Exhaustive Search for Quantum Circuit Optimization using ZX Calculus

Tobias Fischbach[1][0009−001−2535−2577], Pierre Talbot[1][0000−0001−9202−4541], and
Pascal Bouvry[1][0000−0001−9338−2834]

Department of Computer Science, University of Luxembourg, Esch-sur-Alzette, Luxembourg
`firstname.lastname@uni.lu`

Abstract. Quantum computers allow a near-exponential speed-up for specific applications when compared to classical computers. Despite recent advances in the hardware of quantum computers, their practical usage is still severely limited due to a restricted number of available physical qubits and quantum gates, short coherence time, and high error rates. This paper lays the foundation towards a metric independent approach to quantum circuit optimization based on exhaustive search algorithms. This work uses depth-first search and iterative deepening depth-first search. We rely on ZX calculus to represent and optimize quantum circuits through the minimization of a given metric (e.g. the T-gate and edge count). ZX calculus formally guarantees that the semantics of the original circuit is preserved. As ZX calculus is a non-terminating rewriting system, we utilise a novel set of pruning rules to ensure termination while still obtaining high-quality solutions. We provide the first formalization of quantum circuit optimization using ZX calculus and exhaustive search. We extensively benchmark our approach on 100 standard quantum circuits. Finally, our implementation is integrated in the well-known libraries PyZX and Qiskit as a compiler pass to ensure applicability of our results.

Keywords: Quantum Circuit Optimization · ZX Calculus · Exhaustive Search.

1 Introduction

Quantum computers allow a near-exponential speed-up for specific applications when compared to classical computers. Typical examples that are benefitted by quantum computing include the simulation of quantum systems, solving combinatorial problems, performing machine learning and breaking cryptography [23]. However, current quantum computers lack the resources to address complex real-world problems. These restrictions concern the number of available physical qubits and quantum gates, high error rates, and a short coherence time. Quantum error correction aims to mitigate these challenges at the cost of a higher resource demand [9].

Similarly to classical computing, quantum circuits describe quantum programs within the quantum gate model. These circuits are independent of an underlying architecture and allow universal computation [10]. Users typically use Clifford gates with the T-gate as the chosen universal gate set because it can be efficiently simulated on classical computers [1]. However, different types of gates require varying amounts of resources, with the T-gate requiring more physical qubits and error correction code than Clifford gates to be implemented in a quantum device [19].

Inherent limitations in quantum architecture are the number of available physical qubits and quantum gates, as well as a short coherence time. These limitations can be addressed by architecture-dependent optimization that improves the mapping of a quantum circuit onto a specific quantum

hardware. Architecture-dependent quantum circuit optimization can be treated as a classical optimization problem that can be solved exactly [41]. Other methods include heuristics and deep learning [30,20].

In this paper, we target *architecture-independent optimization* which aims to simplify a quantum circuit by reducing general and common limitation factors across architectures such as the number of quantum gates, logical qubits and the circuit depth. Despite the existence of infinite universal gate sets, common methods employ gate commutation rules [26] and circuit simplification [35] for frequently used universal gate sets. These heuristic approaches suffer from several drawbacks. First, it is necessary to prove that each new simplification rule is correct, to make sure that the semantics of the original circuit is preserved. Second, there is no guarantee of optimality, even for small circuits. And finally, the heuristics are tailored to optimize one particular objective and must be redesigned when the requirements change. To tackle these challenges, several optimization methods based on ZX calculus (Section 2) recently emerged [15,38].

ZX calculus is a universal, compact and complete rewriting system [11,13]. An object in ZX calculus is depicted graphically and called a *ZX diagram*. Quantum circuits and ZX diagrams both represent a linear map between qubits. ZX calculus is *universal* because every quantum circuit can be converted to a ZX diagram. It is *complete* because applying any rule preserves semantics, which means that the linear map of qubits remains unchanged [3,4,27]. Finally, ZX calculus is *compact* because it consists only of two generators and eight rules. A ZX diagram can be converted back to a quantum circuit, which is a non-trivial process known as the *circuit extraction problem* [6].

However, some characteristics of ZX calculus make the design of an optimization algorithm challenging:

- **High memory requirements**: Real-world quantum circuits result in large ZX diagrams with high memory requirements for every state.
- **Non-terminating**: Infinite rewriting sequences exist.
- **Failed states**: The extraction of ZX diagrams to quantum circuit might fail (or be prohibitively long) with current algorithms. Checking if a diagram is extractable is time consuming and prevents us from exploring a large number of nodes.

In light of these challenges, we contribute a proof of concept to quantum circuit optimization by employing *iterative deepening depth-first search* (IDDFS) to systematically explore the rewritten ZX diagrams. IDDFS is a well-known state-space search strategy, which is simple, memory efficient and provides a good trade-off between exploration and exploitation [33]. Our approach is general in the sense that the same search strategy can be employed to optimize different metrics. A metric can be defined on the basis of characteristics of the ZX diagram or its corresponding quantum circuit. In particular, we aim to find a ZX diagram that minimizes the T-gate count due to its high impact on the practicability of current quantum architectures. Furthermore, we demonstrate the metric independence of our approach by optimizing the edge count of the ZX diagram. In sum, the contributions of this paper are as follows:

1. A formal description of ZX diagram optimization and the first state-space search algorithm applied to ZX diagram optimization (Section 3).
2. Proof of concept implementation that is extensively benchmarked on 100 standard quantum circuits that is equating the state-of-the-art full reduce algorithm on 89% of the circuits within 1.5 hours. (Section 4.1).
3. A PyZX and Qiskit based transpiler pass to support the practical adoption of our results.

