

Reducing 2-qubit gate count for ZX-calculus based quantum circuit optimization

Korbinian Staudacher, Tobias Guggemos, Sophia Grundner-Culemann

MNM-Team

Ludwig-Maximilians-Universität München
Munich, Germany

{staudacher,guggemos,grundner-culemann}@nm.ifi.lmu.de

Wolfgang Gehrke

Research Institute CODE

Universität der Bundeswehr München Munich, Germany

wolfgang.gehrke@unibw.de

In the near term, programming quantum computers will remain severely limited by low quantum volumes. Therefore, it is desirable to implement quantum circuits with the fewest resources possible. For the common Clifford+T circuits, most research is focused on reducing the number of T gates, since they are an order of magnitude more expensive than Clifford gates in quantum error corrected encoding schemes. However, this optimization sometimes leads to more 2-qubit gates, which, even though they are less expensive in terms of fault-tolerance, contribute significantly to the overall circuit cost. Approaches based on the *ZX-calculus* have recently gained some popularity in the field, but reduction of 2-qubit gates is not their focus. In this work, we present an alternative for improving 2-qubit gate count of a quantum circuit with the *ZX-calculus* by using heuristics in *ZX*-diagram simplification. Our approach maintains the good reduction of the T gate count provided by other strategies based on *ZX-calculus*, thus serving as an extension for other optimization algorithms. Our results show that combining the available *ZX-calculus*-based optimizations with our algorithms can reduce the number of 2-qubit gates by as much as 40% compared to current approaches using *ZX-calculus*. Additionally, we improve the results of the best currently available optimization technique of Nam et. al [22] for some circuits by up to 15%.

1 Introduction

Many famous quantum algorithms, like Shor [26], HHL [13] or Grover [12], base upon techniques like Quantum Fourier Transformation, Quantum Phase Estimation or Amplification, respectively. Although these algorithms provide significant (sometimes even exponential) speed-ups, current quantum chips can only execute toy problems, mostly due to the low gate fidelity. Even for problems that can be easily solved on a state-of-the-art desktop PC, those algorithms require tens of thousands of gates, and are therefore infeasible to run on near-term quantum devices. However, applications in quantum simulation are supposed to achieve significant improvements in quantum chemistry, material sciences, or high-energy physics on near-term devices. With variational algorithms (e.g., QAOA [10] or VQE [25]), real-world applications like optimization problems on real quantum chips may become feasible. While the associated speed-up is unknown for many use cases, they require only few qubits and quantum gates to achieve promising results. Quantum Machine Learning (QML) is such an example: Here, the combination of clever encoding strategies, variational algorithms, and classical pre- and post-processing achieves high accurate classification rates with fewer qubits compared to classical bits.

Still, even algorithms with smaller circuits cannot be executed on current devices and gate optimization is a vibrant research topic. While global optimization of arbitrary quantum circuits is generally QMA-hard [16], different algorithms like quantum optimal control [18] have been proposed to reduce the size of a quantum circuit. In this context the so-called ZX-calculus [6] is considered a promising tool. It provides an abstract graphical language for describing quantum systems and can be seen as an alternative to the predominant description in the Hilbert space. We can transform any quantum circuit into a ZX-diagram equivalent, apply the rules of the ZX-calculus to simplify the diagram, and re-extract a quantum circuit from it.

Scope of this work

Our work is based on optimizing circuits with ZX-calculus, where several optimization strategies have been proposed recently [9, 20, 27]. Currently, these strategies yield very good results for pure *Clifford* circuits, as well as for T gate elimination in *Clifford+T* circuits, which is worthwhile for fault-tolerant quantum computers with error-corrected gates. For such devices, the cost of a T gate is sometimes estimated to be up to a hundred times higher than the cost of a CNOT gate [24] (even though recent studies suggest lower rates [21]).

However, reducing 2-qubit gates is generally of interest for quantum hardware that is not error corrected (e.g., NISQ devices) or in which quantum states do not tend to interact easily, e.g., in Photonic Quantum computing [3]. A major drawback of the current ZX-calculus based strategies is that these gates in particular are not optimized very well; in fact, for many large Clifford+T circuits, the 2-qubit gate count even increases when using algorithms like the one in [9].

We propose new optimization approaches especially for reducing 2-qubit gates. To do so, we use heuristics for estimating the 2-qubit gate count in ZX-diagrams as cost functions for classical search algorithms like **I.**) random selection and **II.**) greedy algorithm. By combining them with existing optimization approaches, we maintain the T gate count reduction rate and improve the total gate count and the 2-qubit gate count for most given circuits. We evaluate the performance on circuits from the *Tpar* benchmark [1]. We find that our optimizations can outperform existing ZX-based approaches and can additionally be used to further improve already optimized circuits.

2 Background

Throughout this paper, we use the notation from [23] for quantum gates; the most essential ones are detailed in Table 1 by name and matrix-, gate- and ZX-calculus-representation. Every unitary operation can be decomposed into a combination of CNOTs and single-qubit gates [23]. A well-studied example for a minimal gate set with which to approximate any unitary operation is the so-called *Clifford+T set*, i.e., the gate set generated by $\{H, T, CNOT\}$. That is why many optimization algorithms target circuits generated with the *Clifford+T set*. The *Clifford* set generated by $\{H, S, CNOT\}$ is also well-known and useful for quantum circuit simulations on classical computers, but not every unitary operation can be represented with it. For convenience, we abbreviate some gates in the *Clifford+T* set, namely X, Y, Z, S , and CZ (instead of writing, for example, $Z = T \cdot T \cdot T \cdot T$).

2.1 ZX-Calculus

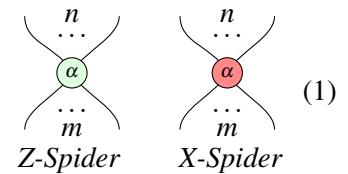
Since ZX-calculus and its optimization strategies rely on graph theory, we provide some background in Appendix B. The ZX-calculus [7, 8] is a graphical language for expressing linear maps on qubits as

Table 1: Overview of important quantum gates and the respective ZX-Spiders.

