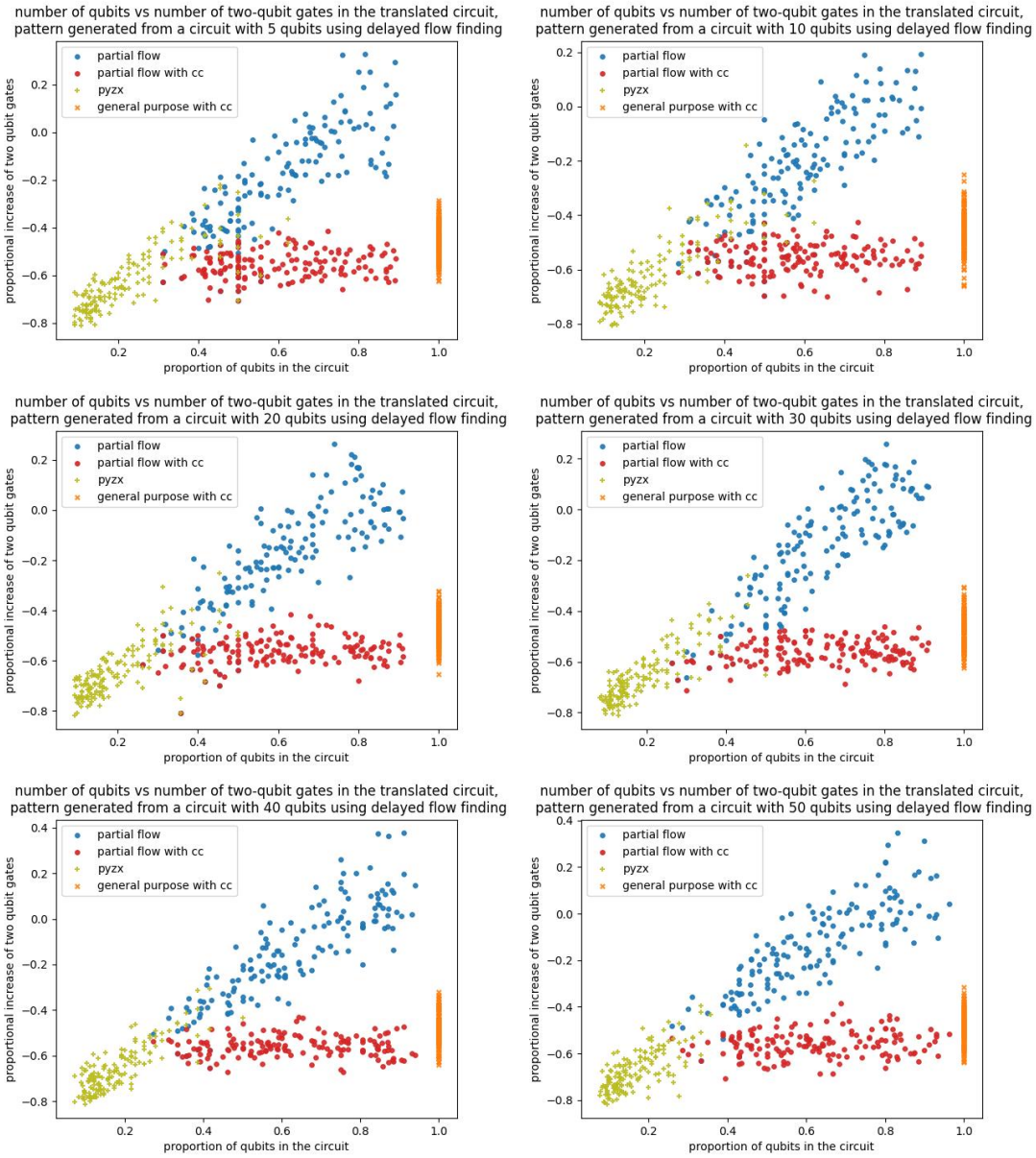


Figure 4.9: Analogous experiment to the one represented by figure 4.8 but using delayed flow finding

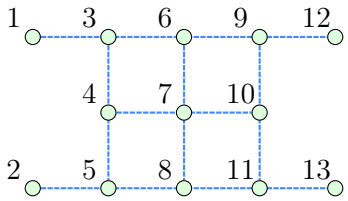


Figure 4.10: Open graph of the measurement pattern implementing SWAP operation that has no generalized flow (or Pauli flow)

