Equivalence Checking of Quantum Circuits

Christian Schilling

Overview

Overview

Circuit compilation

- When designing (classical or quantum) circuits, it is useful to have many gate types available
- Real hardware only supports a few gates types
- A **compiler** translates a high-level circuit with many gate types to a low-level circuit with few gate types
- Quantum gates incur different amounts of error
- Deeper quantum circuits incur more errors
- Quantum compilers also optimize for reducing errors
- Important that the compiled circuits are **equivalent** (same input $\rightsquigarrow$ same output)
- How can we check equivalence of circuits?

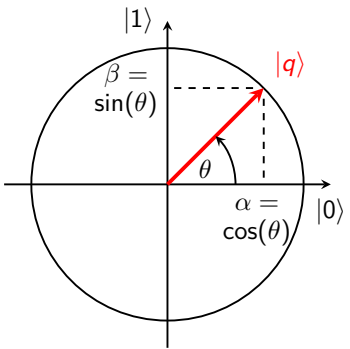
Overview

Qubit

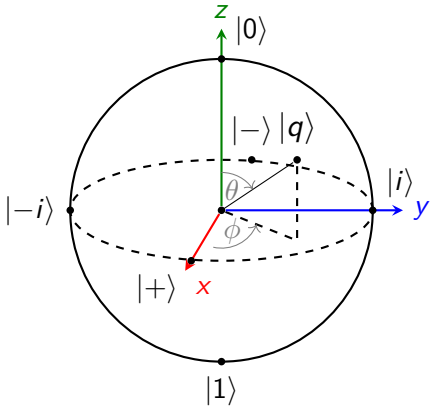
- Ground state $|0\rangle$, excited state $|1\rangle$
- Superposition $|q\rangle = \alpha\,|0\rangle + \beta\,|1\rangle, \quad \alpha, \beta \in \mathbb{C}$
- Written as 2D vector: $|q\rangle \equiv \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$
- Constraint $|\alpha|^2 + |\beta|^2 = 1$
- Ignoring complex numbers,
  a qubit state describes a circle

Qubit

- Ground state $|0\rangle$, excited state $|1\rangle$
- Superposition $|q\rangle = \alpha\,|0\rangle + \beta\,|1\rangle, \quad \alpha, \beta \in \mathbb{C}$
- Written as 2D vector: $|q\rangle \equiv \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$
- Constraint $|\alpha|^2 + |\beta|^2 = 1$
- Ignoring complex numbers, a qubit state describes a circle



Bloch sphere



- Polar coordinates: $|q\rangle = \underbrace{\cos\frac{\theta}{2}}_{\alpha}|0\rangle + \underbrace{e^{i\phi}\sin\frac{\theta}{2}}_{\beta}|1\rangle$

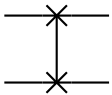
Multiple qubits

$$
\begin{aligned}
|q_0, q_1\rangle = |q_0\rangle \otimes |q_1\rangle &\equiv \begin{bmatrix} \alpha_0 \\ \beta_0 \end{bmatrix} \otimes \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix} \\
&= \alpha_0\alpha_1 |00\rangle + \alpha_0\beta_1 |01\rangle + \beta_0\alpha_1 |10\rangle + \beta_0\beta_1 |11\rangle \\
&\equiv \begin{bmatrix} \alpha_0\alpha_1 \\ \alpha_0\beta_1 \\ \beta_0\alpha_1 \\ \beta_0\beta_1 \end{bmatrix}
\end{aligned}
$$

- Not every multi-qubit state is a product state

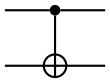
Quantum gates

- Unitary matrix operations
- Example: Swapping of two qubits



$$SWAP(|q_0, q_1\rangle) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_0\alpha_1 \\ \alpha_0\beta_1 \\ \beta_0\alpha_1 \\ \beta_0\beta_1 \end{bmatrix} = \begin{bmatrix} \alpha_0\alpha_1 \\ \beta_0\alpha_1 \\ \alpha_0\beta_1 \\ \beta_0\beta_1 \end{bmatrix} = |q_1, q_0\rangle$$