Name	Identity	Z	Z-Phase	T	X	X-Phase	H	CNOT
Matrix	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\frac{1}{2} \begin{pmatrix} 1+e^{i\alpha} & 1-e^{i\alpha} \\ 1-e^{i\alpha} & 1+e^{i\alpha} \end{pmatrix}$	$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
Gate								
Spider/Wire								

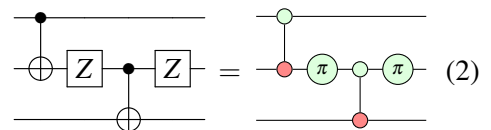
ZX-diagrams. Relations in multi-qubit systems are often difficult to understand in Dirac notation, since the matrix size doubles with every qubit and the complex number space quickly becomes confusing.

ZX-calculus provides a way to represent quantum circuits as 2-dimensional diagrams where nodes (*spiders*) and edges (*wires*) form an undirected graph. In contrast to quantum circuits, the number of input- and output wires does not have to match, hence the resulting transformations are not necessarily unitary. However, many important concepts in quantum mechanics follow very intuitively from this representation and we will briefly introduce the main principles.



2.1.1 Representing Quantum Circuits

Any transformation on a single qubit can be described as a rotation around the X and Z axes. Further, we can represent any quantum gate as a combination of X- (red) and Z-spiders (green) in ZX-diagrams (c.f. Eq. 1), of which the most important are shown in Table 1. We call the wires on the left- and rightmost the *input* and *output* of the graph, respectively.



The three generators of the universal *Clifford+T* set are constructed with the H-wire, the Z-spider with phase $\pi/4$, and a combination of an empty X- and Z-spiders (phase $\alpha = 0$) representing a CNOT. In general, we can read a ZX-diagram in any direction since only the connectivity of the spiders matters, but for comparison with common quantum circuits it is convenient to read ZX-diagrams horizontally as shown in Eq. 2.

2.1.2 Basic Rules

We introduce the most important transformation rules in the ZX language that are useful for optimization [6] in Figure 1. All ZX-rules can be applied in both directions and also apply with inverted colors.

Any two *Clifford* diagrams (i.e., diagrams only containing spiders with a *Clifford* phase $\alpha = k \cdot \frac{\pi}{2}; k \in \mathbb{Z}$) that represent the same linear map can be transformed into each other by some combination of those rules. Recent developments have introduced rule sets where this is also possible for *Clifford+T* diagrams and for all ZX-diagrams [17, 28].

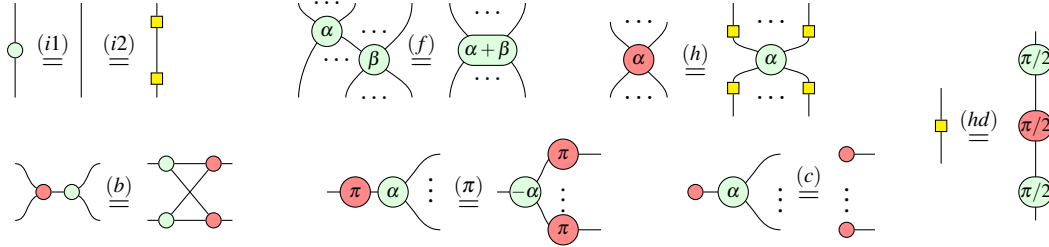


Figure 1: The important rules in ZX-Calculus that can be used for optimization are: Identity- (i1,i2), Fusion- (f), Hadamard- (h), Bialgebra- (b), Pi- (π), Copy- (c) and Hadamard-Decomposition (hd) rule. Each holds for all $\alpha, \beta \in [0, 2\pi]$. Due to (h) and (i2), all rules hold with the colours interchanged.

3 Circuit optimization with ZX-calculus

With the rules of ZX-calculus, the optimization of quantum circuits becomes a *simplification* problem on the ZX-diagram. By simplification we mean reducing the total number of either spiders or wires in a diagram in order to obtain a smaller diagram. The general process is as follows:

- 1.) Transform the circuit to a ZX-diagram
- 2.) (optional:) transform to a *graph-like* diagram, i.e.:
 - All spiders are Z-spiders. wires and are connected to at most one spider.
 - All connections are Hadamard wires. Every spider has at most one input and one output.
 - There are no loops.
 - Inputs and outputs are the only non-Hadamard
- 3.) Simplify the diagram using ZX-rules.
- 4.) Extract a quantum circuit out of the ZX-diagram.

This allows powerful optimization of circuits, which are not obvious at a first glance (we provide an intuitive example in Appendix C).

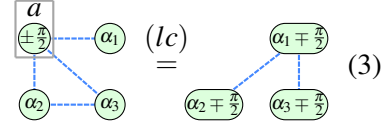
3.1 Diagram simplification

The presented ZX-rules allow many degrees of freedom, hence, simplification is still a difficult problem. The term “simplification of diagrams” has to be taken with a grain of salt since decreasing the number of spiders in a diagram can also lead to more complex extracted circuits. Since rules can be applied in both directions it is important to find *terminating* algorithms for diagram simplification. A common approach has been to only use ZX-rules which *decrease* the total number of spiders in a diagram with every application, thus ensuring termination. We present some common approaches, many of which are implemented in the PyZX-library [19].

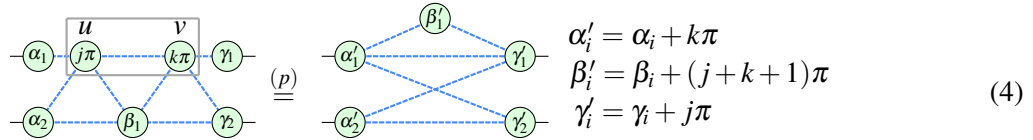
3.1.1 Clifford spider simplification

The core of most strategies are two rules from graph theory – namely *local complementation* and *pivoting* – which work on diagrams that are *graph-like*. Both rules allow the elimination of interior Clifford spiders (phase $0, \pi/2, \pi$, or $-\pi/2$; not connected to an input or output) from ZX-diagrams.

Local complementation (lc) In ZX-calculus, local complementation from Section B.1 is applicable on graph-like diagrams. If the spider a in $G \star a$ has a phase of $\pm\pi/2$, the phase is subtracted from those of the neighboring spiders and the spider is eliminated for simplification of the diagram as shown in Eq. 3.



