Reinforcement Learning for Efficient Quantum Circuit Equivalence Checking via Tensor Network Contraction

Suhaib Al-Rousan[1], **Kim Guldstrand Larsen**[1], **Christian Schilling**[1], and **Max Tschaikowski**[2]
[1] *Aalborg University, Aalborg, Denmark* [2] *Sapienza University of Rome, Italy*
{suhaibar,kgl,christianms}@cs.aau.dk, tschaikowski@di.uniroma1.it

Abstract

Verifying whether two quantum circuits realize the same unitary is vital for optimization and compilation. However, approaches based on tensor network contraction often suffer from exponential blow-ups, not least since finding an optimal contraction order is NP-hard. We address this by employing tensor decision diagrams and formulating contraction as a sequential decision-making task. A reinforcement learning agent, trained via proximal policy optimization with a graph neural network policy, learns contraction orders to minimize both floating-point operations and peak memory. Our evaluation shows that the learned policy consistently outperforms heuristic baselines in running time while maintaining competitive memory usage, highlighting the potential of AI for scalable quantum circuit verification.

Keywords: Quantum circuit, Tensor network, Decision diagram, Reinforcement learning, Graph neural network

1 Introduction

Quantum computing has the potential to solve problems that are intractable for classical computers by leveraging the principles of superposition and entanglement. As quantum systems grow exponentially in the number of qubits, ensuring the correctness of quantum circuits becomes increasingly challenging. Equivalence checking, the task of verifying whether two circuits implement the same unitary, is essential for circuit optimization and compilation. Formally, it requires verifying whether the unitary operators U and U' satisfy $U = e^{i\theta}U'$, where θ is a global phase. Equivalently, one can check

$$UU'^{\dagger} = e^{i\theta} I_n, \tag{1}$$

where $UU'^{\dagger}$ is the product of one circuit's unitary and the conjugate transpose of the other, and I_n is the n-qubit identity operator.

A recent survey classifies equivalence-checking approaches into *formal* and *simulative* methods, stressing the need for scalable hybrids [Wille and Burgholzer, 2024]. Decision diagram frameworks address this by exploiting structural regularities in large unitaries, yet standard heuristics still suffer from exponential

blow-ups. Another promising approach is the contraction of tensors in a network corresponding to the circuit; however, finding the optimal order in which to contract the tensors is NP-hard [Lam et al., 1997]. Decision diagrams and tensors have recently been married in the form of tensor decision diagrams (TDDs) [Hong et al., 2022], and corresponding contraction heuristics have been proposed [Larsen et al., 2024], showing advantages in handling mixed-dimensional tensors but still admitting worst-case growth. At the same time, recent work on reinforcement learning (RL) suggests that learned policies can outperform hand-crafted contraction heuristics [Meirom et al., 2022].

Building on these insights, we propose an RL approach to *efficient quantum-circuit equivalence checking* using TDD contraction. Two circuits are concatenated (Eq. (1)) and then translated into a TDD network (TDDN). If the circuits are equivalent, contracting the network to a single node yields the identity TDD. We train a graph neural network (GNN) agent to learn a contraction policy that aims to minimize both floating-point operations (FLOPs) and peak TDD size.

2 Methodology

We cast the contraction of TDDNs as a *sequential decision-making problem*, with the goal to discover contraction orders that jointly minimize FLOPs and peak memory. To this end, we formulate the task as a Markov decision process and train an RL agent to guide the contraction. At step t, the environment is represented by a weighted graph $G_t = (V_t, E_t, w_t)$, where vertices correspond to tensors and edges encode shared indices with weights indicating computational cost. The agent performs an action a_t by selecting an edge $(u, v) \in E_t$ to contract, after which the environment merges u and v into a single node (resp. contracts these tensors), yielding the updated graph G_{t+1}. The instantaneous cost $c_t = C(s_t, a_t)$, where s_t encodes the current graph state, reflects the contraction cost, combining estimated FLOPs and memory usage. If edge e is contracted at step t, its cost is $w_t(e)$, and the total cost of a contraction path $P = (e_1, \ldots, e_k)$ is $C(P) = \sum_{t=1}^{k} w_t(e_t)$. We further extend this model by accounting for the peak memory footprint across all contractions. The contraction-planning problem is then to identify a contraction order that minimizes the cumulative cost, $P^*(G_0) = \arg\min_P C(P)$.

We adopt Proximal Policy Optimization (PPO) [Schulman et al., 2017] as the RL backbone and employ a GNN as the policy network (Figure 1), since TDDNs are naturally represented as graphs. Our choice of PPO was motivated by recent work by [Meirom et al., 2022]. The GNN leverages message passing [Battaglia et al., 2018] to capture both local and global structural information, enabling the agent to reason about contraction trade-offs.