Another quantum gate: Controlled NOT



$$CNOT(|q_0, q_1\rangle) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha_0\alpha_1 \\ \alpha_0\beta_1 \\ \beta_0\alpha_1 \\ \beta_0\beta_1 \end{bmatrix} = \begin{bmatrix} \alpha_0\alpha_1 \\ \alpha_0\beta_1 \\ \beta_0\beta_1 \\ \beta_0\alpha_1 \end{bmatrix}$$

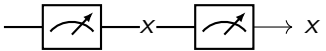
$$= |q_0, q_0 \oplus q_1\rangle$$

where $\oplus$ is addition modulo 2 ("XOR")

Measurement

- Probabilistically projects to basis state ($|0\rangle$ or $|1\rangle$)
- Not invertible

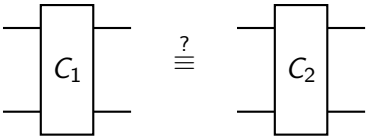
$$|q\rangle \equiv \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$



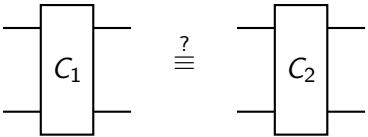
$$x = \begin{cases} |0\rangle & \text{with probability } |\alpha|^2 \\ |1\rangle & \text{with probability } |\beta|^2 \end{cases}$$

Overview

Black-box equivalence check for classical circuits

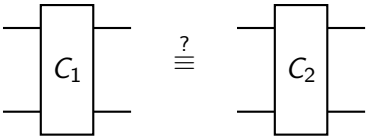


Black-box equivalence check for classical circuits



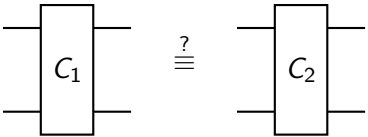
- Classical circuits: Pick an input and execute both circuits

Black-box equivalence check for classical circuits



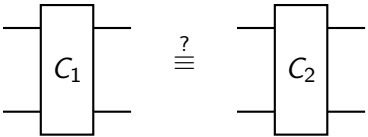
- Classical circuits: Pick an input and execute both circuits
  - One-way result: Disagreement implies different circuits
  - **Exponentially many** classical inputs

Black-box equivalence check for quantum circuits



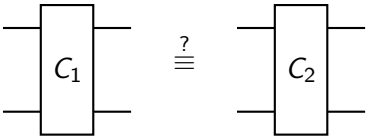
- Classical circuits: Pick an input and execute both circuits
  - One-way result: Disagreement implies different circuits
  - **Exponentially many** classical inputs

Black-box equivalence check for quantum circuits



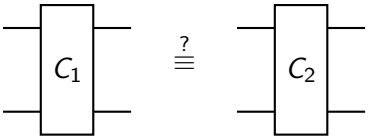
- Classical circuits: Pick an input and execute both circuits
  - One-way result: Disagreement implies different circuits
  - **Exponentially many** classical inputs
- Infinitely many quantum states as input?

Black-box equivalence check for quantum circuits



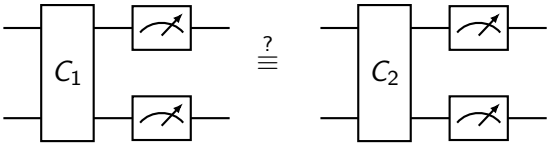
- Classical circuits: Pick an input and execute both circuits
  - One-way result: Disagreement implies different circuits
  - **Exponentially many** classical inputs
- Infinitely many quantum states as input?
  - **Sufficient to check basis states**
  - **Exponentially many** basis states

Black-box equivalence check for quantum circuits



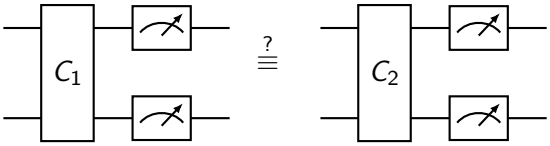
- Classical circuits: Pick an input and execute both circuits
  - One-way result: Disagreement implies different circuits
  - **Exponentially many** classical inputs
- Infinitely many quantum states as input?
  - **Sufficient to check basis states**
  - **Exponentially many** basis states
- **Problem: Can only observe basis states (measurements)**

Black-box equivalence check for quantum circuits



- Classical circuits: Pick an input and execute both circuits
  - One-way result: Disagreement implies different circuits
  - **Exponentially many** classical inputs
- Infinitely many quantum states as input?
  - **Sufficient to check basis states**
  - **Exponentially many** basis states
- **Problem: Can only observe basis states (measurements)**
  - Disagreement does **not** imply different circuits

