Towards Faster Quantum Circuit Simulation Using Graph Decompositions, GNNs and Reinforcement Learning

Alexander Koziell-Pipe* **Richie Yeung*** **Matthew Sutcliffe***
Department of Computer Science
University of Oxford
Oxford, UK
[firstname.lastname]@cs.ox.ac.uk

Abstract

In this work, we train a graph neural network with reinforcement learning to more efficiently simulate quantum circuits using the ZX-calculus. Our experiments show a marked improvement in simulation efficiency using the trained model over existing methods that do not incorporate AI. In this way, we demonstrate a machine learning model that can reason effectively within a mathematical framework such that it enhances scientific research in the important domain of quantum computing.

In the present-day 'Noisy Intermediate Scale' (NISQ) era of quantum computing [30], quantum resources are still largely limited. Given this limit on quantum resources, being able to simulate quantum computations efficiently and at scale on classical hardware can accelerate quantum computing research and set a standard for benchmarking quantum computers.

While in general quantum circuit simulation can be #P-hard [28], a subclass of quantum circuits known as stabiliser circuits can be simulated in polynomial time with respect to size [1]. Hence a technique for simulating quantum circuits is to decompose them into an ensemble of efficiently simulated stabiliser circuits, the aggregation of which simulates the same computation as the original circuit. Decompositions are calculated iteratively, where sub-circuits are decomposed in a sequential manner until the ensemble of stabiliser circuits is achieved. At each step in the decomposition, the choice of sub-circuit to decompose can greatly affect the number of stabiliser circuits that need to be simulated at the end – in the worst case, this is exponential with respect to the number of a certain type of gate, called a T-gate, in the original circuit.

In this work, we formulate the challenge of choosing good sub-circuit decompositions as a reinforcement learning problem, where an agent learns to make decisions in a combinatorially large action space. To facilitate this, we utilise a mathematical framework known as the ZX-calculus, in which quantum circuits are represented as graphs and reasoning amounts to a set of rules allowing one graph to be transformed into another. Formulating our problem in terms of graphs enables the use of Graph Neural Networks (GNNs), which have seen promising applications in other scientific domains including bioinformatics [39], social networks [17], and combinatorial optimisation [7].

We show that, for classes of quantum circuit known not to be efficiently classically simulated, our GNN agent trained using reinforcement learning achieves significantly more efficient decompositions compared to current methods that do not incorporate AI. Moreover, we show that additional algebraic rules can be added to the decomposition strategy to achieve even further improvements in simulation efficiency. As such, our model demonstrates the ability of an AI-agent to reason about a task that typically requires strong mathematical reasoning skills and a deep understanding of the algebraic

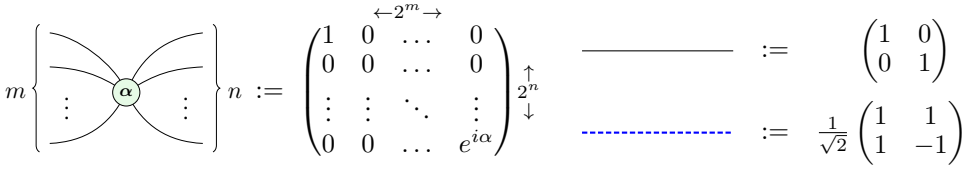
*Co-first authors

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

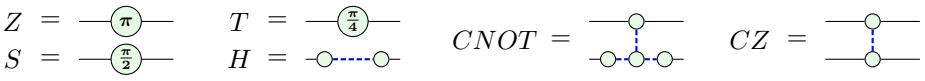
structures underlying quantum circuits. Furthermore, it improves our capacity to conduct scientific research in the increasingly important field of quantum computing.