Pivoting (p) Similarly we can eliminate a pair of spiders uv with phase 0 or π by applying a graph-theoretic pivot $G \wedge uv$ (c.f. Section B.2) on the diagram as in the following example ($j, k \in \mathbb{Z}$):



3.1.2 Clifford simplification algorithm

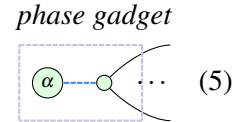
These rules allow constructing an algorithm for graph-like diagrams which removes most interior Clifford spiders [9]. The procedure is as follows:

1. Eliminate empty spiders with two wires using the identity rule and subsequently fuse the adjacent spiders in order to maintain a graph-like diagram.
2. Apply local complementation on every spider of phase $\pm\pi/2$ and pivoting on every pair of connected spiders of phase 0 or π as often as possible.
3. If step 2 modified the diagram, start again with step 1, else stop the iteration.

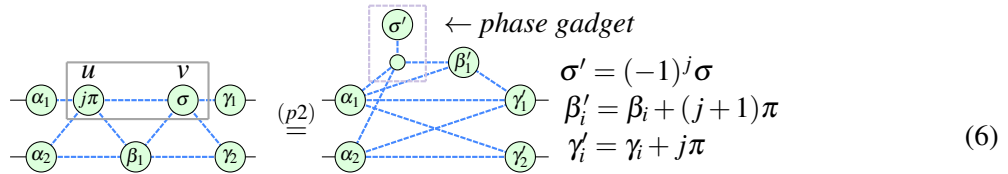
That allows us to remove every interior spider with phase $\pm\pi/2$ and every pair of connected spiders with phase 0 or π . However, after simplification some interior Clifford spiders with phase 0 or π may remain.

3.1.3 Phase gadget simplification (p2)

We can use phase gadgets to apply pivoting on a pair of spiders where one spider has a non-Clifford phase ($\neq 0, \pi/2, \pi, -\pi/2$). A phase gadget as defined in [20] is a parameterized spider with only one wire connected via Hadamard edge to a phaseless spider as in Eq. 5.



We can modify pivoting (Eq. 4) to exchange the spider with a non-Clifford phase to a phase gadget:



With the additional rules in Appendix A, we can eliminate every interior Clifford spider in a diagram [20].

3.2 Circuit extraction

Extracting a quantum circuit from a simplified diagram can be challenging, since spiders with an arbitrary number of wires have no direct gate representation [2]. The most general circuit extraction routine makes use of so called “flow properties” originating in measurement-based quantum computing

(MBQC). Graph-like ZX-diagrams can be seen as an extension of MBQC graph states where the phases of spiders represent measurements in either the XY, XZ or YZ plane of the Bloch sphere:

meas. plane	XY	XZ	YZ
meas. effect		=	=

(7)

Diagrams simplified with the Clifford simplification algorithm from [9] only contain spiders in the XY measurement plane and preserve a flow property called *focused generalized flow (gflow)*. Those diagrams can be extracted by converting every spider with phase α to an $R_z(\alpha)$ gate and a Hadamard wire as either a H or CZ gate or a combination of $CNOT$ gates. As an example consider the diagrams from Eq. 4. Extracting the diagram on the left hand side yields the following circuit:

(8)

whereas extracting the diagram after rule application yields the equivalent smaller circuit:

(9)

However, ZX-diagrams simplified with Eq. 6 may contain spiders in XZ and YZ plane as well. While it has been shown that those diagrams still preserve *generalized flow (gflow)*, the circuit extraction routine has to convert those spiders back into XY spiders using either pivoting (YZ) or a combination of local complementation and pivoting (XZ) before extracting the diagram [2]. An algorithm to efficiently extract diagrams that do not admit the gflow property is yet to be discovered; however, recent findings suggest that such an algorithm may not exist for general ZX-diagrams [4]. Hence, even though the diagram may represent a unitary matrix, we cannot extract a quantum circuit from the diagram efficiently.

4 Enhancing reduction of 2-qubit gates

As seen in Eq. 8 and 9, a Hadamard wire gets extracted to H, CNOT or CZ gates. The number of Hadamard wires in a graph-like ZX-diagram therefore correlates with the number of 2-qubit gates in the extracted circuit. The diagram simplification algorithms shown in Section 3 focus on eliminating spiders while neglecting – or even increasing – the number of Hadamard wires. Hence, this section introduces methods which additionally minimize the amount of Hadamard wires (ref. as `#wires` in the following).

To do so, it is crucial to examine *where* and *when* local complementation and pivoting are applied. Both rules can either increase or decrease `#wires`, depending on the connectivity of the relevant neighbors. Eq. 3 and 4 show examples in which `#wires` decreases. However, as Eq. 6 shows it can also increase `#wires` and we easily construct extreme cases like the one shown in Eq. 10. Applying local complementation to the central spider with phase $\pi/2$ yields a diagram containing one spider less but a significantly higher `#wires`. Extracting the left diagram with the current version of the PyZX-library produces a circuit with six 2-qubit gates, while the diagram on the right gets extracted as a circuit with 21 2-qubit gates. Generally, applying local complementation on a spider with

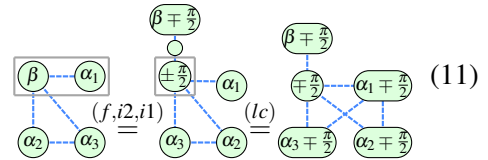
(10)

n unconnected neighbors leads to $\frac{n(n+1)}{2} - n$ new wires. As pivoting involves local complementation on two spiders, the effect usually even worsens for this rule.

To prevent such cases and to guide the simplification process towards a minimal #wires, we introduce cost functions for local complementation and pivoting allowing us to calculate #wires after rule applications. We take those as a heuristic for estimating how rule applications change the number of 2-qubit gates and implement decision strategies for diagram simplification based on the heuristics.

4.1 Pivoting and local complementation on spiders with arbitrary phases

In contrast to the Clifford simplification algorithm from Section 3, we can apply local complementation and pivoting on spiders with *arbitrary* phases. Similar to the *Pivot Phase Gadget* (p2) rule, we can change a spider with non-Clifford phase by a combination of the rules $(f, i2, i1)$ as in Eq. 11.



With that we can apply local complementation on spiders with phase different from $\pm\pi/2$ (this introduces one XZ -spider) and pivoting on pairs of spiders where one/no spider has a phase of 0 or π (this introduces one/two YZ -spiders). Note that this does not change the gflow property (c.f. [2, Lemma 3.1]).

4.2 Local Complementation Heuristic (LCH)

The costs for local complementation are calculated on the following proposition:

Proposition 4.1 *Let $G = (V, E)$ be an open graph; $u \in V$ an arbitrary vertex with neighbors $N(u) \subset V$; $n = |N(u)|$ the number of neighbors; and m the number of edges between the neighbors, i.e.,*

$$m = |\{(a, b) \in E | a, b \in N(u)\}|.$$

For $G \star u$, n remains the same, but m changes to $m' = \Delta_{n-1} - m$, where $\Delta_{n-1} = \frac{n(n-1)}{2}$.