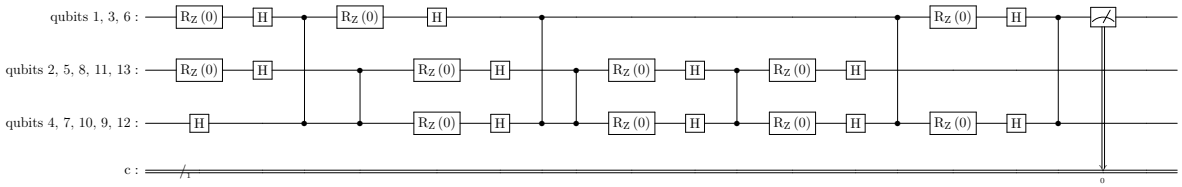


Figure 4.11: circuit without classical control extracted out of the SWAP measurement pattern

hand. The pattern consists of 13 qubits with inputs $I = \{1, 2\}$ and outputs $O = \{12, 13\}$ and is defined as follows:

$$\mathcal{P} = X_{12}^{9} X_{13}^{11} Z_{12}^{6} X_{12}^{7} X_{13}^{7} Z_{13}^{8} X_{13}^{3} Z_{13}^{4} Z_{12}^{4} X_{12}^{5} Z_{13}^{1} Z_{12}^{2} M_{11}^{0} \dots M_{1}^{0} E_{11,13} \dots E_{1,3} N_{13} \dots N_{3}$$

The open graph of the pattern, that fills in the missing edge definition from the pattern, can be seen in figure 4.10 (all nodes are measured at a 0 angle) and the extracted circuit can be seen in figure 4.11, and the version with classical control can be seen in figure 4.12.

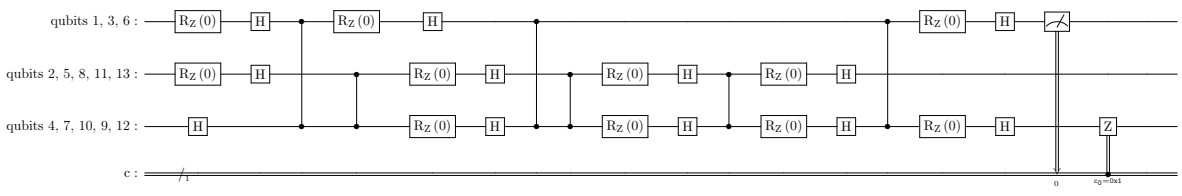


Figure 4.12: circuit with classical control extracted out of the SWAP measurement pattern

We note that, to our knowledge, the only other methods that are capable of extracting circuits out of this measurement pattern are the ones introduced by Broadbent et al [5] and they would require using at least (depending on the specific method) 13 qubits, while our approach finds a circuit with 3 qubits.

5 Discussion

We conclude this thesis by discussing some possible applications of the work done here and by providing some directions on how this work can be extended. Furthermore, we reflect on the results achieved throughout this project and their usefulness.

5.1 Applications

Beyond the obvious application of translating measurement patterns so that they can be executed as quantum circuits, and as such, on hardware which uses quantum circuit model of computation, work done in this thesis can also be applied in other ways.

The main application that this project was done with mind in, is quantum circuit optimization. In particular, current methods of circuit rewrites with ZX calculus are constrained to the ones that preserve existence of generalized flow [13] so that one can extract the circuit out of the resulting ZX diagram. Our method allows for any rewrite rules to be used, if one starts with a quantum circuit, the underlying pattern is strongly deterministic and any rewrite rule will maintain that property (since it will rewrite the pattern to an equivalent one). As such, if one is able to keep track how correction change (computing corrections for the pattern created from a circuit is not hard as, before any rewrite rules are applied, it will have generalized flow) with each application of a rewrite rule, one can then extract the resulting circuit even if the resulting ZX diagram has no generalized flow. This unlocks a wide range of rewrite rules and most likely, it is possible to keep track of how the corrections change for all of the 4 basic rules of ZX-calculus, which implies that then any rewrite rule would be applicable.

5.2 Future work

The first area that would be interesting to explore and would have potentially large impact on the performance of extraction method shown in this work is further understanding partial causal flow and how to compute it. Clearly, finding an algorithm that would be guaranteed to compute largest partial causal flow would be ideal, that being said even algorithm which always finds causal flow larger than the bound proposed here would already be a significant improvement as the larger the partial causal flow the less ancillary qubits (and less two-qubit gates) are in the resulting circuit.