Fig. 1: The basic rewriting rules of ZX calculus.

2 Preliminaries

2.1 ZX Calculus

ZX calculus, introduced by Coecke and Duncan in 2008, is a universal framework for diagrammatic reasoning between linear maps of qubits [12,13]. It provides a complete set of sound semantic-preserving rewriting rules, even for arbitrary real phases [3,4,27,28]. Completeness signifies that the linear map of a ZX diagram remains unchanged after modifications through a rewriting rule. ZX calculus is universal and allows users to represent any quantum circuit as a ZX diagram. The elementary building blocks of quantum circuits are quantum gates and wires. Analogously, the elementary building blocks in ZX calculus—called *generators*—are spiders, wires, swap and Bell states.

A spider is a tensor which operates on qubits in either the Z-basis $\{|0\rangle, |1\rangle\}$ (green) or X-basis $\{|-\rangle, |+\rangle\}$ (red). Spiders possess n inputs, m outputs and carry a phase α. The linear map of a spider for phase $\alpha = 0$ results in the identity matrix $\mathbb{1}_{2^m \times 2^n}$ and therefore acts as a wire. Spiders with phases that are multiples of $\frac{\pi}{2}$ can implement all Clifford gates. The T-gate corresponds to a Z-spider with a phase of $\frac{\pi}{4}$. Clifford gates and the T-gate form a universal gate set together.

$$n \underset{}{\cdots} \boxed{\alpha} \underset{}{\cdots} m = |0,...,0\rangle \langle 0,...,0| + e^{i\alpha} |1,...,1\rangle \langle 1,...,1|$$
$$n \underset{}{\cdots} \boxed{\alpha} \underset{}{\cdots} m = |+,...,+\rangle \langle +,...,+| + e^{i\alpha} |-,...,-\rangle \langle -,...,-|$$

Wires connect the outputs of one spider with the inputs of other spiders. The identity matrix $\mathbb{1}$ implements the linear map of wires.

$$— = |0\rangle \langle 0| + |1\rangle \langle 1|$$

A yellow box connected by wires indicates a Hadamard generator. Euler decomposition, known as the Hadamard rule (hd), splits a Hadamard generator into a sequence of Z and X spiders [16].

$$—\square— = \boxed{\tfrac{\pi}{2}}\boxed{\tfrac{\pi}{2}}\boxed{\tfrac{\pi}{2}}$$

The swap generator swaps the spiders on a wire and implements the identical linear map as the swap gate in the quantum circuit notation.

$$\asymp = |00\rangle \langle 00| + |01\rangle \langle 10| + |10\rangle \langle 01| + |11\rangle \langle 11|$$

In ZX calculus, bent wires depict the Bell state and the Bell effect and are known as cup and cap.

$$\subset = |00\rangle + |11\rangle$$
$$\supset = \langle 00| + \langle 11|$$

A typical ZX diagram consists of many connected spiders and Hadamard generators. Matrix multiplication composes the linear map of sequentially connected spiders. The tensor product composes the linear map between non-sequential connected spiders and Hadamards, meaning that generators are parallel to each other.

Only topology matters is an important concept in ZX calculus. It states that the linear map between qubits of a ZX diagram remains unchanged as long as its connectivity stays the same. As a consequence, bending wires (e.g. cups and caps) and moving spiders do not change the ZX diagram [13].

2.2 Rewriting Rules

This section introduces the basic rewriting rules of ZX calculus that are outlined in Figure 1 [13,40]. All rules remain valid under colour inversion. We give an example of the application of successive rewriting rules, explained below, on a simple ZX diagram in Figure 2.

Spider fusion (f) Connected spiders of the same colour fuse through modulo-2π addition of their phases. The reverse unfusing operation is always possible, because connecting additional spiders with a phase of $\alpha = 0$ will not change the modulo-2π addition. As a consequence, infinite spiders can be unfused. Figure 2 highlights the fusion of two green non-phase-carrying spiders with their neighbouring phase-carrying spiders.

Local complementation (lc) The local complementation rule [34] originates from graph theory. For all directly connected spiders of a target spider, local complementation connects previously unconnected spiders and disconnects previously connected spiders. Local complementation of the highlighted red spider in the bottom qubit row is illustrated in Figure 2. The two green spiders connected to the highlighted red spider connect via local complementation. Performing a second local complementation at the same red spider would disconnect the two green phase-carrying spiders again. Pivoting describes a series of local complementations.

Colour change (h) Adding Hadamard generators to each input and output inverts the colour of a spider. In Figure 2, all red spiders turn green with the addition of Hadamard generators.