Hence, the difference in the number of wires after application of the local complementation rule is:

$$(n + m) - (n + (\Delta_{n-1} - m)) = 2m - \Delta_{n-1} \quad (12)$$

With respect to the phase $\varphi(u)$ of the spider u , the graph changes as follows:

- If $\varphi(u) = \pm\pi/2$: Remove u from the graph and eliminate all wires between u and $N(u)$.
- If $\varphi(u)$ is non-Clifford: All wires between u and $N(u)$ remain and we get an additional wire for the *phase gadget*.
- If $\varphi(u)$ is 0 or π : No phase gadget is needed and we can use the π -copy rule.

The *LCH* is calculated as follows:

$$LCH(u) = \begin{cases} 2m - \Delta_{n-1} + n & \text{if } \varphi(u) = \pm\frac{\pi}{2} \\ 2m - \Delta_{n-1} & \text{if } \varphi(u) = k \cdot \pi, k \in \mathbb{Z} \\ 2m - \Delta_{n-1} - 1 & \text{otherwise} \end{cases} \quad (13)$$

4.3 Pivoting Heuristic (PH)

We calculate the upper bound of new connections with the sets A, B, C (see Appendix B):

$$C_{max} = |A| \cdot |B| + |A| \cdot |C| + |B| \cdot |C| \quad (14)$$

We denote the number of neighbors of u and v by $n_u = |N(u)|$ and $n_v = |N(v)|$, respectively, and the number of edges between neighbors of different sets as m .

The changes of the graph G to $G \wedge uv$ have the following cases ($j, k \in \mathbb{Z}$):

- (C1) $\varphi(u) = j \cdot \pi, \varphi(v) = k \cdot \pi$: If both spiders have a phase of 0 or π , all connections between $\{u, v\}$ and $N(u) \cup N(v)$ are eliminated.
- (C2) $\varphi(u) = j \cdot \pi, \varphi(v) \neq k \cdot \pi$: If v becomes a phase gadget and u gets eliminated, all neighbors of u get connected to v and we have an additional wire for the phase gadget.
- (C3) $\varphi(u) \neq j \cdot \pi, \varphi(v) = k \cdot \pi$: If u becomes a phase gadget and v gets eliminated, all neighbors of v get connected to u and we have an additional wire for the phase gadget.
- (C4) $\varphi(u) \neq j \cdot \pi, \varphi(v) \neq k \cdot \pi$: If both spiders become phase gadgets, all neighbors of u get connected to v and all neighbors of v get connected to u . Furthermore, u gets connected to v again and we have two more wires for the phase gadgets.

With these conditions, the PH is calculated as follows:

$$PH(u, v) = \begin{cases} 2m - C_{max} + n_u + n_v - 1 & \text{for (C1)} \\ 2m - C_{max} + n_v - 1 & \text{for (C2)} \\ 2m - C_{max} + n_u - 1 & \text{for (C3)} \\ 2m - C_{max} - 2 & \text{for (C4)} \end{cases} \quad (15)$$

4.4 Decision strategies

With the two heuristics (LCH, PH) at hand we can now implement different strategies to decide where and when local complementation or pivoting are applied during the simplification. A single simplification step in our procedure consists of the following actions:

1. Filter all possible rule applications of the current ZX-diagram.
2. Select rule according to selection strategy (see below).
3. Apply rule on the ZX-diagram.

For filtering rule applications we can specify a lower bound for the heuristic, e.g., LCH or $PH = -5$ says that we do not consider rule applications which increase $\#wires$ by more than five. We can also specify whether rule applications are allowed on boundary spiders (c.f. Appendix A) and whether rules are allowed on arbitrary phased spiders. For selecting a rule we implemented two different strategies:

- **Random selection:** Rules are chosen by a random coin flip.
- **Greedy selection:** Chooses the rule application which maximally decreases $\#wires$.

Each algorithm terminates if we allow only rule applications with a $LCH/PH > 0$ and when there is no rule left that decreases $\#wires$. They also terminate if we allow negative gains ($LCH/PH \leq 0$) and restrict the matches to interior spiders that do not generate new spiders. This is the case in standard local complementation on a spider with phase $\pm\pi/2$ and pivoting on a pair of spiders with phase 0 or π . The

algorithm eliminates at least one spider in every step and terminates when neither interior spiders with phase $\pm\pi/2$ nor pairs with phase 0 or π are left.

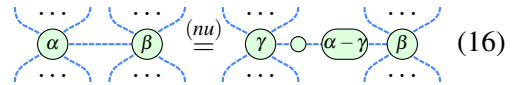
On the other hand, allowing rule applications on spiders of arbitrary phases which increase `#wires` may result in loops and therefore no termination. For such cases we only allow rule applications which increase `#wires` on spiders present since the very beginning of our simplification procedure. On newly generated spiders only rules which decrease `#wires` are allowed.

5 New optimization rule: Neighbor Unfusion

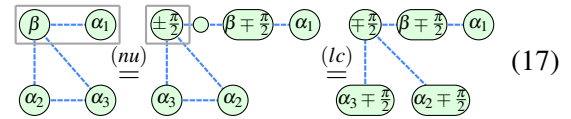
As shown in Eq. 6, the application of local complementation (lc) and pivoting (p) on spiders with many neighbors can not only decrease but also increase `#wires`. The heuristics shown in Section 4 may help to identify and prevent extreme cases as in Eq. 10. However, spiders that are measured in YZ - or XZ -plane (c.f. Section 3.2) require special attention: When extracting a spider in YZ -plane (e.g., the empty spider of the phase gadget in Eq. 5 and 6), pivoting has to be applied to maintain the focused gflow property. The same happens for spiders in XZ -plane, but they are resolved by local complementation [2]. This affects `#wires` after the simplification and a simplified diagram containing some spiders in YZ - and XZ -plane may result in an expensive circuit.

However, when applying either rule on diagrams with arbitrary spiders as discussed in Section 4.1), spiders in YZ - or XZ -plane are generated. We introduce the *neighbor unfusion* (nu) rule, which allows lc and p on such arbitrary-phase spiders without introducing spiders in YZ - or XZ -plane.