To enable effective learning, we design a feature set that encodes information at the node, edge, and graph levels. Node features include the number of contractions a node has participated in, the number of 1-qubit and 2-qubit gates associated with it, a one-hot encoding of accumulated gate types (from a finite selection), and the size of the TDD representing the corresponding ten-

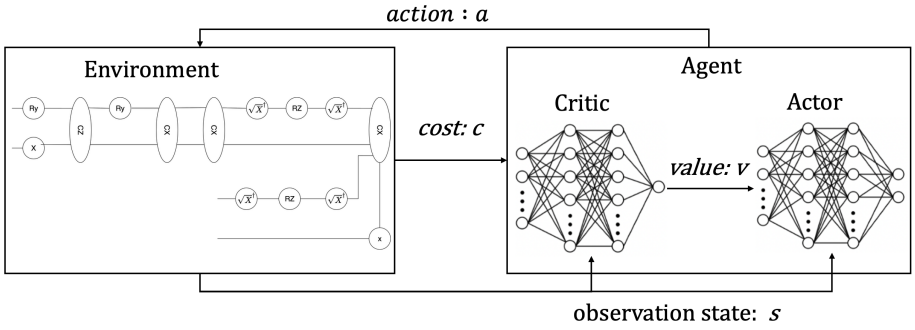

Figure 1: RL framework for TDDN contraction.

sor. Edge features estimate the contraction cost as described above. Global graph features capture aggregate statistics over the entire network, namely the averages of the node features and the average edge cost. By embedding these features through message passing, the policy network integrates local connectivity, tensor structure, and global cost context, learning contraction policies that consistently outperform hand-crafted heuristics.

Algorithm 1 outlines the training loop over a corpus of circuits spanning various sizes and gate patterns. We fix upper bounds on nodes and edges (M_n, M_e) for the policy, pad to these budgets, and mask inactive slots when smaller circuits are loaded; during contraction, the graph (and valid action set) shrinks and masking is updated accordingly.

3 Evaluation

We evaluate our methodology on *MQT Bench* [Quetschlich et al., 2023], focusing on algorithmic (level 1) and target-dependent (level 3) compilation levels, which differ in gate sets and layouts and thus stress both contraction quality and robustness. Our study spans five circuit families—Deutsch–Jozsa (DJ), GHZ, Bernstein–Vazirani (BV), W-state, and Graph-state—instantiated for 2–23 qubits (110 circuits in total). We compare against four established heuristics from Gray and Kourtis [2021] implemented in CoTenGra [Gray, 2025], which are also used as baselines in other works [Meirom et al., 2022, Larsen et al., 2024]: *Betweenness* (community detection to expose hierarchical structure), *Random Greedy* (sampling of multiple greedy paths and take the best), *Greedy* (minimize immediate cost), and *Hyper KaHyPar* (hypergraph partitioning for improved paths but higher planning time).

Across all benchmarks, our RL model consistently shows a lower contraction time while maintaining a competitive peak TDD size compared to the baseline heuristics (Figure 2). For example, on the BV circuits, the RL model achieves TDD sizes comparable to the hypergraph-based heuristics but is two orders of magnitude faster. In other families, *KaHyPar* occasionally produces

Algorithm 1 Training of a GNN policy for contraction planning with PPO

Input: Circuit set $\mathcal{S} = \{c_1, \ldots, c_k\}$, maximal number of nodes M_n, maximal number of edges M_e, number of training steps s

Output: Trained policy π_θ

procedure TrainPolicy($\mathcal{S}, M_n, M_e, s$)
  Initialize policy π_θ, value net, and environment parameters
  for $i = 1$ to s **do**
    $C \leftarrow$ Sample($\mathcal{S}$)
    $TN \leftarrow$ CircuitToTN(C); $TDDN \leftarrow$ BuildTDDN(TN)
    $G \leftarrow$ InitGraph($TDDN$); $\mathcal{E} \leftarrow$ ResetEnv(G)
    while |Edges($\mathcal{E}$)| > 0 **do**
      $o \leftarrow$ Observe($\mathcal{E}$) ▷ GNN features over G
      $a \leftarrow$ Sample($\pi_\theta(o)$) ▷ Select edge to contract
      $(\mathcal{E}, r) \leftarrow$ Step($\mathcal{E}, a$) ▷ Contract and compute reward
      StoreTransition(o, a, r) ▷ Append to PPO rollout buffer
    UpdatePolicyPPO(π_θ) ▷ Compute advantages; update policy/value with PPO; clear buffer
  return π_θ

Benchmark	Depth (min–med–max)	Gates (min–med–max)	% single-qubit (mean ± sd)
DJ	12–32–54	17–134–249	83.31 ± 1.41
GHZ	6–27–48	6–27–48	20.69 ± 14.69
BV	9–18–29	18–68–128	84.81 ± 2.08
W-state	12–75–138	14–140–266	67.54 ± 1.01
Graph-state	13–19–33	18–78–138	66.67 ± 0.00

Table 1: Benchmark corpus characteristics for varying qubit counts. We report per-family minimum/median/maximum for circuit depth and gate count as well as the amount of single-qubit gates (mean and standard deviation).

slightly smaller TDDs, but only with significantly longer planning times. The key advantage of our approach is that the RL agent learns a policy tailored to contraction dynamics rather than relying on static heuristics. By combining local structural cues with global context through message passing, the GNN can anticipate long-term cost growth and avoid locally optimal but globally expensive contractions. This allows our method to balance time and memory more effectively than traditional heuristics, resulting in robust performance across different circuit families.