Another approach to improve the performance of this algorithm is to investigate rewrite rules that are used to make reduce number of ancillary qubits from the general purpose method, such approaches have been used to provide quantum circuits without ancillary qubits for patterns with generalized flow by Dias da Silva et al. [39] and the methods there could potentially be used in our setting (as J gates are essentially Z rotation gates with a Hadamard gate applied after, that is exactly what appears in this work).

Furthermore, in this work we study partial causal flow, but in principle one could study partial generalized flow and apply similar method of extraction. In general a method of finding partial generalized flow is an easy extension of the algorithm shown here and the gflow finding algorithm by Mhalla et al. [27]. The extraction method is also a relatively simple extension of work done by Miyazaki et al. [28] and it seems that the primary challenge in this extension is scheduling two-qubit gates appropriately and showing that the resulting circuit is correct (the original method has a significant amount of graph rewiring which makes this less clear than in the case of partial causal flow).

Additionally, throughout this work a question was posed about hardness of computing corrections for an arbitrary measurement pattern (which does not have corrections provided) to make the pattern deterministic. It is conjectured that this is computationally hard, however should that not be the case, ability of computing corrections for an arbitrary pattern could prove incredibly useful as such making this an important question.

Finally, the possibly most important extension of this work is to develop optimization routines, as described in section 5.1 that apply the work done in this thesis and examine their performance.

5.3 Conclusion

In this work we provided a novel approach to extracting quantum circuits out of strongly deterministic measurement patterns. Our method works on all strongly deterministic measurement patterns which is a much larger family of patterns than what other similar work in the field focuses on. At the same time our method is more resource efficient and produces smaller number of two-qubit gates on average than the general purpose methods for translating measurement patterns to quantum circuits. Furthermore, we provide detailed theoretical analysis of our algorithm and the general approach introduced in this upper bounding important properties of the algorithm like number of qubits in the outputted circuit and number of two-qubit gates.

That being said, while our bound on number of qubits works for the case in which largest partial causal flow is used for extraction, partial flow finding algorithm shown in this work does not find largest partial causal flow. Furthermore, as it has been seen in the experiments provided in this work, in some cases our estimation method estimates larger partial causal flows than our flow finding method. As such, the bound does not work for the exact method provided in this work but only the general approach. Moreover, computing that bound is computationally hard, while our method of computing partial causal flow takes polynomial amount of time, hence the use cases of that theoretical result are limited.

On the other hand, as it can also be seen from empirical results our method performs significantly better than the general purpose method and in some cases even outperforms more specialised algorithms. As such work done in this thesis provides meaningful contribution to the field of circuit extraction filling a void in extraction algorithms that would work on a broader class of patterns than just robustly deterministic patterns without being extremely resource inefficient. Furthermore, we provide additional un-

derstanding to the resource trade-off in translating measurement patterns to quantum circuits and many future directions that can be explored to improve the performance of this method and apply it to various tasks like optimization of quantum circuits.

Bibliography

1. C. Adami and N. J. Cerf. *Quantum computation with linear optics*. 1998. arXiv: quant-ph/9806048 [quant-ph].
2. M. Backens, H. Miller-Bakewell, G. de Felice, L. Lobski, and J. van de Wetering. "There and back again: A circuit extraction tale". *Quantum* 5, 2021, p. 421. DOI: 10.22331/q-2021-03-25-421. URL: https://doi.org/10.22331%2Fq-2021-03-25-421.
3. N. de Beaudrap. *Unitary-circuit semantics for measurement-based computations*. 2009. arXiv: 0906.4261 [quant-ph].
4. N. de Beaudrap, A. Kissinger, and J. van de Wetering. "Circuit Extraction for ZX-Diagrams Can Be #P-Hard". en. In: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. DOI: 10.4230/LIPICS.ICALP.2022.119. URL: https://drops.dagstuhl.de/opus/volltexte/2022/16460/.
5. A. Broadbent and E. Kashefi. "Parallelizing quantum circuits". *Theoretical Computer Science* 410:26, 2009, pp. 2489–2510. DOI: 10.1016/j.tcs.2008.12.046. URL: https://doi.org/10.1016%2Fj.tcs.2008.12.046.
6. D. E. Browne, E. Kashefi, M. Mhalla, and S. Perdrix. "Generalized flow and determinism in measurement-based quantum computation". *New Journal of Physics* 9:8, 2007, pp. 250–250. DOI: 10.1088/1367-2630/9/8/250. URL: https://doi.org/10.1088%2F1367-2630%2F9%2F8%2F250.
7. T. Cam and S. Martiel. *Speeding up quantum circuits simulation using ZX-Calculus*. 2023. arXiv: 2305.02669 [quant-ph].