Identity removal (i1, i2) Non-phase-carrying spiders that are directly connected to other spiders function as wires and leave the linear map of qubits unchanged. The identity matrix $\mathbb{1}_{2^m \times 2^n}$ represents the linear map of such spiders. A single wire replaces a phaseless spider with $n = 1$ and $m = 1$. Similarly, two directly connected Hadamard generators cancel each other out and act as a wire. Applying the identity removal rule on the fused diagram in Figure 2 removes all non-phase-carrying spiders that possess one input and one output. Furthermore, identity rules are used to convert a ZX diagram to be graph-like (see Definition 1) by ensuring that spiders always connect with each other through Hadamard generators and by the addition of potentially missing spiders at the input and output.

Bialgebra (b) The bialgebra rule originates from the algebraic commutation relation between the *copy* and the *or* gate. It allows connected and opposite-coloured spiders to move through each other at the cost of potentially adding spiders.

Copy (π, c) π copying moves an input spider that carries the phase $\alpha = \pi$ through an opposite coloured spider to all connected wires while multiplying the phase by -1. If the input spider does not have any input wire ($n = 0$) and the phase is a multiple of π, the opposite coloured spider vanishes. This second rule is referred to as the state copying, because it copies the computational basis through an opposite-coloured spider.
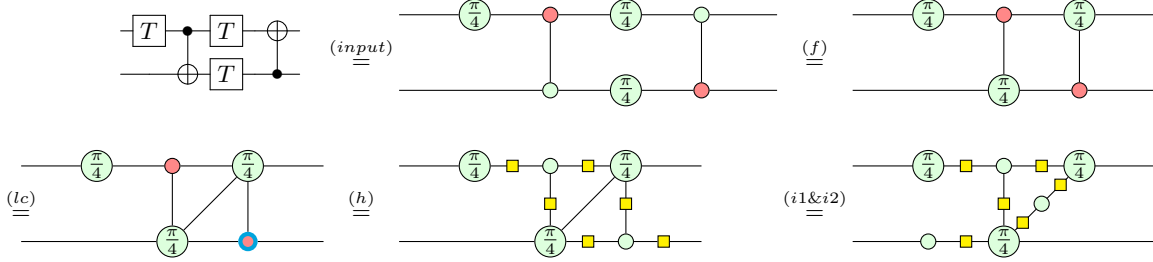
Fig. 2: Successive applications of rewriting rules to a simple ZX diagram (to be read from left to right and top to bottom).

2.3 ZX-based Circuit Optimization

Recent advances in quantum circuit optimization combine the ZX calculus with algorithmic, heuristic or deep learning approaches [15,42,38]. PyZX is a popular Python library to work with ZX calculus and supports state-of-the-art circuit optimization with a focus on T-gate reduction [31,32]. To simplify ZX diagrams using basic rewriting rules, PyZX assumes that the ZX diagrams are graph-like.

Definition 1. *Graph-like ZX diagrams are only composed of Z spiders (green) that are connected by Hadamard wires. Input / Output possesses at most one wire that can only connect to one spider.*

The final diagram in Figure 2 is graph-like. Every ZX diagram can be converted to be graph-like using the h-rule and the identity rules i1 and i2.

Reducing the number of T-gates in a given quantum circuit is crucial because implementing the required quantum error correction on quantum hardware demands significantly more resources compared to Clifford gates [9,14].

The *full reduce algorithm* is the main optimization algorithm of PyZX. It aims to decrease the number of T-gates of a given graph-like ZX diagram by targeting spiders that carry phase a multiple of $\frac{\pi}{2}$ and π [32,15].

After optimizing a ZX diagram, the corresponding quantum circuit needs to be extracted. While converting a quantum circuit into ZX diagram is a straightforward process, the opposite is #P-hard (at least as hard as NP-complete problems) and is known as the circuit extraction problem [6]. Polynomial time algorithms exist for ZX diagrams that are graph-like and preserve the generalized flow [5]. The disadvantage of current circuit extraction algorithms is that the connectivity of spiders in the ZX representation is replicated by two-qubit gates in the resulting quantum circuit.

Recent advances optimize two-qubit gates by reducing the number of edges in a given ZX diagram. Staudacher et al. proposed a heuristic that calculates the cost based on the number of edges after rule application and uses a greedy or stochastic algorithm to select the next rule [40].

This paper focuses on ZX diagram optimization and not on the circuit extraction problem. Therefore, the standard PyZX extraction algorithm is used for all experiments [15].

3   ZX Diagram Optimization

Let **ZX** be the infinite set of all finite ZX diagrams, **QC** the set of quantum circuits and **LM** the set of linear maps of qubits. We have two functions $\alpha : \mathbf{QC} \to \mathbf{ZX}$ and $extract : \mathbf{ZX} \to \mathbf{QC} \cup \{\perp\}$ that convert a quantum circuit into a ZX diagram and conversely. The function $extract$ can map to a special element $\perp$ when it fails to extract a quantum circuit from a diagram. Additionally, we have a function $\gamma : \mathbf{QC} \to \mathbf{LM}$ that maps a quantum circuit to its linear map of qubits.

