RL-Guided Contraction of Symbolic Tensor Networks for Quantum Circuit Equivalence

AALBORG UNIVERSITY DENMARK

Suhaib Al-Rousan, Kim Guldstrand Larsen, Christian Schilling, and Max Tschaikowski

Department of Computer Science, Aalborg University, Denmark

{suhaibar,kgl,christianms,tschaikowski}@cs.aau.dk

DeiC DANISH eINFRASTRUCTURE CONSORTIUM

VILLUM FONDEN

Overview

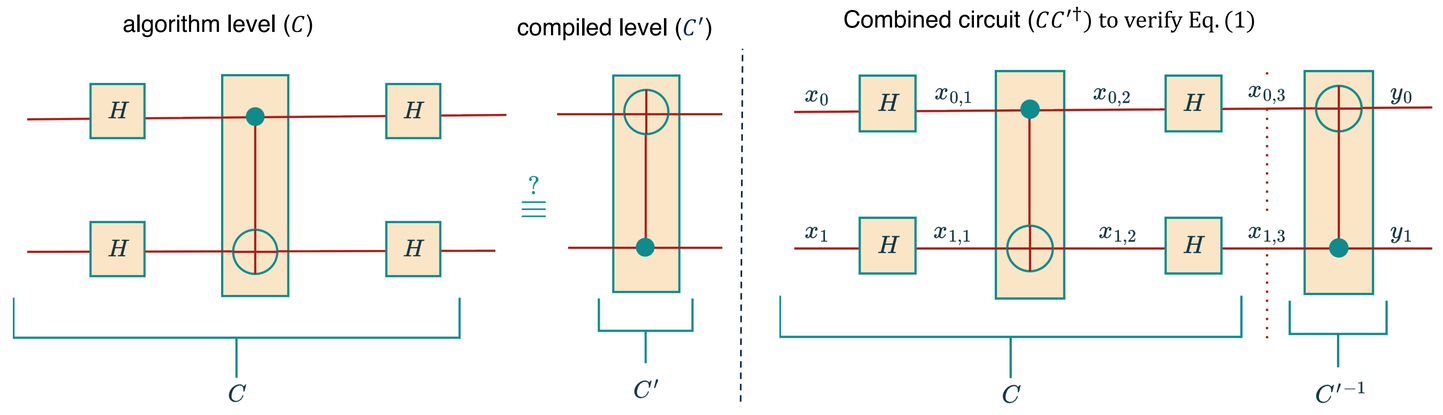
1

- **Goal:** Checking equivalence of two quantum circuits, $C \equiv C'$ iff $U = e^{i\theta}U'$ (1)
  - → $UU'^{\dagger} = U'U^{\dagger} = e^{i\theta}I_n$ where U and U' are the unitaries of the quantum circuits C and C' respectively, θ is a global phase, and I_n is the *n*-qubit identity.
  - → $CC'^{\dagger}$ is converted to a tensor network. Thus, we apply contraction until we have one tensor that represents the unitary. It should be $e^{i\theta}I_n$ if $C \equiv C'$.
- **Challenges:** 1. The algorithm scales exponentially in the number of qubits.
  2. Finding the optimal contraction order is an NP-hard problem.
- **Our approach:** We combine a symbolic data structure called **decision diagrams** (→ challenge 1) to mitigate the explosion and train a neural agent to choose the best contraction steps (→ challenge 2). Our work is a synergy of recent results[1,2].

1. Meirom, Eli, et al. "Optimizing tensor network contraction using reinforcement learning." International Conference on Machine Learning. 2022.
2. Larsen, Christian Bogh, et al. "Contraction Heuristics for Tensor Decision Diagrams." Entropy 26.12 (2024)".