1 ZX-Calculus

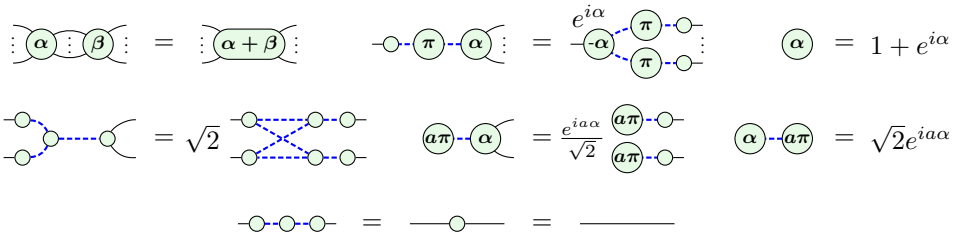
Quantum algorithms can be expressed graphically in circuit notation, with quantum gates composed together in a time-ordered structure. The ZX-calculus [11, 12, 23, 36], offers a powerful alternative which has proven effective for reasoning about quantum computing problems such as circuit compilation and optimisation [5, 8, 13–15, 18, 27, 29] as well as classical simulation [2, 9, 10, 22, 24, 26, 33–35]. In our work we use a variation of the ZX-calculus comprised of graphs whose vertices, called *spiders*, are labelled by a real number $\in [0, 2\pi)$ (the *phase*) and two types of edges:



The way spiders are wired together by edges in a ZX-diagram with m inputs and n outputs determines a matrix in $\mathbb{C}^{2^n \times 2^m}$. Furthermore, wiring the inputs of one diagram to the outputs of another amounts to multiplication of their respective matrices, while juxtaposing two diagrams in parallel amounts to taking the Kronecker product. Indeed, for arbitrary m, n the ZX-calculus is sufficient to express any matrix in $\mathbb{C}^{2^n \times 2^m}$, hence any quantum circuit acting on qubits. In particular, standard gates in quantum computing may be expressed as ZX-diagrams:



Note that when no number is present on a spider, the phase is implicitly taken to equal 0. Diagrams may be deformed arbitrarily and still represent the same quantum computation, provided the graph topology is conserved. They may also be modified using *rewrite rules*, which express how sub-diagrams may be replaced without changing the semantics (the matrix they represent) [36]:



Note that $\alpha, \beta \in \mathbb{R}$, $a \in \{0, 1\}$ and addition is taken modulo 2π in the above diagrams. Rewrite rules can be used to simplify ZX representations of quantum circuits, an example of which may be found in appendix section B.1.

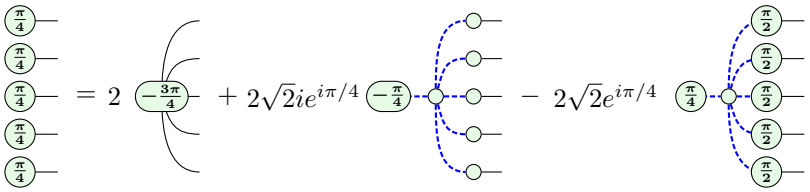
In the literature, there is nomenclature for certain important classes of ZX-diagram. Spiders whose phases are multiples of $\frac{\pi}{2}$ are referred to as *Clifford spiders*, hence diagrams containing only Clifford spiders are referred to as *Clifford diagrams*, also known as *stabiliser diagrams*. Spiders whose phases are an odd multiple of $\frac{\pi}{4}$ are often referred to as *T-spiders*, hence we call diagrams whose spiders only have multiple of $\frac{\pi}{4}$ phases *Clifford+T diagrams*. Moreover, we call diagrams with neither inputs nor outputs *closed diagrams*. Clifford+T diagrams are sufficient for approximating any quantum computation to arbitrary accuracy [25], but cannot be classically simulated efficiently. Clifford diagrams, on the other hand, can be classically simulated efficiently but are not universal for quantum computing. This manifests in that Clifford diagrams containing N spiders may be simulated in $O(N^3)$ operations [22], whereas Clifford+T diagrams require operations exponential with the number of T-spiders. Furthermore, while the rewrite rules above are sufficient to reduce any closed Clifford ZX-diagram to a single scalar value, more advanced techniques (such as the graph decompositions described below) are required to compute the scalar of a closed Clifford+T diagram without resorting to matrix calculations.

2 Circuit Simulation via Graph Decompositions

Where near-term quantum hardware is insufficient for computing quantum circuits of non-trivial scale, the use of classical simulation can be very helpful in verifying their behaviour. Specifically, there is *weak* simulation, wherein a quantum circuit is emulated to provide some probabilistic output, and *strong* simulation, where the probability of a particular measurement outcome is determined. The latter is strictly more powerful as it can be used to achieve the former, and is the focus of our work.