A *quantum circuit optimization* algorithm is a function $f : \mathbf{QC} \to \mathbf{QC}$ that optimizes some properties of the quantum circuit. We say that f is *semantic-preserving* whenever, for all $q \in \mathbf{QC}$, we have $\gamma(q) = \gamma(f(q))$.

Let $R = \{h, b, lc, f, i1, i2, \pi, c, hd\}$ be the set of rewriting rules presented in Section 2.2. A ZX rewriting rule $r \in R$ is a function $\mathbf{ZX} \to \mathbf{ZX}$ such that the function $extract \circ r \circ \alpha$ is semantic-preserving when the extraction succeeds.

A *ZX-based quantum circuit optimization* algorithm is searching for an extractable ZX diagram that optimizes one or more properties of the quantum circuit. Let $q \in \mathbf{QC}$ be a quantum circuit and $opt : \mathbf{ZX} \to \mathbb{Z} \cup \{\perp\}$ be the optimization function mapping a ZX diagram to a metric (e.g. the number of T-gates, edges or two-qubit gates). The function opt can map the special element $\perp$ when it fails to compute a metric from a ZX diagram. Without loss of generality, we consider that we aim at minimizing opt. The *ZX state-space* of q is a set $W \subseteq \mathbf{ZX}$ such that $w \in W$ if there exists a finite sequence of rewriting rules $r_1, \ldots, r_n$ such that $w = (r_n \circ \ldots \circ r_1 \circ \alpha)(q)$. The set of *solutions* $S \subseteq W$ are all the extractable ZX diagrams in W, that is, $\forall w \in W, \ extract(w) \neq \perp \Leftrightarrow w \in S$. The set of *optimal solutions* is the largest set $O \subseteq S$ such that for all $o \in O, s \in S$, we have $opt(o) \leq opt(s)$.

There are challenges pertaining to the exploration of the ZX state-space. Firstly, real-world quantum circuits result in large ZX diagrams with a high memory demand for every state. Secondly, the state-space to explore is infinite because the ZX rules rewriting system is non-terminating, e.g. unfusing phaseless spiders and colour changing is always possible. Thirdly, to find a solution we must extract a circuit, which is a computationally expensive operation that may fail if the general flow is not preserved [5]. Although it is possible to optimize a metric completely defined on the ZX diagram (e.g. number of T-gates, vertices and edges), other metrics of interest (e.g. number of two-qubits gates, circuit depth and overall gate count) are defined on the extracted circuit. Even though we focus in our experiments on the T-gate and edge count, our approach is general and can be reused for any of those metrics.

In this paper, we rely on iterative deepening depth-first search (IDDFS), which is a simple and efficient optimization algorithm, to tackle these challenges [33]. More advanced optimization algorithms are difficult to use in the context of ZX optimization. For example, constraint-based combinatorial optimization such as linear programming and constraint programming require explicitly encoding the rewriting rules as constraints and do not usually support unbounded rewriting sequences out of the box. Dynamic programming is an interesting approach to avoid re-exploring the same state multiple times, but it requires to store a prohibitively high number of states. We would need to find a more compact representation of states, which we leave for future work.

An interesting aspect of IDDFS is to strike the right balance between exploration and exploitation. It is based on depth-first search (DFS) which is a backtracking algorithm applying rules in sequence until it finds a leaf node, at which point it backtracks to the previous decision made. The problem with DFS is that wrong decisions taken early in the search tree condemn the search strategy to explore large uninteresting subtrees. In the worst-case scenario, DFS reaches a deep leaf node with a state that is unextractable. Instead, IDDFS applies DFS with a depth bound, exploring successive search trees of increasing depth. It is especially useful when we do not have a good heuristic to select the next node to explore, which is the case here, since it is hard to predict which rules might lead to a better solution. Another advantage of IDDFS is to use as much memory as DFS while exploring the search tree in breadth as well. Note that for completeness, we experiment with DFS when evaluating our approach.

A *leaf node* is a ZX diagram $d \in \mathbf{ZX}$ such that one of these conditions is true:

- For all rules $r \in R \setminus \{h\}$ we have $r(d) = d$, that is, no rule besides colour change is able to rewrite the diagram.
- One of the pruning conditions is true.

When we reach a leaf node, we test whether $opt(d)$ is better than the previously found solution, if any. In case of improvement, we check whether the ZX diagram is extractable or not using $extract(d) = \bot$. If it is extractable, we save the ZX diagram as the current best solution.

Pruning conditions reduce the search tree effectively and ensure that the search terminates in finite time. The pruning conditions used throughout this paper are the following:

- *No spider unfusion.* It is always possible to separate one spider into multiple spiders as long as the mod π sum of all involved spiders remains unchanged.
- *Rule bundling.* If a rule can rewrite a given ZX diagram more than once, all possible modifications are performed in a batch, hence generating only one node in the search tree.
- *No colour cycle.* Disallowing consecutive colour changing rule application avoids infinite paths that consist only of recolouring spiders.
- *Global time limit.* After a set time limit is exceeded, the search is terminated and the best solution found so far is returned.