Example: Two Quantum Circuits and Combined Circuit

2



Tensor Network (TN)

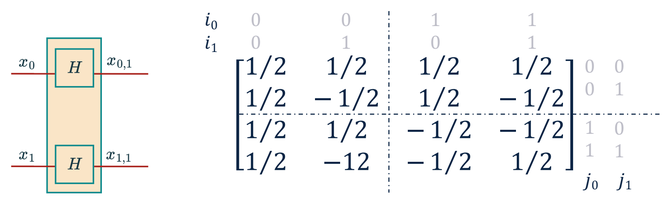
3

- **From quantum gates to tensors:** A k-qubit gate corresponds to a node with a tensor corresponding to the unitary with rank $2k$ indices/edges.
- **Tensor contraction:** Contracting two nodes in a tensor network connected by an edge yields a new node, rewiring the edges and combining the tensors.
- **Tensor network contraction:** Iteratively contracting two nodes always results in the same final tensor, but the intermediate tensors may differ in complexity. The goal is to find a contraction order that keeps intermediate tensors small.

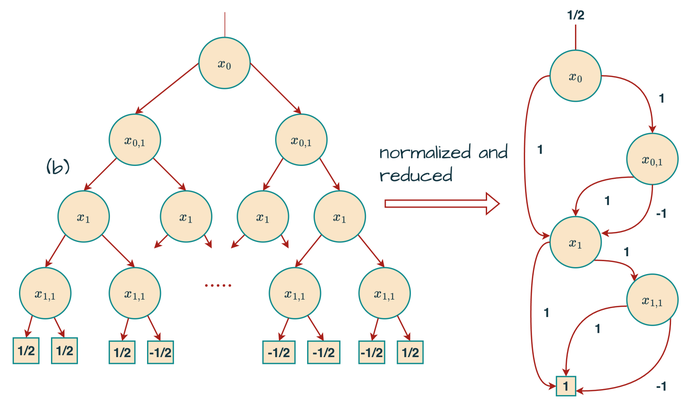
Tensor Decision Diagram (TDD) and TDD-Network

4

The unitary representation of two parallel *H* gates



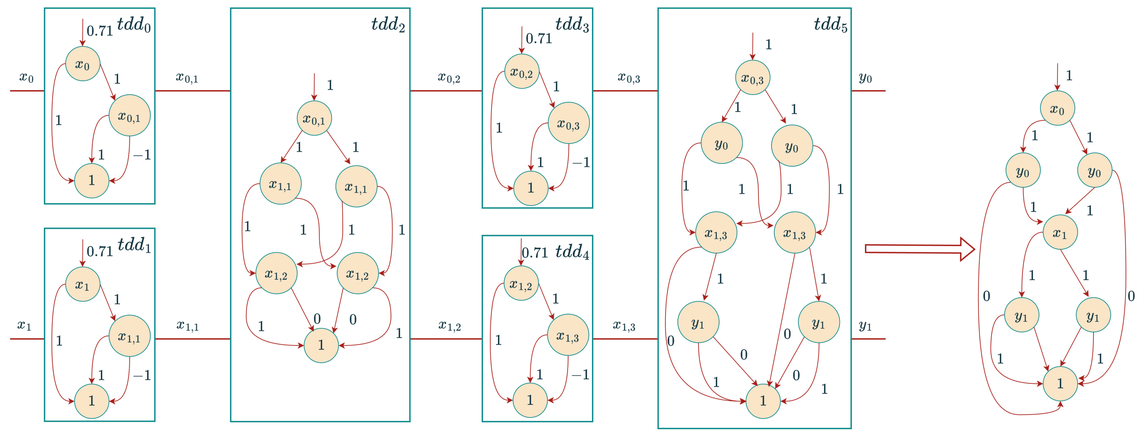
TDD representation before and after normalization



- **Tensor decision diagram:** Before normalization, a TDD is a tensor as a binary tree over the indices. Normalization removes redundancies, yielding a compact directed graph.
- **TDD-network:** A tensor network where all tensors are represented as a TDD.

Example: Combined Circuit as TDD-Network

5



The TDD-network derived from the combined circuit $CC'^{\dagger}$, where each node contains a TDD representing the gate unitary.