Strong simulation of a quantum circuit can be performed by first representing it as a ZX-diagram, then reducing it to a scalar number via rewrite rules – this scalar represents the probability amplitude of the quantum computation. Where the rewrite rules are insufficient, *decompositions* may be employed to remove problematic sub-diagrams at the cost of splitting the original diagram into a weighted sum of diagrams. For Clifford+T diagrams, one state of the art decomposition used for classical simulation is the $|\mathrm{magic}_5\rangle$ decomposition, introduced by Kissinger et al. [24]:

Lemma 1. *The $|\mathrm{magic}_5\rangle$ decomposition [24]:*



exchanges a set of 5 T-spiders for 3 partial stabiliser terms.

This decomposition removes 4 T-spiders at the cost of replacing a single graph term with 3 terms. Efficient decompositions for Clifford+T diagrams remove hard-to-simulate T-spiders while introducing as few new terms as possible in the weighted sum. We quantify this efficiency via the *decomposition efficiency coefficient* α, defined as follows:

Definition 1. *The efficiency of a particular decomposition can be measured via:*

$$\alpha := \frac{\log_2 N}{t}$$

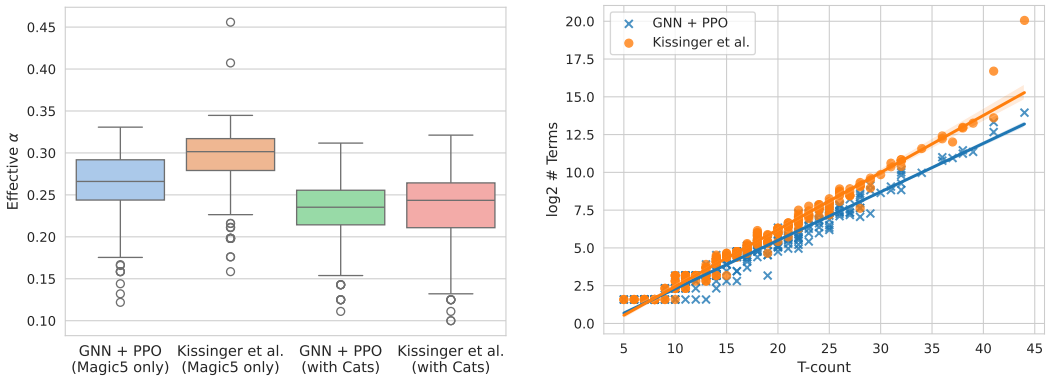
where N is the number of terms produced and t is the number of T-spiders removed by the decomposition. The overall efficiency of a sequence of decompositions and diagram rewrites, $\alpha_{\text{effective}}$, can be measured similarly.

A lower α means a more efficient decomposition. For the $|\mathrm{magic}_5\rangle$ decomposition of equation (1), the efficiency is $\alpha \approx 0.396$. In practice, the decomposition is applied to a sub-circuit, and after diagram rewrites have been applied to the resulting terms, the number of T-spiders remaining may be reduced even further. This can lead to an $\alpha_{\text{effective}}$ far lower than 0.396.

The present state-of-the-art-algorithm [24] deterministically applies efficient structure-specific decompositions (see appendix A) wherever applicable, relying on the $|\mathrm{magic}_5\rangle$ decomposition only when these structures are no longer found. In each case, this algorithm selects the 5 T-spiders upon which to apply this decomposition at random. However, we emphasise that this choice of 5 T-spiders greatly influences the effective α during a sequence of decompositions and diagram rewrites. As such, selecting spiders that lead to more efficient decompositions, thus yielding fewer stabiliser terms to simulate overall can significantly reduce the computational cost of strong simulation. It is this problem of selecting spiders giving more efficient decompositions that we tackle using AI.

3 Experiments

Data Generation Training, validation and test data is generated using PyZX: the Python library for quantum circuit rewriting and optimisation using the ZX-calculus [21]. We generate three different types of ZX-diagrams: 1. Clifford+T quantum circuits, 2. grid-like diagrams, and 3. random graphs generated using the $G(n, m)$ Erdös-Rényi model [16]. Generating random samples from these 3 classes of diagram requires specifying parameters determining the diagram size and phases that appear on the spiders. Specific parameter details used for generating the data can be found in the appendix section C.1.