4   Computational Experiments

The proposed DFS and IDDFS approaches are implemented in Python and evaluated against PyZX's implementation of full reduce using a diverse set of quantum circuits [15,37].

Although the search algorithms employed are fairly simple and well-known, their combination with ZX-calculus and the pruning condition implemented in a reproducible and integrated framework is not straightforward. A tight integration with PyZX and Qiskit, as well as the contribution of a Qiskit transpiler pass, ensures the reusability of our approach. The overall project is 7000 lines of code.

Every search instance was executed on a Xeon Gold 6132 clocked at 2.6 GHz and 64 GB of 2400 MHz DDR4 of available RAM running Rocky Linux 8.10 with Python 3.12.

We evaluated the performance of the various algorithms on the complete *set of 100 standard quantum circuits* using the pruning conditions introduced in Section 3. A *global timeout of 1.5 hours* is set for every instance. The rules are ordered such that a change connectivity (e.g. pivoting and local complementation) takes priority over spider count reduction (e.g. fusion and identity removal).

Upon completion of an instance, the optimized quantum circuit can be fed into a transpilation pipeline for further processing. In a first set of experiments, we minimized the T-gate count of ZX diagrams. To demonstrate the generality of our approach, a second set of experiments that minimizes the edge count of ZX diagrams is executed. The full results for every instance, including a comparison with full-reduce and the algorithm runtimes, are available in the supplementary material.

4.1 T-gate Reduction

Solution comparison of IDDFS and DFS
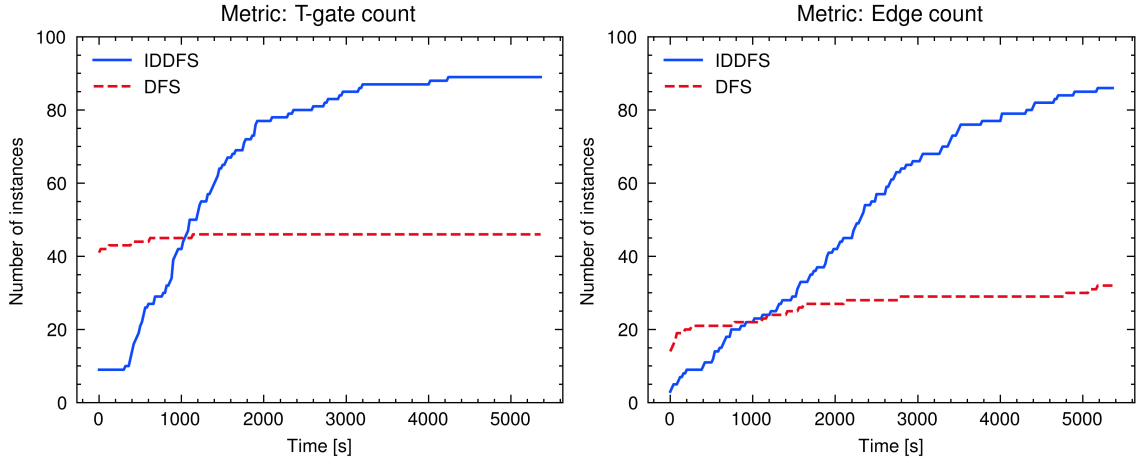
(a) Time evolution against the solution of full reduce.   (b) Time evolution of the best solution found.

The DFS search exhibits poorer performance, only equating full reduce in 46% of the instances, compared to IDDFS search. IDDFS equates full reduce in 89% of the instances. Neither DFS nor IDDFS approaches are able to outperform full reduce within the 1.5 hour time limit. Nevertheless, it should be noted that the DFS search is able to equate full reduce on three circuits on which IDDFS leads to poorer results.

Figure 3a shows the time evolution of the best solution of DFS (red) and IDDFS (blue). As neither DFS nor IDDFS outperform full reduce, only solutions that equate full reduce are shown. DFS almost immediately equals full reduce in 41% of the instances and only equating full reduce on additional 5% of the instances for the remaining 1.5 hours. In contrast, IDDFS requires 16 minutes to level the performance of DFS and equals full reduce in 80% of the instances within the first 60 minutes. Overall, IDDFS equals full reduce on 89% of the instances within the 1.5 hour time limit.

4.2 Edge Reduction

Figure 3b visualizes the time evolution of the best solution of DFS (red) and IDDFS (blue). Within the 1.5 hour time limit, IDDFS is able to find the best solution in 86% of the instances.

DFS results only in the best solution exclusively in 1% of the instances and shares the best solution in 31% of the instances. IDDFS improves the performance of DFS and leads to an exclusively best solution for 54% of the instances and shares its best solution in 32% of the instances.

It should be noted, that despite being designed to reduce the T-gate count, full reduce achieves the exclusively best solution on 13% of the instances and shares a best solution in an additional 2% of the instances.

Compared to the unoptimized quantum circuit, DFS improves the edge count on average by 11%. IDDFS exhibits better performance and reduces the edge count by 22% within the 1.5 hour time limit. Full reduce improves the edge count by 3%. We optimized the edge count to demonstrate the applicability of our algorithm to other metrics. Furthermore, optimizing the edge count is interesting because current state-of-the-art circuit extraction algorithms replicate the connectivity of spiders with two-qubit gates, therefore potentially increasing the circuit depth and two-qubit gate count.