Neighbor unfusion combines the fusion (f) and identity rules ($i1, i2$) as shown in Eq. 16. If a spider with phase α is connected to a neighbor, we change its phase to an arbitrary phase γ by inserting an empty spider and a spider with phase $\alpha - \gamma$ between the spider and its neighbor. It allows changing the phase of an arbitrary spider to $\gamma = \pm\frac{\pi}{2}$ and thus local complementation and removal of spiders with arbitrary phases.



For illustration, we apply neighbor unfusion (nu) to the example of Eq. 11. We can move the β spider to any direction (in Eq. 17 towards α_1), so it is then not



affected by the application of local complementation (lc). Comparing Eq. 11 and Eq. 17 we see that we not only reduce `#wires`, but also prevent the generation of a spider in XZ -plane. However, the neighbor unfusion rule sometimes leads to diagrams which do not have focused gflow property. This is due to the insertion of the empty spider and the spider with phase $\alpha - \gamma$ in 16. We observed that this problem does not occur if the spiders with phase α and β get extracted to the same qubit. Currently, we find such pairs of spiders by using the flow hierarchy of the *maximally delayed gflow* of the diagram (c.f. [2]), which is quite costly, because we need to recalculate the gflow at each simplification step. Therefore, diagram simplification with neighbor unfusion has a much higher runtime than the other simplification procedures. It is an open question whether neighbour unfusion can only destroy the focused gflow property, or also more general flow properties like gflow or Pauli flow.

6 Evaluation

We evaluate our heuristic-based simplification algorithms on a set of circuits first used in [1]. They implement various arithmetic problems as quantum circuits and were used as a benchmark set for comparing different optimization strategies [20, 22]. We use it to compare our heuristic-based approaches

Table 2: Circuit metrics for original benchmark circuits, Post-optimization metrics of the standard Clifford simplification [9], PyZX [20], Nam et al. [22], our heuristic-based simplification method, and the combined approach of Nam et al. and our heuristic-based methods. Only our best optimization is presented: 1) Greedy, 2) Random, 3) Greedy with neighbor unfusion, 4) Random with neighbor unfusion, the lower bound for the heuristics is denoted in brackets. If the best PyZX result is achieved by the $t|ket\rangle$ -library, the respective cell is marked with \star . The best results of each metric in each row are marked.

Circuit	Original		Clifford algorithm		PyZX/ $t ket\rangle^*$		Nam et al.		Heuristic algorithm			Nam+Heuristic		
	Σ	$2Q$	Σ	$2Q$	Σ	$2Q$	Σ	$2Q$	Σ	$2Q$	Alg.	Σ	$2Q$	Alg.
Mod 5 ₄	63	28	36	21	24*	12*	51	28	41	23	2(-20)	38	23	3(1)
VBE-Adder ₃	150	70	116	59	101	54	89	50	87	42	3(1)	87	42	4(1)
CSLA-MUX ₃	170	80	177	97	156	75	155	70	155	74	3(-5)	156	67	3(1)
CSUM-MUX ₃	420	168	455	271	327*	158*	266	140	303	150	3(1)	266	140	1(1)
QCLA-Com ₇	443	186	397	223	316	148	284	132	295	138	4(-5)	275	132	1(1)
QCLA-Mod ₇	884	382	903	475	717	324	-	-	705	311	4(-20)	-	-	-
QCLA-Adder ₁₀	521	233	562	305	435	199	399	183	417	193	4(-20)	398	182	4(1)
Adder ₈	900	409	779	429	675	339	606	291	597	295	4(1)	514	256	4(1)
RC-Adder ₆	200	93	206	113	393*	164*	140	71	159	71	1(1)	152	71	1(1)
Mod-Red ₂₁	278	105	260	130	217	93	180	77	196	85	3(1)	179	76	1(1)
Mod-Mult ₅₅	119	48	124	74	91	42	91	40	90	40	1(1)	90	41	1(1)
Toff-Barenc ₃	58	24	50	26	59*	18*	40	18	46	21	1(1)	40	18	3(-5)
Toff-NC ₃	45	18	41	20	40	16	35	14	36	15	3(1)	35	14	1(1)
Toff-Barenc ₄	114	48	117	60	95	44	72	34	88	40	4(1)	72	34	3(1)
Toff-NC ₄	75	30	86	43	65	26	55	22	57	24	3(1)	55	22	1(1)
Toff-Barenc ₅	170	72	149	86	140	66	104	50	122	57	4(1)	102	48	3(1)
Toff-NC ₅	105	42	92	42	90	36	75	30	78	33	3(1)	75	30	1(1)
Toff-Barenc ₁₀	450	192	392	196	365	176	264	130	325	151	4(1)	252	118	3(1)
Toff-NC ₁₀	255	102	237	100	215	86	175	70	183	78	3(1)	175	70	1(1)
GF(2 ⁴)-Mult	225	99	245	140	193	99	187	99	195	101	2(1)	180	98	3(-5)
GF(2 ⁵)-Mult	347	154	351	197	304	154	296	154	306	156	1(1)	289	155	4(-20)
GF(2 ⁶)-Mult	495	221	545	308	422	221	403	221	418	217	4(-5)	390	218	3(-5)
GF(2 ⁷)-Mult	669	300	736	417	573	300	555	300	572	299	4(-5)	535	292	4(-20)
GF(2 ⁸)-Mult	883	405	1015	606	745	405	712	405	745	405	1(1)	691	399	1(1)
Avg. reduction			~ 3%	~ -22%	~ 14%	~ 9%	~ 27%	~ 19%	~ 23%	~ 16%		~ 29%	~ 21%	

with the Clifford simplification algorithm described in [9], and some of the best results reported for circuit optimizations with [20] and without using ZX-calculus [22]. We also investigate how ZX-calculus based approaches perform when using the TODD-algorithm [14] for additional T gate reduction.

6.1 Implementation

With the exception of [20] – which omits simplifying the diagram (step 3) and extraction (step 4) – we use the following pipeline for ZX-calculus based optimization algorithms:

1. Optimize circuit using gate cancellation and commutation.
2. Transform circuit to ZX-diagram and apply phase teleportation to reduce T-count (as in [20]).
3. Simplify ZX-diagram (standard Clifford or heuristic-based simplification).
4. Extract circuit from ZX-diagram.
5. Optimize circuit as in step 1.