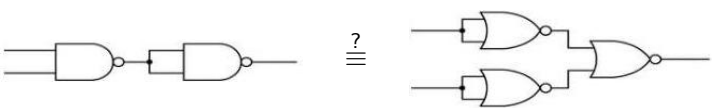
Black-box equivalence check for quantum circuits



- Classical circuits: Pick an input and execute both circuits
  - One-way result: Disagreement implies different circuits
  - **Exponentially many** classical inputs
- Infinitely many quantum states as input?
  - **Sufficient to check basis states**
  - **Exponentially many** basis states
- **Problem: Can only observe basis states (measurements)**
  - Disagreement does **not** imply different circuits
  - **Approximation** by many executions is **expensive**

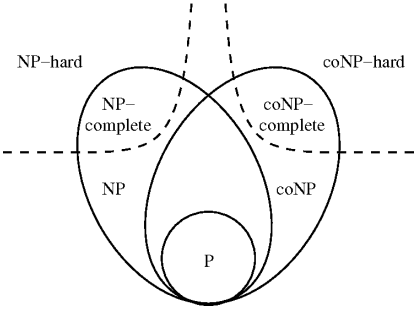
White-box equivalence check for classical circuits

- What if we know the circuits?

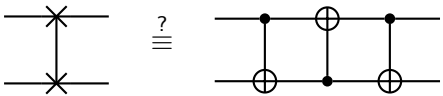


White-box equivalence check for classical circuits

- Fundamentally, the problem is likely not easier
- Checking that two classical circuits are equivalent is co-NP-complete
  - Believed to require exponential complexity
  - So no better way than checking all possible inputs

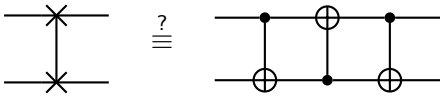


White-box equivalence check for quantum circuits



- What if we know the quantum circuits?

White-box equivalence check for quantum circuits



- What if we know the quantum circuits?
- We can check equivalence by comparing the matrices

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Simple algorithm to check equivalence of quantum circuits

- Given: Two circuits C_1, C_2
- Question: Are the circuits equivalent ($C_1 \equiv C_2$)?

Simple algorithm to check equivalence of quantum circuits

- Given: Two circuits C_1, C_2
- Question: Are the circuits equivalent ($C_1 \equiv C_2$)?
- Simple algorithm
  1. Compute unitary matrices U_1, U_2
  2. Check equality up to a factor (global phase $e^{i\theta}$)

$$U_1 \stackrel{?}{=} e^{i\theta} \cdot U_2$$

Simple algorithm to check equivalence of quantum circuits

- Given: Two circuits C_1, C_2
- Question: Are the circuits equivalent ($C_1 \equiv C_2$)?
- Simple algorithm
  1. Compute unitary matrices U_1, U_2
  2. Check equality up to a factor (global phase $e^{i\theta}$)

$$U_1 \stackrel{?}{=} e^{i\theta} \cdot U_2$$

- Problem: Matrices are **exponentially large**

  For n qubits $\rightsquigarrow 2^n \times 2^n$

Equivalence checking is hard

- Checking exact equivalence is co-NQP-complete[1]
- Checking approximate equivalence is co-QMA-complete[2,3]
- Problems in these complexity classes are again widely believed to require exponential computations in the worst case

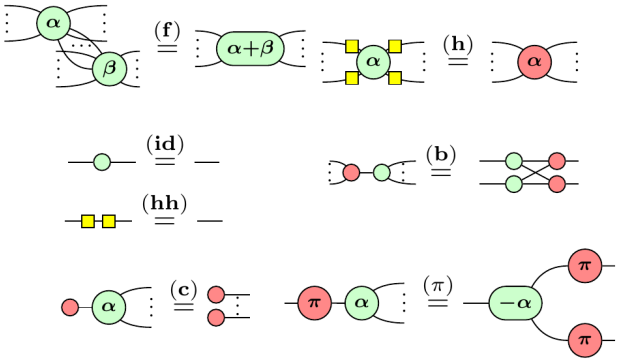
[1] Y. Tanaka. *Int. J. Quantum Inf.* (2010).
[2] D. Janzing, P. Wocjan, and T. Beth. *Int. J. Quantum Inf.* (2005).
[3] Z. Ji and X. Wu. 2009. arXiv: 0906.5416.