Our results are more contrasted as reducing the edge count did not necessarily translate in a reduction of the of two-qubit gate count. On the contrary, more two-qubit gates are added for the vast majority of instances. DFS results in a higher two-qubit gate count in 41% of the instances and only reduces the two-qubit gate count in 1% of the instances. IDDFS yields to poorer performance and adds two-qubit gates in 85% of the instances and reduces the two-qubit gate count in 1% of the instances. translates in a reduction of the two-qubit gate count. Staudacher et al. showed that they were able to translate an average reduction in the edge count by 29% in a reduction of the two-qubit gate count by 21% [40].

5   Related Work

Recent advances have been made to optimize the T-gate count in quantum circuits. Fault tolerant quantum computing introduces quantum error correction code that increases the resource demand, especially for the T-gate. Improvements in the T-gate count that leverages the quantum error code were achieved with Matroid partitioning [2]. Template based techniques improve the quantum circuit synthesis by reducing the T-gate count and circuit depth [8].

The first proposed optimization strategy using ZX calculus is restricted to Clifford gates [18]. The state-of-the-art optimization algorithm full reduce targets Clifford and T-gates [15,32]. Other techniques optimize the T-gate count through the treatment of Clifford gates and Pauli operators as $\frac{\pi}{4}$ rotations around each other [43]. Additionally, new causal flow preserving optimization techniques ensure the extractability of a quantum circuit from a ZX diagram [24]. An improved T-gate count for arithmetic circuits, e.g. integer multiplication, was found by applying the ZX rewriting rules [29]. Reinforcement learning strategies based on ZX calculus that target the T-gate and two-qubit gate count emerged in recent years [38,39]. Heuristics that target the two-qubit counts ensures the usefulness of ZX calculus for photonic quantum computing and other quantum hardware that does not perform error correction [40]. Other approaches combine heuristics and ZX calculus for the architecture-aware optimization of quantum circuits [21,42].

Heuristic approaches deal with the time complexity involved in quantum circuit optimization. In principle, a heuristic pattern matching algorithm is combined with gate commutation rules to minimize the total gate count [25]. Boolean satisfiability is an exact approach for the optimization of classical circuits. Despite the challenging encoding of quantum gates, advances have been made to bring this approach to quantum circuit optimization [36,7]. Recently, reinforcement learning techniques emerged for quantum circuit optimization and mapping of quantum circuits for specific quantum architectures [20,17]. Gate commutation rules and templates proved also advantageous for the mapping of quantum circuits [26].

6   Conclusion

This paper lays the foundation to apply exhaustive search to ZX diagrams for the optimization of quantum circuits. The combination of the semantics-preserving rewriting rule of ZX calculus with the exhaustive search algorithms *depth-first search (DFS)* and *iterative deepening depth-first search (IDDFS)* enables to target metrics of the ZX diagram or its corresponding quantum circuit without being designed for one specific metric.

Our results indicate that IDDFS is a more effective approach for ZX diagram optimization than DFS. Within the 1.5 hour time limit IDDFS is able to equate state-of-the-art algorithms that reduce the T-gate count in 89% of the instances and competes with novel approaches that reduce the edge count, demonstrating the applicability of our approach.

A Qiskit compiler pass that implements the DFS and IDDFS approach, with configurable pruning conditions and integration with the PyZX library, is available on GitLab (`https://gitlab.com/NetForceExplorer/zx_dfs/-/releases/v1.0-OLA_2025`).

Our results demonstrate that not every reduction in the edge count translates into a reduction in the two-qubit gate count. Upcoming work could focus on the enhancement of the edge count metric to better approximate the two-qubit gate count after circuit extraction. Future efforts should address the scalability issue for large circuits of the IDDFS and DFS based optimization. The principal idea of ZX diagram optimization is to change the connectivity and the fusion of spiders, hence a *limited discrepancy search* could improve the performance [22]. The application of dynamic programming techniques could trade computational performance for higher memory requirements. Finally, bridging the gap from architecture-independent optimization towards quantum architecture-aware optimization could address the execution of real-world quantum circuits on next-generation hardware.