Since our heuristic-based algorithms do not reduce T gates, we always apply the phase teleportation in step 2 since this reduces T gates as far as currently possible with ZX-calculus. This ensures comparable results regarding the 2-qubit gate count against non-ZX-calculus based approaches that optimize all types of gates. We implemented our algorithms in a clone of the PyZX library which also contains our optimized circuits in the OpenQASM format¹. All results were proven to be correct by checking whether the optimized circuit together with the adjoint of the original circuit can be reduced to the identity.

6.2 Results

For each circuit we compare the total gate count Σ and the 2-qubit gate count $2Q$. The results are summarized in Table 2 and 3: Their columns show circuit name, metrics of the original circuit of the benchmark, metrics of one (or more) existing optimization algorithms and the metrics of the best performing heuristic-based algorithm. For the latter, we denote the simplification strategy achieving the best result in the last column: 1. Greedy, 2. Random, 3. Greedy with neighbor unfusion, 4. Random with neighbor unfusion.

As a very first result, the last column in the “Heuristic Algorithm” section of Table 2 prominently indicates the great value of neighbor unfusion (Alg. 3 and 4), as it achieves the best performance of our heuristics in $> 70\%$ of the cases.

We now compare our heuristic-based simplifications following against other ZX-calculus based optimizations in Table 2. For most circuits our heuristic-based simplification clearly outperforms the standard Clifford simplification [9], both in total and 2-qubit gate reduction. Moreover, while our approaches almost always decrease circuit metrics, the standard approach often yields circuits with higher metrics than the original circuit (e.g., “CSLA-MUX₃”, “GF(2⁶)-Mult”). Especially for 2-qubit gates our approaches *decrease* 2-qubit gate count by 16%, while the standard approach even *increases* the count by 22%. In a direct comparison our approaches have up to 33% (“Toff-NC₄”) fewer total and 47% (“Mod-Mult₅₅”) fewer 2-qubit gates than the standard Clifford approach.

Second, we compare against the best available PyZX implementation [20] and the recommended optimization pipeline of the t|ket)-library [27] with the routines PauliSimp and FullPeepholeOptimize, which use similar strategies. The column “PyZX/t|ket)” in Table 2 shows the best optimization results for both implementations and \star indicates results from t|ket). Except for two circuits (“Mod 5₄” and “Toff-Barenco₃”), our algorithms outperform all ZX-calculus based algorithms in terms of total gate count and 2-qubit gate count.

Third, Table 2 also shows our result in comparison to the cutting-edge non-ZX-calculus based algorithm from Nam et al. [22]. It can be seen that the algorithm from [22] outperforms any ZX-calculus based algorithm for most circuits. Still, we were able to achieve better results for the circuits “VBE-Adder₃” and “Mod-Mult₅₅”. Note that we did not compare for the “QCLA-Mod₇” circuit, because [15] reports that the optimized circuit from [22] does not correspond to the original.

Last, the rightmost columns of Table 2 show a combination of Nam et al’s approach with ours. We

Table 3: Circuit metrics for the original benchmark circuits, Post-optimization metrics for PyZX+TODD and of our heuristic-based algorithms+TODD.

Circuit	Original			PyZX+TODD			Heuristic+TODD			Alg.
	Σ	$2Q$	\mathcal{T}	Σ	$2Q$	\mathcal{T}	Σ	$2Q$	\mathcal{T}	
CSLA-MUX ₃	170	80	70	262	175	43	257	169	43	2(1)
CSUM-MUX ₃	420	168	196	575	428	74	411	261	74	4(-5)
QCLA-Com ₇	443	186	203	454	274	93	389	211	93	4(1)
QCLA-Adder ₁₀	521	233	238	800	517	143	677	391	143	4(-20)
Mod-Mult ₅₅	119	48	49	107	56	27	104	55	27	4(-5)
GF(2 ⁴)-Mult	225	99	112	298	221	52	295	220	52	1(-5)
GF(2 ⁵)-Mult	347	154	175	538	420	88	524	403	88	3(1)
GF(2 ⁶)-Mult	495	221	252	943	764	134	933	750	134	3(1)
GF(2 ⁷)-Mult	669	300	343	1253	1036	180	1223	993	180	4(1)
GF(2 ⁸)-Mult	883	405	448	1791	1521	224	1780	1507	224	3(1)

¹<https://github.com/mnm-team/pyzx-heuristics>

use the output circuits from the Nam et al. optimization as input for our algorithms and observe that we achieve equally good or better results for almost all circuits. The larger the circuit, the more significantly this combination improves the previous best known results. Most notably, we improve the total count of the “Adders” circuit by more than 15% and the 2-qubit gate count by more than 12%.

Apart from the 2-qubit gate count, the T gate count of a quantum circuit is an important metric, since T-gates are more complex to implement for an error-corrected quantum computer. Therefore, we compare our algorithms to the other ZX-calculus based approaches using the TODD algorithm as optimization step 1). It is designed to reduce T gate count by introducing ancilla qubits, but sometimes also reduces T-gates in the ancilla-free case. Table 3 shows those benchmarks circuits where the combination of TODD and a ZX-calculus based algorithm reduces T gate count even more compared to the best result in Table 2. We compare the best combination of our heuristic-based algorithm and TODD against the best combination of an existing ZX-calculus based algorithm and TODD.

While we observe a general increase in 2-qubit and total gates using TODD algorithm, our best algorithm yields better results than the existing ZX-calculus based algorithms in every case.

7 Conclusions and Future Work

In this work we introduce two functions, namely the Local Complementation Heuristic LCH (for the local complementation rule) and the Pivot Heuristic PH (for the pivot rule). The functions calculate the number of Hadamard wires that would be added or removed by applying the respective rule, thus serving as a heuristic for estimating the 2-qubit gate count of the underlying circuit. This allows us to develop a more sophisticated strategy for ZX-diagram simplification: First, dismiss the applicable rules that cost too much and then either select a rule randomly or select the rule with the best wire count decrease.

Notably, the T gate count remains unchanged throughout this process, which is why our approach and others that mainly decrease the T gate count complement each other well. Further, we introduce the new *Neighbor Unfusion* rule which combines the established *fusion* and *identity* rules. This rule allows introducing spiders with arbitrary phases into the circuit if needed, for example when the local complementation or pivot rule would be useful to reduce Hadamard wires. As a side note, we also formally describe how to use the local complementation and the pivoting rule on spiders with non-Clifford phases, which is a common implementation practice but has never been mentioned in theory.