8. F. R. K. Chung. "On the Cutwidth and the Topological Bandwidth of a Tree". *SIAM Journal on Algebraic Discrete Methods* 6:2, 1985, pp. 268–277. DOI: 10.1137/0606026. eprint: https://doi.org/10.1137/0606026. URL: https://doi.org/10.1137/0606026.

9. B. Coecke and R. Duncan. "Interacting quantum observables: categorical algebra and diagrammatics". *New Journal of Physics* 13:4, 2011, p. 043016. DOI: 10.1088/1367-2630/13/4/043016. URL: https://doi.org/10.1088%2F1367-2630%2F13%2F4%2F043016.

10. V. Danos and E. Kashefi. "Determinism in the one-way model". *Physical Review A* 74:5, 2006. DOI: 10.1103/physreva.74.052310. URL: https://doi.org/10.1103%2Fphysreva.74.052310.

11. V. Danos, E. Kashefi, and P. Panangaden. "Parsimonious and robust realizations of unitary maps in the one-way model". *Physical Review A* 72:6, 2005. DOI: 10.1103/physreva.72.064301. URL: https://doi.org/10.1103%2Fphysreva.72.064301.

12. V. Danos, E. Kashefi, and P. Panangaden. *The Measurement Calculus*. 2007. arXiv: 0704.1263 [quant-ph].

13. R. Duncan, A. Kissinger, S. Perdrix, and J. van de Wetering. "Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus". *Quantum* 4, 2020, p. 279. DOI: 10.22331/q-2020-06-04-279. URL: https://doi.org/10.22331%2Fq-2020-06-04-279.

14. R. Duncan and S. Perdrix. "Rewriting measurement-based quantum computations with generalised flow". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2010, pp. 285–296.

15. R. P. Feynman. "Simulating physics with computers". *International Journal of Theoretical Physics* 21:6, 1982, pp. 467–488. ISSN: 1572-9575. DOI: 10.1007/BF02650179. URL: https://doi.org/10.1007/BF02650179.

16. D. Gottesman and I. L. Chuang. "Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations". *Nature* 402:6760,

1999, pp. 390–393. DOI: 10.1038/46503. URL: https://doi.org/10.1038%2F46503.

17. L. K. Grover. *A fast quantum mechanical algorithm for database search*. 1996. arXiv: quant-ph/9605043 [quant-ph].

18. H. Haffner, C. Roos, and R. Blatt. "Quantum computing with trapped ions". *Physics Reports* 469:4, 2008, pp. 155–203. DOI: 10.1016/j.physrep.2008.09.003. URL: https://doi.org/10.1016%2Fj.physrep.2008.09.003.

19. E. Hazan, A. Ménard, I. Ostojic, and M. Patel. *The next tech revolution: Quantum Computing*. 2020. URL: https://www.mckinsey.com/fr/our-insights/the-next-tech-revolution-quantum-computing.

20. D. A. Hoang. "On the Complexity of Distance-d Independent Set Reconfiguration". In: *WALCOM: Algorithms and Computation*. Springer Nature Switzerland, 2023, pp. 254–266. DOI: 10.1007/978-3-031-27051-2_22. URL: https://doi.org/10.1007%2F978-3-031-27051-2_22.

21. M. Houshmand, M. Houshmand, and J. F. Fitzsimons. "Minimal qubit resources for the realization of measurement-based quantum computation". *Physical Review A* 98:1, 2018. DOI: 10.1103/physreva.98.012318. URL: https://doi.org/10.1103%2Fphysreva.98.012318.

22. A. Joshi, A. Kairali, R. Raju, A. Athreya, R. M. P, S. Vishwakarma, and S. Ganguly. *Quantum Circuit Optimization of Arithmetic circuits using ZX Calculus*. 2023. arXiv: 2306.02264 [cs.ET].

23. A. Kissinger and B. Coecke. "Picturing Quantum Processes", 2015.

24. A. Kissinger and A. M.-v. de Griend. *CNOT circuit extraction for topologically-constrained quantum memories*. 2019. arXiv: 1904.00633 [quant-ph].

25. A. Kissinger and J. van de Wetering. "PyZX: Large Scale Automated Diagrammatic Reasoning". *Electronic Proceedings in Theoretical Computer Science* 318, 2020, pp. 229–241. DOI: 10.4204/eptcs.318.14. URL: https://doi.org/10.4204%2Feptcs.318.14.