(a) Distribution of decomposition efficiency over the entire test dataset

(b) Number of stabiliser terms to simulate after decomposition against T-spider count

Figure 1: Comparison of our trained model versus Kissinger et al. [24]

GNN Architecture We use a graph attention network (GAT) [38]-based architecture. For the reinforcement learning algorithm used, the full architecture is divided into a features extractor, policy network and value network. The features extractor processes the input graph. The output of the features extractor is then fed into two separate networks: a policy network, which outputs a probability distribution used to sample vertices for the graph decomposition; and a value network, which assesses the relative value of the current state in the reinforcement learning environment. The value network is only used during training and is not required at inference time.

The features extractor consists of 8 GAT layers each with 4 attention heads and embedding dimension of 64. The policy and value networks follow a transformer-style architecture, consisting of blocks of alternating GAT layers with MLP layers. Residual connections, GELU activations[19], Graph normalisation layers [6], and layer normalisation [4] layers are used. We note the similarity of the policy and value network architectures to a standard transformer architecture [37], with attention layers replaced with GAT layers, and some layer normalisations replaced with graph normalisation. For further details on the architecture, see appendix section C.2.

Reinforcement Learning Setup We train the model using the Proximal Policy Optimisation (PPO) reinforcement learning (RL) algorithm [31] with an adapted version of generalized advantage estimation [32]. Observations in the RL environment are graphs, actions are vertices to which the $|\mathrm{magic}_5\rangle$ decomposition (1) is applied, and rewards are the effective α-efficiencies of applying $|\mathrm{magic}_5\rangle$ to these particular vertices. Further details are given in the appendix section C.3. Data is sampled randomly during training, and intra-training performance is assessed on a validation dataset. We save model weights achieving the best performance on the validation set during a random hyperparameter search; hyperparameters for the top performing weights are listed in table 1, appendix section C.4.

Evaluation & Results We evaluate the models on an unseen test dataset. The best model obtains a mean effective α of 0.263: a marked improvement over selecting the vertices for the decomposition randomly as in Kissinger et al. [24], which achieves 0.293 on the same data. Note that an asymptotic decrease in α leads to an exponential factor speed-up. As an additional investigation, we compare efficiency coefficients when augmenting both methods with an additional set of decompositions, called the $|\mathrm{cat}_n\rangle$ decompositions (see appendix A). In both cases, the decompositions are applied according to the algorithm in Kissinger et al. [24] which is the best, to our knowledge, heursitics-based algorithm using $|\mathrm{cat}_n\rangle$ and $|\mathrm{magic}_5\rangle$. In this experiment, the model achieves a mean effective α of 0.232, versus 0.235 for [24]. This improvement in effective α is highlighted by figure 1a. These results are further summarised appendix D.

Moreover, when looking at the number of stabiliser terms to simulate after decomposition, our model observes better scaling behavior as the number of T-spiders, which comprise the non-stabiliser components of the ZX-diagrams, increases – see figure 1b. We hypothesise that this is because the message passing performed by the graph neural network permits, within a limited neighborhood, broader contextual information about the diagram to be taken into account when choosing the site of a decomposition, whereas the heuristic method of [24] does not.

4 Discussion

Our experiments have shown that a machine learning model can be perform effective mathematical reasoning, with applications to the domain of quantum computing. This was enabled by an algebraic framework: the ZX-calculus, which allowed the task of simulating quantum circuits to be formulated in terms of graphs, making the problem amenable to graph neural networks. Furthermore, the algebraic nature of the ZX-calculus provided a way of designing a reinforcement learning environment in which to learn the circuit simulation task. The trained model showed a marked improvement in simulation efficiency over existing methods without the use of AI.

These initial results are extremely promising: our methodology could be extended to include a broader set of decompositions into the model's action space. Recent work has shown that heuristics-based applications of decompositions, such as in [2, 3, 34], are remarkably effective for a broad range of quantum circuits. We also note that the ZX-calculus has applications to many other problems in quantum computing beyond circuit simulation. This suggests that similar approaches applying AI to other pertinent areas of quantum computing research, such as circuit optimisation and error correction, could be facilitated by the ZX-calculus in the same manner. Typically, these problems are solved by domain experts due to a solid understanding of the mathematics required. Our work suggests, however, that given a sufficient framework within which to perform reasoning, a machine learning model can learn to solve these mathematics-intensive problems. Indeed, it is clear that the ability of AI to solve problems requiring a high-level mathematical understanding can significantly enhance research and engineering across a broad range of scientific domains.