We measure the impact of aligning the optimization strategy with the heuristics and adding the neighbor unfusion rule by comparing our algorithm to four other approaches, some based on ZX-calculus and some not, on a set of 24 well-established benchmark circuits. Our approaches show significant improvements compared to all other ZX-based approaches, especially in 2-qubit gate reduction. On their own, non-ZX-based approaches still yield slightly better results than our ZX-based approaches. However, when combining both we are able to optimize circuits better than the previously best known result, which seems to be a promising field for further research.

Using heuristics for ZX-diagram simplification also provides many possibilities for future improvement. Regarding the selection of rules, both random and greedy strategy are non-optimal for finding a ZX-diagram with minimal number of wires. Instead, we propose using a metaheuristic selection strategy like simulated annealing for escaping local minima during simplification. Furthermore, since simplification with neighbor unfusion tends to yield the best results, we think it is important to further investigate in which cases neighbor unfusion generates XY spiders and if we can preserve valid ZX-diagrams when allowing unfusion on spiders which get extracted on different qubits.

Acknowledgment

This work is partially supported by the German Federal Ministry of Education and Research (BMBF) under the funding programme Quantum Technologies - From Basic Research to Market under contract number 13N16077.

References

- [1] Matthew Amy, Dmitri Maslov & Michele Mosca (2014): *Polynomial-time T -depth optimization of Clifford+ T circuits via matroid partitioning*. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33(10), pp. 1476–1489.
- [2] Miriam Backens, Hector Miller-Bakewell, Giovanni de Felice, Leo Lobski & John van de Wetering (2021): *There and back again: A circuit extraction tale*. *Quantum* 5, p. 421.
- [3] Stefanie Barz (2015): *Quantum computing with photons: introduction to the circuit model, the one-way quantum computer, and the fundamental principles of photonic experiments*. *Journal of Physics B: Atomic, Molecular and Optical Physics* 48(8), p. 083001, doi:10.1088/0953-4075/48/8/083001. Available at <https://doi.org/10.1088/0953-4075/48/8/083001>.
- [4] Niel de Beaudrap, Aleks Kissinger & John van de Wetering (2022): *Circuit Extraction for ZX-diagrams can be # P-hard*. *arXiv preprint arXiv:2202.09194*.
- [5] André Bouchet (1988): *Transforming trees by successive local complementations*. *Journal of Graph Theory* 12(2), pp. 195–207, doi:10.1002/jgt.3190120210.
- [6] Bob Coecke & Ross Duncan (2011): *Interacting quantum observables: categorical algebra and diagrammatics*. *New Journal of Physics* 13(4), p. 043016, doi:10.1088/1367-2630/13/4/043016.
- [7] Bob Coecke, Dominic Horsman, Aleks Kissinger & Quanlong Wang (2021): *Kindergarten quantum mechanics graduates (... or how I learned to stop gluing LEGO together and love the ZX-calculus)*. *arXiv preprint arXiv:2102.10984*.
- [8] Bob Coecke & Aleks Kissinger (2018): *Picturing quantum processes*. In: *International Conference on Theory and Application of Diagrams*, Springer, pp. 28–31.
- [9] Ross Duncan, Aleks Kissinger, Simon Perdrix & John Van De Wetering (2020): *Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus*. *Quantum* 4, p. 279.
- [10] Edward Farhi, Jeffrey Goldstone & Sam Gutmann: *A Quantum Approximate Optimization Algorithm*. Available at <http://arxiv.org/pdf/1411.4028v1>.
- [11] Jim Geelen & Sang-il Oum (2009): *Circle graph obstructions under pivoting*. *Journal of Graph Theory* 61(1), pp. 1–11, doi:10.1002/jgt.20363.
- [12] Lov K. Grover (1996): *A fast quantum mechanical algorithm for database search*. In Gary L. Miller, editor: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, ACM, New York, NY, pp. 212–219, doi:10.1145/237814.237866.
- [13] Aram W. Harrow, Avinandan Hassidim & Seth Lloyd (2009): *Quantum algorithm for solving linear systems of equations*. *Physical Review Letters* 103(15), p. 1474, doi:10.1103/PhysRevLett.103.150502. Available at <https://arxiv.org/pdf/0811.3171>.
- [14] Luke E Heyfron & Earl T Campbell (2018): *An efficient quantum compiler that reduces T count*. *Quantum Science and Technology* 4(1), p. 015004.
- [15] Kesha Hietala, Robert Rand, Shih-Han Hung, Xiaodi Wu & Michael Hicks (2021): *A verified optimizer for Quantum circuits*. *Proceedings of the ACM on Programming Languages* 5(POPL), pp. 1–29.
- [16] Dominik Janzing, Pawel Wocjan & Thomas Beth: *Identity check is QMA-complete*. Available at <https://arxiv.org/pdf/quant-ph/0305050>.

- [17] Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2018): *A complete axiomatisation of the ZX-calculus for Clifford+ T quantum mechanics*. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pp. 559–568.
- [18] Navin Khaneja & Steffen J. Glaser (2001): *Cartan decomposition of $SU(2n)$ and control of spin systems*. *Chemical Physics* 267(1-3), pp. 11–23, doi:10.1016/S0301-0104(01)00318-4. Available at <https://www.sciencedirect.com/science/article/pii/S0301010401003184>.
- [19] Aleks Kissinger & John van de Wetering (2020): *PyZX: Large Scale Automated Diagrammatic Reasoning*. In Bob Coecke & Matthew Leifer, editors: *Proceedings 16th International Conference on Quantum Physics and Logic*, Chapman University, Orange, CA, USA., 10-14 June 2019, *Electronic Proceedings in Theoretical Computer Science* 318, Open Publishing Association, pp. 229–241, doi:10.4204/EPTCS.318.14.
- [20] Aleks Kissinger & John van de Wetering (2020): *Reducing the number of non-Clifford gates in quantum circuits*. *Physical Review A* 102(2), p. 022406.
- [21] Daniel Litinski (2019): *Magic State Distillation: Not as Costly as You Think*. *Quantum* 3, p. 205, doi:10.22331/q-2019-12-02-205.
- [22] Yunseong Nam, Neil J Ross, Yuan Su, Andrew M Childs & Dmitri Maslov (2018): *Automated optimization of large quantum circuits with continuous parameters*. *npj Quantum Information* 4(1), pp. 1–12.
- [23] Michael A. Nielsen & Isaac L. Chuang (2013): *Quantum computation and quantum information*, first south asia edition edition. Cambridge University Press, Cambridge.
- [24] Joe O’Gorman & Earl T. Campbell (2017): *Quantum computation with realistic magic-state factories*. *Physical Review A* 95(3), doi:10.1103/PhysRevA.95.032338.
- [25] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik & Jeremy L. O’Brien (2014): *A variational eigenvalue solver on a photonic quantum processor*. *Nature Communications* 5(1), p. 4213, doi:10.1038/ncomms5213. Available at <https://www.nature.com/articles/ncomms5213>.
- [26] P. W. Shor (1994): *Algorithms for quantum computation: Discrete logarithms and factoring*. In Shafi Goldwasser, editor: *Foundations of Computer Science, 35th Symposium on (FOCS '94)*, IEEE Computer Society Press, pp. 124–134, doi:10.1109/SFCS.1994.365700.
- [27] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington & Ross Duncan (2020): *t|ket>: a retargetable compiler for NISQ devices*. *Quantum Science and Technology* 6(1), p. 014003.
- [28] Renaud Vilmart (2019): *A near-minimal axiomatisation of zx-calculus for pure qubit quantum mechanics*. In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, IEEE, pp. 1–10.