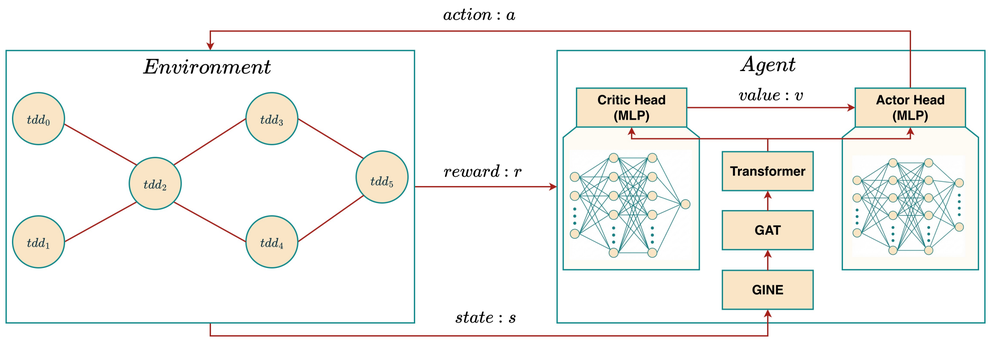
Reinforcement Learning (RL)

6

- **MDP formulation:** We cast contraction planning as an episodic MDP (S, A, P, r) over the partially contracted TDDN:
  - **State** s_t: graph $G_t = (V_t, E_t)$; nodes are TDDs, edges denote shared indices
  - **Action** a_t: pick an edge $(u, v) \in E_t$ to contract two neighboring TDDs
  - **Transition** P: replace (u, v) by a merged node and update shared-index edges; episode terminates when $|V_t| = 1$
  - **Reward** r_t: penalizes memory spikes from the TDD produced by a_t

$$r_t = -\lambda \cdot (\log_{10} |TDD_{t+1}|)^{\beta} \quad (2)$$

- Here $|TDD_{t+1}|$ is the node count of the resulting TDD, $\lambda = 10 / (|V_0| - 1)$ normalizes by episode length, and β controls the severity of large intermediate diagrams
- **Optimization objective:** find a contraction path that maximizes cumulative reward — directly targeting the peak-node memory bottleneck rather than static arithmetic cost



RL framework

Graph Neural Network (GNN)

7

- **From TDD-network to GNN:** Convert dynamic graph G_t into a padded observation $Ot = (Fn, Fe, Fg, It, |V|, |E|, \mu t)$. Fixed capacities (N_{max}, M_{max}) give a consistent input dimension while allowing variable-size graphs via padding; μ_t masks invalid contractions.

Node features (4-dim) — encode TDD complexity:

- Merge count: original TDDs contracted into node
- Memory load: TDD size ($\log_{10}$ of node count)
- Connectivity density: bond dim. (shared indices), normalized by N_{max}
- Node degree: adjacent nodes, normalized