Acknowledgments and Disclosure of Funding

Alexander Koziell-Pipe and Richie Yeung would like to thank Simon Harrison for his generous support via the Wolfson Harrison UKRI Quantum Foundation Scholarship, as well as the enthusiasm he shows toward their research. Alexander Koziell-Pipe and Richie Yeung are part-funded by the EPSRC.

References

[1] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004.

[2] Wira Azmoon Ahmad. Efficient Heuristics for Classical Simulation of Quantum Circuits Using ZX-Calculus. Master's thesis, University of Oxford, 2024.

[3] Wira Azmoon Ahmad and Matthew Sutcliffe. Dynamic t-decomposition for classical simulation of quantum circuits. *[Preprint]*, 2024.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL `https://arxiv.org/abs/1607.06450`.

[5] Agustín Borgna, Simon Perdrix, and Benoît Valiron. Hybrid quantum-classical circuit simplification with the ZX-calculus. In Hakjoo Oh, editor, *Programming Languages and Systems*, pages 121–139, Cham, 2021. Springer International Publishing. doi: 10.1007/978-3-030-89051-3_8.

[6] Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-Yan Liu, and Liwei Wang. Graphnorm: A principled approach to accelerating graph neural network training, 2021. URL `https://arxiv.org/abs/2009.03294`.

[7] Quentin Cappart, Didier Chételat, Elias Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks, 2022. URL `https://arxiv.org/abs/2102.09544`.

[8] Francois Charton, Alexandre Krajenbrink, Konstantinos Meichanetzidis, and Richie Yeung. Teaching small transformers to rewrite ZX diagrams. In *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23*, 2023. URL `https://openreview.net/forum?id=btQ7Bt1NLF`.

[9] Julien Codsi. Cutting-Edge Graphical Stabiliser Decompositions for Classical Simulation of Quantum Circuits. Master's thesis, University of Oxford, 2022. URL https://www.cs.ox.ac.uk/people/aleks.kissinger/theses/codsi-thesis.pdf.

[10] Julien Codsi and John van de Wetering. Classically Simulating Quantum Supremacy IQP Circuits trough a Random Graph Approach. *arXiv preprint arXiv:2212.08609*, 2022.

[11] Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, 2011.

[12] Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017.

[13] Alexander Cowtan, Silas Dilkes, Ross Duncan, Will Simmons, and Seyon Sivarajah. Phase Gadget Synthesis for Shallow Circuits. In Bob Coecke and Matthew Leifer, editors, *Proceedings 16th International Conference on Quantum Physics and Logic, Chapman University, Orange, CA, USA., 10-14 June 2019*, volume 318 of *Electronic Proceedings in Theoretical Computer Science*, pages 213–228. Open Publishing Association, 2020. doi: 10.4204/EPTCS.318.13.

[14] Niel de Beaudrap, Xiaoning Bian, and Quanlong Wang. Fast and Effective Techniques for T-Count Reduction via Spider Nest Identities. In Steven T. Flammia, editor, *15th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2020)*, volume 158 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:23, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. ISBN 978-3-95977-146-7. doi: 10.4230/LIPIcs.TQC.2020.11.

[15] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John van de Wetering. Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus. *Quantum*, 4:279, 6 2020. ISSN 2521-327X. doi: 10.22331/q-2020-06-04-279.

[16] Paul L. Erdos and Alfréd Rényi. On random graphs. i. *Publicationes Mathematicae Debrecen*, 2022. URL https://api.semanticscholar.org/CorpusID:253789267.

[17] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation, 2019. URL https://arxiv.org/abs/1902.07243.

[18] Stefano Gogioso and Richie Yeung. Annealing optimisation of mixed zx phase circuits. In Stefano Gogioso and Matty Hoban, editors, *Proceedings 19th International Conference on Quantum Physics and Logic, Wolfson College, Oxford, UK, 27 June - 1 July 2022*, volume 394 of *Electronic Proceedings in Theoretical Computer Science*, pages 415–431. Open Publishing Association, 2023. doi: 10.4204/EPTCS.394.20.

[19] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023. URL https://arxiv.org/abs/1606.08415.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Aleks Kissinger and John van de Wetering. Pyzx: Large scale automated diagrammatic reasoning. *arXiv preprint arXiv:1904.04735*, 2019.