A Further rules

In addition to the rules in Section 3, additional rules have been developed to eliminate *every* interior Clifford spider.

A.1 Pivoting Boundary Spiders (p1)

The pivoting rule can also be applied if one of the spiders is a boundary spider, i.e., connected to an input or output, using the following transformation:

(18)

Here v gets transformed to an interior spider and both u and v can be removed using the pivoting rule.

A.2 Gadget Fusion (gf):

An important feature of phase gadgets is that we can fuse two phase gadgets connected to the same neighbors by summing up their phases.

(19)

This rule is used for eliminating non-Clifford spiders in a diagram, for instance, two phase gadgets with phase $\pi/4$ connected to the same set of neighbours can be fused into a single phase gadget with phase $\pi/2$. Combining the Clifford simplification algorithm with those extended rules we can eliminate *all* interior Clifford spiders (in exchange for phase gadgets) and *some* interior non-Clifford spiders.

B Graph Theory

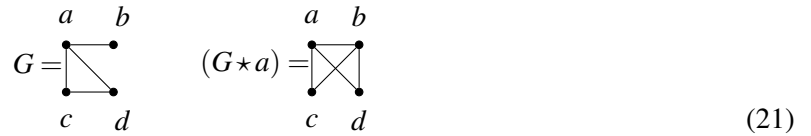
Since ZX-calculus and its optimization strategies rely on graph operations on undirected graphs, we provide some background on it: An *undirected graph* is a tuple $G = (V, E)$ with *vertices* (or “nodes”) V and *edges* $E \subseteq V \times V$.

B.1 Local Complementation

The local complementation \star [5] of an undirected graph $G = (V, E)$ about a vertex u is defined as follows (Δ is the symmetric set difference: $A \Delta B := (A \cup B) \setminus (A \cap B)$):

$$G \star u := (V, E \Delta \{(a, b) \mid (a, u), (b, u) \in E, a \neq b\}) \quad (20)$$

The following example shows a graph G and its local complementation about a . Intuitively, local complementation connects two neighbours of a if they are *not* connected (e.g., b, c) and disconnects them otherwise (e.g., c, d).



B.2 Pivoting [11]

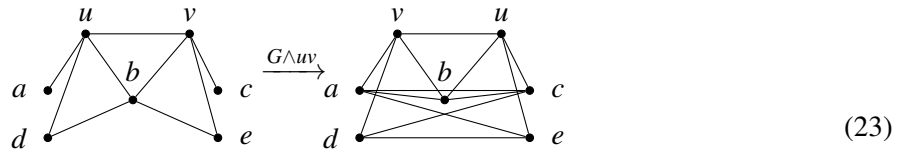
Pivoting \wedge rewrites an edge $(u, v) \in E$ by triple local complementation:

$$G \wedge uv := ((G \star u) \star v) \star u \quad (22)$$

To derive the new graph, we consider three disjoint sets (where the neighborhood of vertex x is defined as $N(x) = \{y \in V \mid (x, y) \in E\}$):

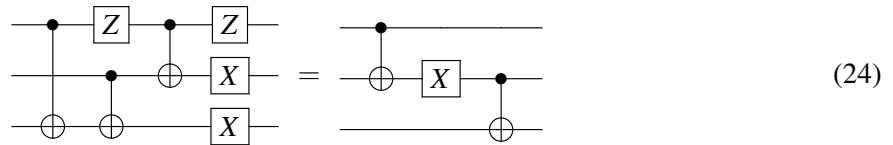
- $A := N(u) \cap N(v)$: Vertices connected to u and v .
- $B := N(u) \setminus N(v)$: Vertices connected to u and not to v .
- $C := N(v) \setminus N(u)$: Vertices connected to v and not to u .

In a pivoted graph $G \wedge uv$, two vertices from different sets A, B or C are connected if, and only if, the two are not connected in G . Connections between vertices of the same set are not modified. As an example, consider the following graphs G (left) and $G \wedge uv$ (right), where $A = \{b\}$, $B = \{a, d\}$, $C = \{c, e\}$. Intuitively, pivoting connects all vertices between A, B, C that are not connected in G (e.g., a, b) and disconnects them otherwise (e.g., b, d):



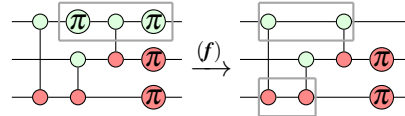
C Example for ZX optimization

The following circuit can be optimized as follows:

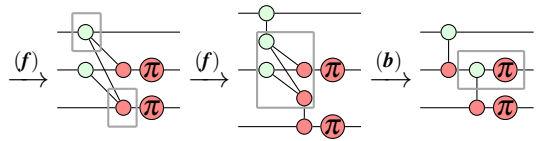


We use the following rules² (the affected spiders/wires to which a rule is applied are framed):

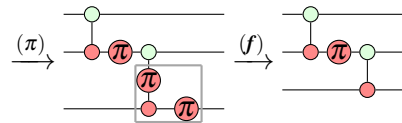
1) Eliminate the two Z-gates using spider fusion (f):



2) Reduce from 3 to 2 CNOTs with fusion (f) and bialgebra rule (b):



3) Eliminate one X by the π copy rule:



²The example is inspired by a talk of Russ Duncan “Quantum Formal Methods” from 2021 (1h 35min) which is publicly available.