Smarter approaches to equivalence checking

- ZX-calculus[1]
- Encoding with decision diagrams[2] ← relevant later
- Tensor network contraction[3] ← relevant later
- Simulation-based approach for the Clifford group[4]
- Weighted model counting[5]

[1] T. Peham, L. Burgholzer, and R. Wille. *IEEE J. Emerg. Sel. Topics Circuits Syst.* (2022).
[2] L. Burgholzer and R. Wille. *ASP-DAC.* 2020.
[3] R. Orús. *Nature Reviews Physics* (2019).
[4] D. Thanos, T. Coopmans, and A. Laarman. *ATVA.* 2023.
[5] J. Mei, T. Coopmans, M. M. Bonsangue, and A. Laarman. *IJCAR.* 2024.

Based on ZX-calculus[1,2]

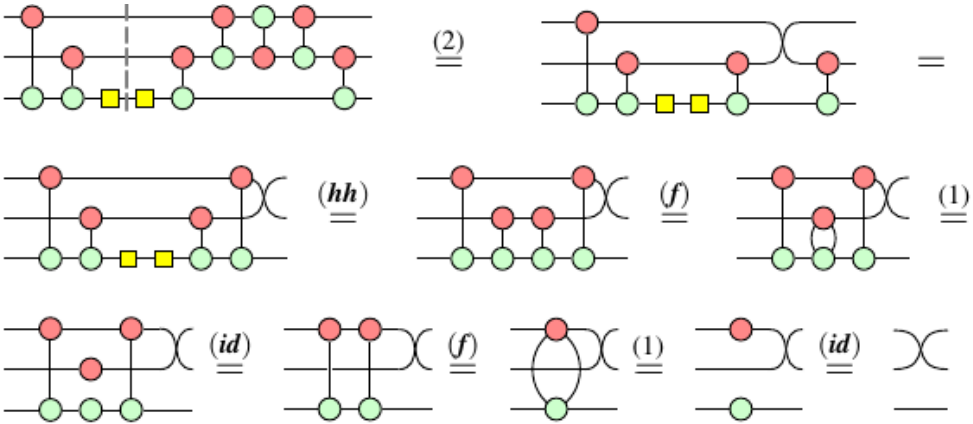


Axioms

[1] B. Coecke and R. Duncan. *ICALP*. 2008.

[2] T. Peham, L. Burgholzer, and R. Wille. *IEEE J. Emerg. Sel. Topics Circuits Syst.* (2022).

Based on ZX-calculus[1,2]

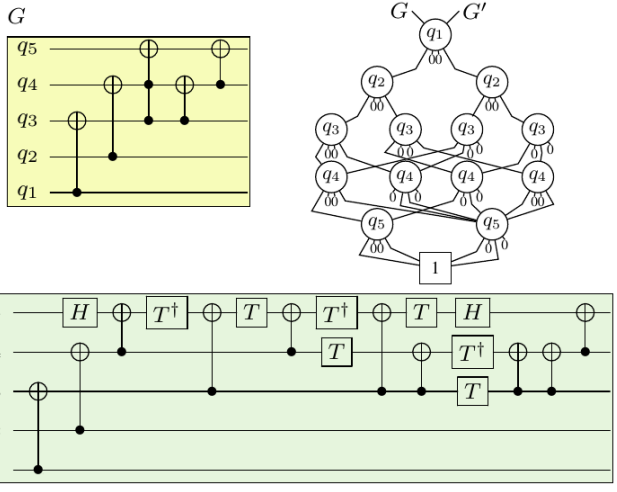


Example proof

[1] B. Coecke and R. Duncan. *ICALP*. 2008.