26. A. Kissinger and J. van de Wetering. "Simulating quantum circuits with ZX-calculus reduced stabiliser decompositions". *Quantum Science and Technology* 7:4, 2022, p. 044001. DOI: 10.1088/2058-9565/ac5d20. URL: https://doi.org/10.1088%2F2058-9565%2Fac5d20.

27. M. Mhalla and S. Perdrix. "Finding Optimal Flows Efficiently". In: *Automata, Languages and Programming*. Springer Berlin Heidelberg, 2008, pp. 857–868. DOI: 10.1007/978-3-540-70575-8_70. URL: https://doi.org/10.1007%2F978-3-540-70575-8_70.

28. J. Miyazaki, M. Hajdušek, and M. Murao. "Analysis of the trade-off between spatial and temporal resources for measurement-based quantum computation". *Physical Review A* 91:5, 2015. DOI: 10.1103/physreva.91.052302. URL: https://doi.org/10.1103%2Fphysreva.91.052302.

29. K. F. Ng and Q. Wang. *A universal completion of the ZX-calculus*. 2017. arXiv: 1706.09877 [quant-ph].

30. K. F. Ng and Q. Wang. *Completeness of the ZX-calculus for Pure Qubit Clifford+T Quantum Mechanics*. 2018. arXiv: 1801.07993 [quant-ph].

31. M. A. Nielsen and I. L. Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.

32. M. A. Nielsen. "Optical Quantum Computation Using Cluster States". *Physical Review Letters* 93:4, 2004. DOI: 10.1103/physrevlett.93.040503. URL: https://doi.org/10.1103%2Fphysrevlett.93.040503.

33. S. Pemmaraju and S. Skiena. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Cambridge University Press, 2003. DOI: 10.1017/CBO9781139164849.

34. Qiskit contributors. *Qiskit: An Open-source Framework for Quantum Computing*. 2023. DOI: 10.5281/zenodo.2573505.

35. R. Raussendorf and H. J. Briegel. "A One-Way Quantum Computer". *Phys. Rev. Lett.* 86, 22 2001, pp. 5188–5191. DOI: 10.1103/PhysRevLett.86.5188. URL: https://link.aps.org/doi/10.1103/PhysRevLett.86.5188.

36. R. Raussendorf, D. Browne, and H. Briegel. "The one-way quantum computer–a non-network model of quantum computation". *Journal of Modern Optics* 49:8, 2002, pp. 1299–1306. DOI: 10.1080/09500340110107487. URL: https://doi.org/10.1080%2F09500340110107487.

37. E. Schrödinger. "An undulatory theory of the mechanics of atoms and molecules". *Physical review* 28:6, 1926, p. 1049.

38. P. W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". *SIAM Journal on Computing* 26:5, 1997, pp. 1484–1509. DOI: 10.1137/s0097539795293172. URL: https://doi.org/10.1137%2Fs0097539795293172.

39. R. D. da Silva and E. F. Galvão. "Compact quantum circuits from one-way quantum computation". *Physical Review A* 88:1, 2013. DOI: 10.1103/physreva.88.012319. URL: https://doi.org/10.1103%2Fphysreva.88.012319.

40. R. D. da Silva, E. Pius, and E. Kashefi. *Global Quantum Circuit Optimization.* 2013. arXiv: 1301.0351 [quant-ph].

41. W. Simmons. "Relating Measurement Patterns to Circuits via Pauli Flow". *Electronic Proceedings in Theoretical Computer Science* 343, 2021, pp. 50–101. DOI: 10.4204/eptcs.343.4. URL: https://doi.org/10.4204%2Feptcs.343.4.

42. P. Walther, K. J. Resch, T. Rudolph, E. Schenck, H. Weinfurter, V. Vedral, M. Aspelmeyer, and A. Zeilinger. "Experimental one-way quantum computing". *Nature* 434:7030, 2005, pp. 169–176. DOI: 10.1038/nature03347. URL: https://doi.org/10.1038%2Fnature03347.

43. G. Wendin. "Quantum information processing with superconducting circuits: a review". *Reports on Progress in Physics* 80:10, 2017, p. 106001. DOI: 10.1088/1361-6633/aa7e1a. URL: https://doi.org/10.1088%2F1361-6633%2Faa7e1a.

44. J. van de Wetering. *ZX-calculus for the working quantum computer scientist.* 2020. arXiv: 2012.13966 [quant-ph].