[22] Aleks Kissinger and John van de Wetering. Simulating quantum circuits with zx-calculus reduced stabiliser decompositions. *Quantum Science and Technology*, 7(4):044001, July 2022. ISSN 2058-9565. doi: 10.1088/2058-9565/ac5d20. URL http://dx.doi.org/10.1088/2058-9565/ac5d20.

[23] Aleks Kissinger and John van de Wetering. *Picturing Quantum Software: An Introduction to the ZX-Calculus and Quantum Compilation*. Preprint, 2024.

[24] Aleks Kissinger, John van de Wetering, and Renaud Vilmart. Classical simulation of quantum circuits with partial and graphical stabiliser decompositions. In *17th Conference on the Theory of Quantum Computation, Communication and Cryptography*, 2022.

[25] A Yu Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191, 1997.

[26] Mark Koch, Richie Yeung, and Quanlong Wang. Contraction of zx diagrams with triangles via stabiliser decompositions. *Physica Scripta*, 2024. URL http://iopscience.iop.org/article/10.1088/1402-4896/ad6fd8.

[27] Tommy McElvanney and Miriam Backens. Flow-preserving ZX-calculus Rewrite Rules for Optimisation and Obfuscation. In Shane Mansfield, Benoit Valîron, and Vladimir Zamdzhiev, editors, *Proceedings of the Twentieth International Conference on Quantum Physics and Logic, Paris, France, 17-21st July 2023*, volume 384 of *Electronic Proceedings in Theoretical Computer Science*, pages 203–219. Open Publishing Association, 2023. doi: 10.4204/EPTCS.384.12.

[28] Ramis Movassagh. The hardness of random quantum circuits. *Nature Physics*, 19(11):1719–1724, 2023.

[29] Maximilian Nägele and Florian Marquardt. Optimizing zx-diagrams with deep reinforcement learning. *Machine Learning: Science and Technology*, 5(3):035077, sep 2024. doi: 10.1088/2632-2153/ad76f7. URL https://dx.doi.org/10.1088/2632-2153/ad76f7.

[30] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, August 2018. ISSN 2521-327X. doi: 10.22331/q-2018-08-06-79. URL http://dx.doi.org/10.22331/q-2018-08-06-79.

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

[32] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018. URL https://arxiv.org/abs/1506.02438.

[33] Matthew Sutcliffe. Smarter k-partitioning of zx-diagrams for improved quantum circuit simulation. *arXiv preprint arXiv:2409.00828*, 2024. URL https://arxiv.org/abs/2409.00828.

[34] Matthew Sutcliffe and Aleks Kissinger. Procedurally optimised zx-diagram cutting for efficient t-decomposition in classical simulation. *Electronic Proceedings in Theoretical Computer Science*, 406:63–78, August 2024. ISSN 2075-2180. doi: 10.4204/eptcs.406.3. URL http://dx.doi.org/10.4204/EPTCS.406.3.

[35] Matthew Sutcliffe and Aleks Kissinger. Fast classical simulation of quantum circuits via parametric rewriting in the zx-calculus. *arXiv preprint arXiv:2403.06777*, 2024. URL https://arxiv.org/abs/2403.06777.

[36] John van de Wetering. Zx-calculus for the working quantum computer scientist, 2020.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

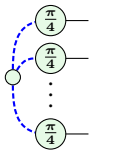
[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018. URL https://arxiv.org/abs/1710.10903.

[39] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks, 2018. URL https://arxiv.org/abs/1806.03536.