References

1. Aaronson, S., Gottesman, D.: Improved simulation of stabilizer circuits. Physical Review A **70**(5), 052328 (Nov 2004). `https://doi.org/10.1103/PhysRevA.70.052328`
2. Amy, M., Maslov, D., Mosca, M.: 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), 1476–1489 (Oct 2014). `https://doi.org/10.1109/TCAD.2014.2341953`
3. Backens, M.: The ZX-calculus is complete for stabilizer quantum mechanics. New Journal of Physics **16**(9), 093021 (Sep 2014). `https://doi.org/10.1088/1367-2630/16/9/093021`
4. Backens, M.: Making the stabilizer ZX-calculus complete for scalars. Electronic Proceedings in Theoretical Computer Science **195**, 17–32 (Nov 2015). `https://doi.org/10.4204/EPTCS.195.2`
5. Backens, M., Miller-Bakewell, H., de Felice, G., Lobski, L., van de Wetering, J.: There and back again: A circuit extraction tale. https://arxiv.org/abs/2003.01664v3 (Mar 2020). `https://doi.org/10.22331/q-2021-03-25-421`
6. de Beaudrap, N., Kissinger, A., van de Wetering, J.: Circuit Extraction for ZX-Diagrams Can Be #P-Hard. In: Bojańczyk, M., Merelli, E., Woodruff, D.P. (eds.) 49th International Colloquium on Automata, Languages, and Programming (ICALP 2022). Leibniz International Proceedings in Informatics (LIPIcs), vol. 229, pp. 119:1–119:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2022). `https://doi.org/10.4230/LIPIcs.ICALP.2022.119`, `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2022.119`

7. Berent, L., Burgholzer, L., Wille, R.: Towards a SAT Encoding for Quantum Circuits: A Journey From Classical Circuits to Clifford Circuits and Beyond. In: DROPS-IDN/v2/Document/10.4230/LIPIcs.SAT.2022.18. Schloss-Dagstuhl - Leibniz Zentrum für Informatik (2022). `https://doi.org/10.4230/LIPIcs.SAT.2022.18`

8. Biswal, L., Das, R., Bandyopadhyay, C., Chattopadhyay, A., Rahaman, H.: A template-based technique for efficient Clifford+T-based quantum circuit implementation. Microelectronics Journal **81**, 58–68 (Nov 2018). `https://doi.org/10.1016/j.mejo.2018.08.011`

9. Campbell, E.T., Terhal, B.M., Vuillot, C.: Roads towards fault-tolerant universal quantum computation. Nature **549**(7671), 172–179 (Sep 2017). `https://doi.org/10.1038/nature23460`

10. Chi-Chih Yao, A.: Quantum circuit complexity. In: Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science. pp. 352–361 (Nov 1993). `https://doi.org/10.1109/SFCS.1993.366852`

11. Coecke, B., Duncan, R.: Interacting Quantum Observables. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) Automata, Languages and Programming. pp. 298–310. Springer, Berlin, Heidelberg (2008). `https://doi.org/10.1007/978-3-540-70583-3_25`

12. Coecke, B., Duncan, R.: Interacting Quantum Observables. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) Automata, Languages and Programming. pp. 298–310. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2008). `https://doi.org/10.1007/978-3-540-70583-3_25`

13. Coecke, B., Kissinger, A.: Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning. Cambridge University Press, Cambridge (2017). `https://doi.org/10.1017/9781316219317`

14. Ding, Y., Holmes, A., Javadi-Abhari, A., Franklin, D., Martonosi, M., Chong, F.T.: Magic-State Functional Units: Mapping and Scheduling Multi-Level Distillation Circuits for Fault-Tolerant Quantum Architectures. https://arxiv.org/abs/1809.01302v1 (Sep 2018). `https://doi.org/10.1109/MICRO.2018.00072`

15. Duncan, R., Kissinger, A., Perdrix, S., van de Wetering, J.: Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus. Quantum **4**, 279 (Jun 2020). `https://doi.org/10.22331/q-2020-06-04-279`

16. Duncan, R., Perdrix, S.: Graph States and the Necessity of Euler Decomposition. In: Ambos-Spies, K., Löwe, B., Merkle, W. (eds.) Mathematical Theory and Computational Practice. pp. 167–177. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2009). `https://doi.org/10.1007/978-3-642-03073-4_18`

17. Elsayed Amer, N., Gomaa, W., Kimura, K., Ueda, K., El-Mahdy, A.: On the optimality of quantum circuit initial mapping using reinforcement learning. EPJ Quantum Technology **11**(1), 19 (Dec 2024). `https://doi.org/10.1140/epjqt/s40507-024-00225-1`

18. Fagan, A., Duncan, R.: Optimising Clifford Circuits with Quantomatic. Electronic Proceedings in Theoretical Computer Science **287**, 85–105 (Jan 2019). `https://doi.org/10.4204/EPTCS.287.5`

19. Fowler, A.G., Mariantoni, M., Martinis, J.M., Cleland, A.N.: Surface codes: Towards practical large-scale quantum computation. https://arxiv.org/abs/1208.0928v2 (Aug 2012). `https://doi.org/10.1103/PhysRevA.86.032324`

20. Fösel, T., Niu, M.Y., Marquardt, F., Li, L.: Quantum circuit optimization with deep reinforcement learning (2021), `https://arxiv.org/abs/2103.07585`

21. Gogioso, S., Yeung, R.: Annealing Optimisation of Mixed ZX Phase Circuits (Nov 2023). `https://doi.org/10.48550/arXiv.2206.11839`

22. Harvey, W.D., Ginsberg, M.L.: Limited discrepancy search. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI). pp. 607–615 (1995)

23. Hassija, V., Chamola, V., Goyal, A., Kanhere, S.S., Guizani, N.: Forthcoming applications of quantum computing: Peeking into the future. IET Quantum Communication **1**(2), 35–41 (2020). `https://doi.org/10.1049/iet-qtc.2020.0026`