Speedups diminish with growing qubit counts as graphs densify and FLOP growth dominates, yet the median speedup stays positive. The peak TDD size varies more because the agent sometimes accepts brief growth for later savings.

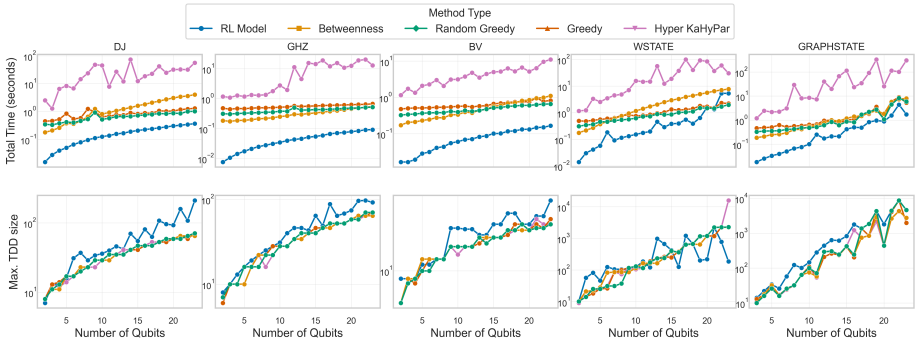

Figure 2: Performance comparison of the RL-based model and baselines across all benchmark families (log scales). The RL model consistently achieves lower contraction time, while the peak TDD size is less stable but overall similar.

4 Conclusions

We presented an RL framework for quantum-circuit equivalence checking via TDDN contraction. A GNN policy trained with PPO learns contraction orders that jointly minimize FLOPs and peak memory. The message-passing architecture, together with carefully designed node, edge, and global features, enables the agent to reason over both local structure and global cost context.

On five *MQT Bench* circuit families, the learned policy consistently reduces contraction time while keeping peak TDD size competitive with established heuristics. In particular, it outperforms greedy methods in runtime and approaches the quality of hypergraph-based planners without their planning overhead.

This work highlights the promise of combining decision diagrams with RL for contraction-order optimization. Looking ahead, we aim to scale to larger circuits and integrate hardware-aware features. This is a first step: we are broadening benchmarks (layouts, entanglement patterns, gate sets) to expose richer contraction regimes. Training across wide size ranges remains open—our current setup fixes node/edge budgets (M_n, M_e) and masks inactive slots as graphs shrink. While this stabilizes the action space, it may under-use small cases and constrain large ones; adaptive budgets and curriculum-style scaling are natural next steps.

Acknowledgments

This work was supported, in part, by the Danish e-infrastructure Consortium (DeiC) under reference number 4316-00006B, the Danish National Research Foundation (DNRF) through the Center CLASSIQUE under grant number 187, and the Villum Investigator Grant S4OS under reference number 37819.

References

P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

J. Gray. Cotengra documentation. `https://cotengra.readthedocs.io/en/latest/`, 2025. Accessed: 2025-10-13.

J. Gray and S. Kourtis. Hyper-optimized tensor network contraction. *Quantum*, 5:410, 2021. doi: 10.22331/Q-2021-03-15-410.

X. Hong, X. Zhou, S. Li, Y. Feng, and M. Ying. A tensor network based decision diagram for representation of quantum circuits. *ACM Trans. Design Autom. Electr. Syst.*, 27(6):60:1–60:30, 2022. doi: 10.1145/3514355.

C. Lam, P. Sadayappan, and R. Wenger. On optimizing a class of multi-dimensional loops with reductions for parallel execution. *Parallel Process. Lett.*, 7(2):157–168, 1997. doi: 10.1142/S0129626497000176. URL `https://doi.org/10.1142/S0129626497000176`.

C. B. Larsen, S. B. Olsen, K. G. Larsen, and C. Schilling. Contraction heuristics for tensor decision diagrams. *Entropy*, 26(12), 2024. doi: 10.3390/e26121058.

E. A. Meirom, H. Maron, S. Mannor, and G. Chechik. Optimizing tensor network contraction using reinforcement learning. In *ICML*, volume 162, pages 15278–15292. PMLR, 2022. URL `https://proceedings.mlr.press/v162/meirom22a.html`.

N. Quetschlich, L. Burgholzer, and R. Wille. MQT Bench: Benchmarking software and design automation tools for quantum computing. *Quantum*, 7:1062, 2023. doi: 10.22331/Q-2023-07-20-1062.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

R. Wille and L. Burgholzer. Verification of quantum circuits. *Handbook of Computer Architecture*, pages 1413–1440, 2024.