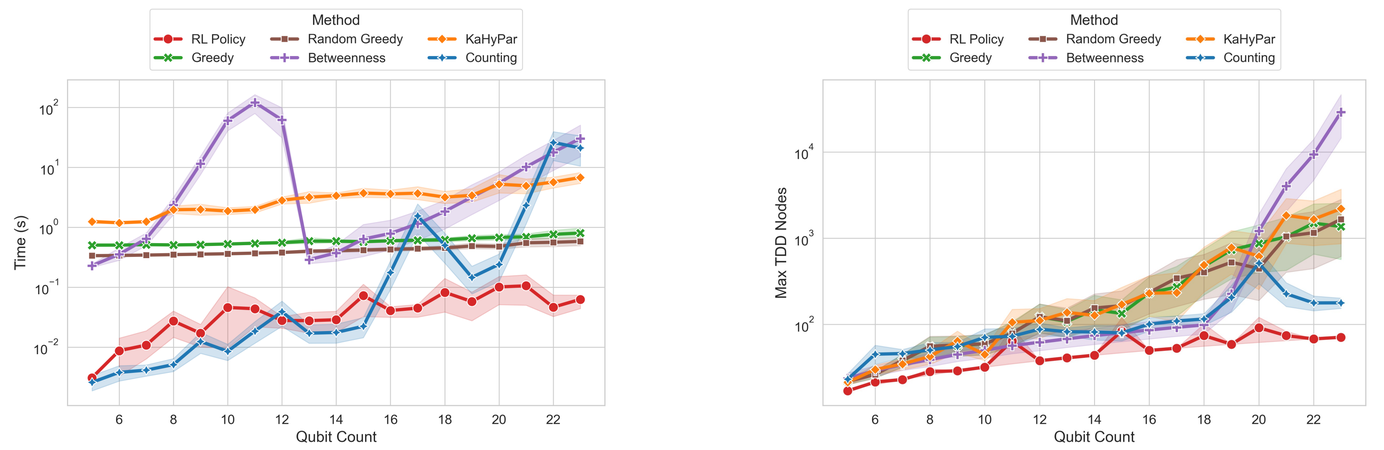
Edge features (2-dim) — encode connectivity:

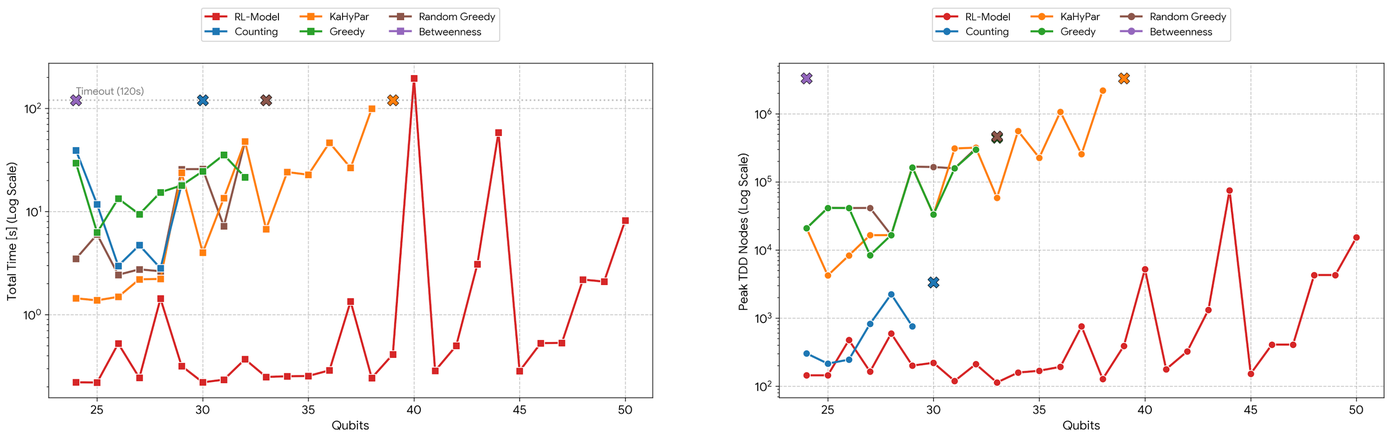
- Bond strength: shared indices, normalized
- Local topology: avg. endpoint bond dim., normalized by N_{max}

Global features (5-dim): normalized active node and edge counts, average and maximum TDD sizes (log-scaled), and a normalized episode step counter.

Results

8





CTG–KaHyPar is part of the KaHyPar hyper-graph partitioner[1].
CTG–Greedy originates from the Contraction-Tree-Generator framework[1].

1. Gray, Johnnie, and Stefanos Kourtis. "Hyper-optimized tensor network contraction." Quantum 5 (2021): 410.
2. Larsen CB, Olsen SB, Larsen KG, Schilling C. Contraction Heuristics for Tensor Decision Diagrams. Entropy (Basel). 2024 Dec 5;26(12):1058.