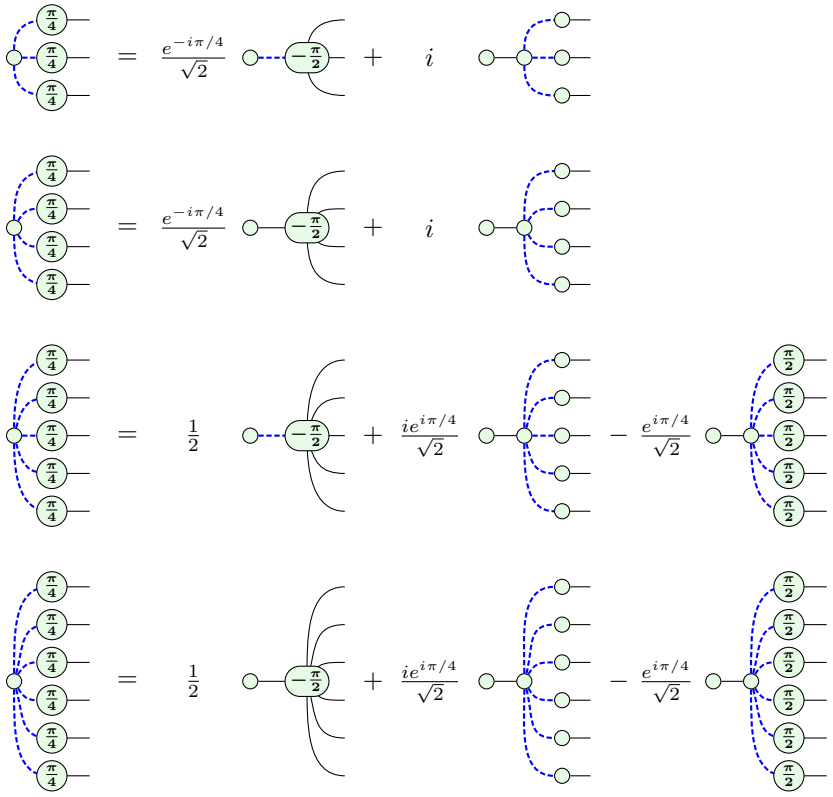
Appendix

A Cat state decompositions

A $|\mathrm{cat}_n\rangle$ state is defined as follows:

$$|\mathrm{cat}_n\rangle := \frac{1}{\sqrt{2}}$$  $$n$$

Such states can be decomposed more efficiently than 'magic' $|\mathrm{T}\rangle^{\otimes n}$ states. In particular, the best known $|\mathrm{cat}_3\rangle$ to $|\mathrm{cat}_6\rangle$ decompositions are as follows [24]:



These respectively achieve decomposition efficiencies of:

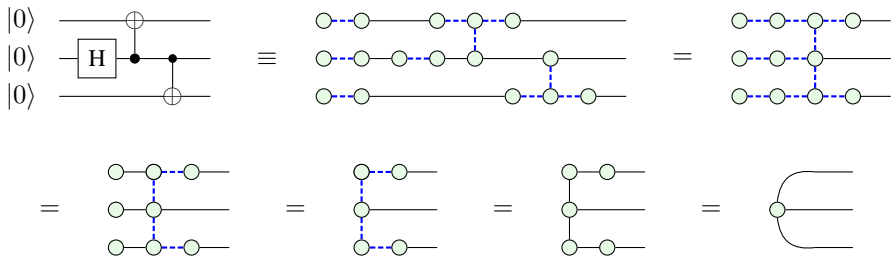
$$\begin{aligned}
\alpha_{|\mathrm{cat}_3\rangle} &\approx 0.333, \\
\alpha_{|\mathrm{cat}_4\rangle} &= 0.250, \\
\alpha_{|\mathrm{cat}_5\rangle} &\approx 0.317, \\
\alpha_{|\mathrm{cat}_6\rangle} &\approx 0.264.
\end{aligned}$$

$|\mathrm{cat}_n\rangle$ states of larger n may be decomposed into sums of the above, with an asymptotic ($n \to \infty$) efficiency equivalent to that of the $|\mathrm{magic}_5\rangle$ decomposition, namely $\alpha \approx 0.396$ [24].

B ZX-Calculus Examples

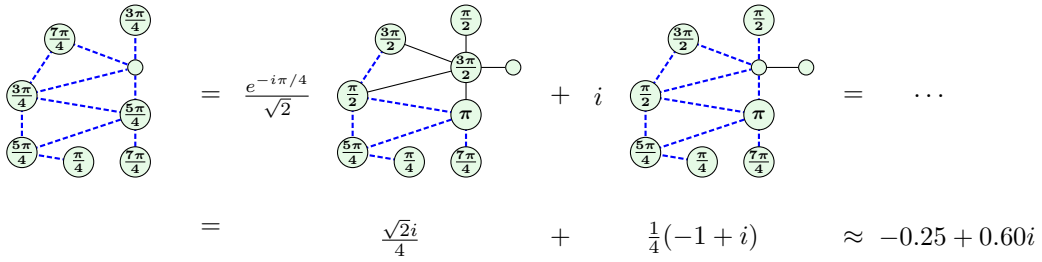
B.1 Simplification Example

The following example [36] demonstrates how a quantum circuit may be translated into a ZX-diagram and subsequently simplified via applications of the rewriting rules. The ZX representation of the same circuit is device-agnostic, highlights the symmetries of the quantum circuit, and can often be extracted to an equivalent circuit with fewer quantum gates:



B.2 Decomposition Example

As an illustrative example, the following shows a fully reduced Clifford+T ZX-diagram and how, with the application of a single $|\mathrm{cat}_4\rangle$ decomposition, it may be reduced to two terms which are each reducible (via the rewriting rules) to a scalar:



C Experiment Details

C.1 Training Data Parameters

We specify the parameters used in generating the training data below. Note that the floor function is applied to any non-integer samples drawn from the below probability distributions.

The Clifford+T circuits are generated to have a number of qubits between 20 and 30 sampled from a clipped normal distribution $\mathcal{N}(20, 25)$[2], and a qubit-dependent depth uniformly sampled from *num_qubits* $\cdot\ (7.5 + \mathcal{U}(0, 67.5))$.

The grid diagrams are of width sampled from a clipped normal distribution $\mathcal{N}(7.5, 1)$ between 7 and 10 nodes and a height sampled from $\mathcal{N}(5.5, 1)$ between 5 and 9 nodes.

The random graphs have a number of vertices sampled from $\mathcal{N}(25, 4)$ clipped between 20 and 32, and a number of edges sampled from $|V| \cdot (|V| - 1) \cdot (0.25 + \mathcal{U}[0, 0.15))$.

C.2 Model Architecture

The features extractor consists of 8 GAT layers each with 4 attention heads and embedding dimension of 64. Residual connections are placed across each GAT layer, after which ReLU activations are applied. Graph normalisation layers[6] are placed between each ReLU activation and the next GAT layer.

The policy and value networks follow a transformer-style architecture, consisting of blocks of alternating GAT layers with MLP layers. Residual connections are placed across each GAT and each MLP layer, graph normalisation is placed before each GAT layer, and layer normalisation[4] is placed before each MLP layer. GELU activations are placed after each GAT and MLP layer. Note the similarity to a standard transformer architecture[37], with attention layers replaced with GAT layers, and some layer normalisations replaced with graph normalisation. Each of the policy and value network consist of 8 such GAT + MLP blocks with embedding dimension 256, only differing

[2] we use $\mathcal{N}(\mu, \sigma^2)$ to represent the normal distribution of mean μ and standard deviation σ and $\mathcal{U}[a, b)$ to represent the uniform distribution sampling from the half open interval $[a, b)$.

in output: for the value network, the output vertex embeddings are aggregated into a single value by taking their mean, while the policy network outputs a single logit for each vertex in the graph.

C.3 Reinforcement Learning Environment Description

The reinforcement learning environment begins an episode with a singleton list containing a graph to be fully decomposed. At each timestep, a graph is popped from the front of the list and the model selects vertices to decompose. A graph decomposition is applied to these vertices to produce a number of new graphs. These graphs are simplified as much as possible using classically-efficient methods. If any graph can be fully reduced to zero vertices, in other words, classically simulated efficiently, it is discarded. Otherwise, the graph is added to the list of graphs to be decomposed. This procedure is continued until no graphs remain in the list, which amounts to a classical simulation of the original graph. At each time step, a reward equal to the effective α-efficiency of the decomposition is given.

C.4 PPO Hyperparameters

Table 1: PPO hyperparameters used to train our model

Hyperparameter	Value
Steps Trained	$967,680$
Batch Size	80
PPO Clip Range	0.223
GAE gamma	0.977
GAE lambda	0.976
Optimizer	Adam[20]
Learning rate	3×10^{-4}
Entropy coefficient	5.67×10^{-3}
Value coefficient	0.602
Min. steps per rollout	$3,840$
PPO updates per rollout	15
GAE normalisation	True
Gradient norm clipping	1
Max KL divergence per rollout	0.1

D Detailed Evaluation Results

Table 2: Comparison between our model and [24]

	Mean $\alpha \pm$ std dev	
	Magic5	Magic5 with Cats
GNN + PPO	$\mathbf{0.263 \pm 0.04}$	$\mathbf{0.232 \pm 0.04}$
Kissinger et al. [24]	0.293 ± 0.04	0.235 ± 0.04