24. Holker, C.: Causal flow preserving optimisation of quantum circuits in the ZX-calculus (Jan 2024). `https://doi.org/10.48550/arXiv.2312.02793`

25. Iten, R., Moyard, R., Metger, T., Sutter, D., Woerner, S.: Exact and Practical Pattern Matching for Quantum Circuit Optimization. ACM Transactions on Quantum Computing **3**(1), 4:1–4:41 (Jan 2022). `https://doi.org/10.1145/3498325`

26. Itoko, T., Raymond, R., Imamichi, T., Matsuo, A.: Optimization of quantum circuit mapping using gate transformation and commutation. Integration **70**, 43–50 (Jan 2020). `https://doi.org/10.1016/j.vlsi.2019.10.004`

27. Jeandel, E., Perdrix, S., Vilmart, R.: 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. LICS '18, Association for Computing Machinery, New York, NY, USA (Jul 2018). `https://doi.org/10.1145/3209108.3209131`

28. Jeandel, E., Perdrix, S., Vilmart, R.: Diagrammatic Reasoning beyond Clifford+T Quantum Mechanics. In: Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science. pp. 569–578. LICS '18, Association for Computing Machinery, New York, NY, USA (Jul 2018). `https://doi.org/10.1145/3209108.3209139`

29. Joshi, A., Kairali, A., Raju, R., Athreya, A., P, R.M., Vishwakarma, S., Ganguly, S.: Quantum Circuit Optimization of Arithmetic circuits using ZX Calculus (Jun 2023). `https://doi.org/10.48550/arXiv.2306.02264`

30. Khairy, S., Shaydulin, R., Cincio, L., Alexeev, Y., Balaprakash, P.: Reinforcement-Learning-Based Variational Quantum Circuits Optimization for Combinatorial Problems. ArXiv (Nov 2019)

31. Kissinger, A., van de Wetering, J.: PyZX: Large Scale Automated Diagrammatic Reasoning. Electronic Proceedings in Theoretical Computer Science **318**, 229–241 (May 2020). `https://doi.org/10.4204/EPTCS.318.14`

32. Kissinger, A., van de Wetering, J.: Reducing T-count with the ZX-calculus. Physical Review A **102**(2), 022406 (Aug 2020). `https://doi.org/10.1103/PhysRevA.102.022406`

33. Korf, R.E.: Depth-first iterative-deepening: An optimal admissible tree search. Artificial Intelligence **27**(1), 97–109 (Sep 1985). `https://doi.org/10.1016/0004-3702(85)90084-0`

34. Kotzig, A.: Eulerian lines in finite 4-valent graphs and their transformations. Theory of Graphs pp. 219–230 (1968)

35. Maslov, D., Dueck, G.W., Miller, D.M., Negrevergne, C.: Quantum Circuit Simplification and Level Compaction. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **27**(3), 436–444 (Mar 2008). `https://doi.org/10.1109/TCAD.2007.911334`

36. Meuli, G., Soeken, M., De Micheli, G.: SAT-based {CNOT, T} Quantum Circuit Synthesis. In: Kari, J., Ulidowski, I. (eds.) Reversible Computation. pp. 175–188. Lecture Notes in Computer Science, Springer International Publishing, Cham (2018). `https://doi.org/10.1007/978-3-319-99498-7_12`

37. Nam, Y., Ross, N.J., Su, Y., Childs, A.M., Maslov, D.: GitHub - meamy/t-par: A quantum circuit optimizer based on sum-over-paths representations. https://github.com/meamy/t-par (Nov 2019)

38. Nägele, M., Marquardt, F.: Optimizing zx-diagrams with deep reinforcement learning (Nov 2023)

39. Riu, J., Nogué, J., Vilaplana, G., Garcia-Saez, A., Estarellas, M.P.: Reinforcement Learning Based Quantum Circuit Optimization via ZX-Calculus (Dec 2023)

40. Staudacher, K., Guggemos, T., Grundner-Culemann, S., Gehrke, W.: Reducing 2-QuBit Gate Count for ZX-Calculus based Quantum Circuit Optimization. In: Electronic Proceedings in Theoretical Computer Science. vol. 394, pp. 29–45 (Nov 2023). `https://doi.org/10.4204/EPTCS.394.3`

41. Venturelli, D., Do, M., Rieffel, E., Frank, J.: Compiling quantum circuits to realistic hardware architectures using temporal planners. Quantum Science and Technology **3**(2), 025004 (Feb 2018). `https://doi.org/10.1088/2058-9565/aaa331`

42. Winderl, D., Huang, Q., Mendl, C.B.: A recursively partitioned approach to architecture-aware ZX Polynomial synthesis and optimization. In: 2023 IEEE International Conference on Quantum Computing and Engineering (QCE). pp. 837–847 (Sep 2023). `https://doi.org/10.1109/QCE57702.2023.00098`

43. Zhang, F., Chen, J.: Optimizing T gates in Clifford+T circuit as $\pi/4$ rotations around Paulis (Mar 2019)