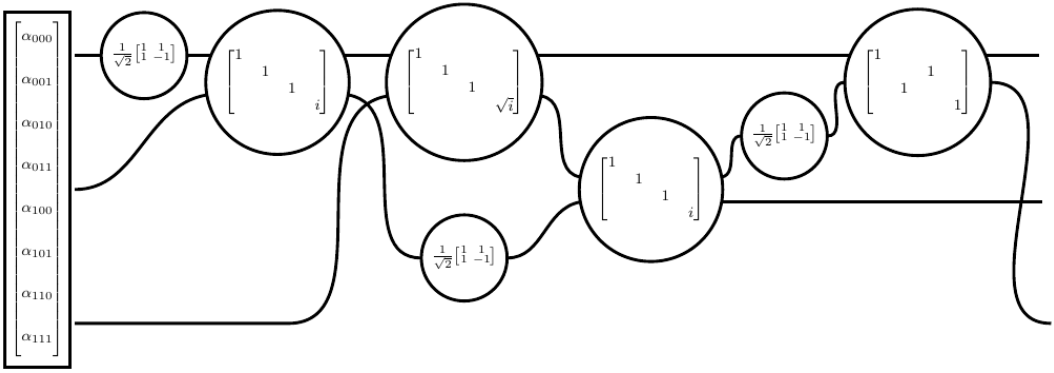
[2] T. Peham, L. Burgholzer, and R. Wille. *IEEE J. Emerg. Sel. Topics Circuits Syst.* (2022).

Based on decision diagrams[1]



[1] L. Burgholzer and R. Wille. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* (2021).

Based on tensor network contraction[1,2]



[1] X. Hong, Y. Feng, S. Li, and M. Ying. *ICCAD*. 2022.

[2] L. Burgholzer, A. Ploier, and R. Wille. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* (2023).

Based on a folklore theorem

Theorem 1. *Let U, V be two unitaries on $n \geq 1$ qubits. Then U is equivalent to V if and only if the following conditions hold:*

1. *for all $j \in \{1, 2, \ldots, n\}$, we have $UZ_jU^\dagger = VZ_jV^\dagger$; and*
2. *for all $j \in \{1, 2, \ldots, n\}$, we have $UX_jU^\dagger = VX_jV^\dagger$.*

Here, as before, we have denoted $Z_j = I \otimes \cdots \otimes I \otimes Z \otimes I \otimes \cdots \otimes I$, i.e. an n-fold tensor product of identity gates I with the Pauli Z gate at the j-th position. Analogously, $X_j = I \otimes \cdots \otimes I \otimes X \otimes I \otimes \cdots \otimes I$ where X is the Pauli X gate.

- Reduces to **simulation** for **Clifford circuits**[1]
  - $UZ_jU^\dagger$ is the same as computing $U|0\rangle^{\otimes n}$
  - $UX_jU^\dagger$ is the same as computing $U|+\rangle^{\otimes n}$
- Simulation is generally hard, but it is easy for Clifford circuits
- Extension to general circuits based on reduction to **weighted model counting**[2]

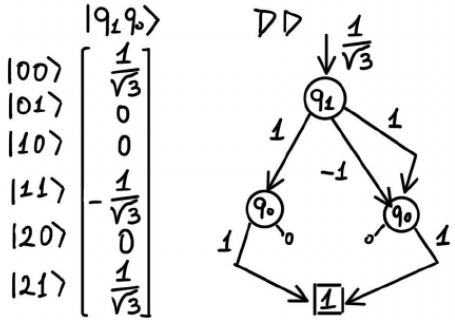
[1] D. Thanos, T. Coopmans, and A. Laarman. *ATVA*. 2023.

[2] J. Mei, T. Coopmans, M. M. Bonsangue, and A. Laarman. *IJCAR*. 2024.

Overview

Quantum decision diagrams (QDDs)[1]

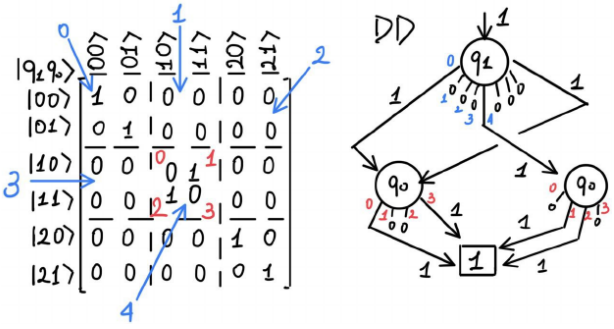
- Similar to BDDs, but represent complex numbers
- Available for both vectors and matrices
- Examples below: Mixed qudit systems (≥ 2 bases)



[1] P. Niemann, R. Wille, D. M. Miller, M. A. Thornton, and R. Drechsler. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* (2016).

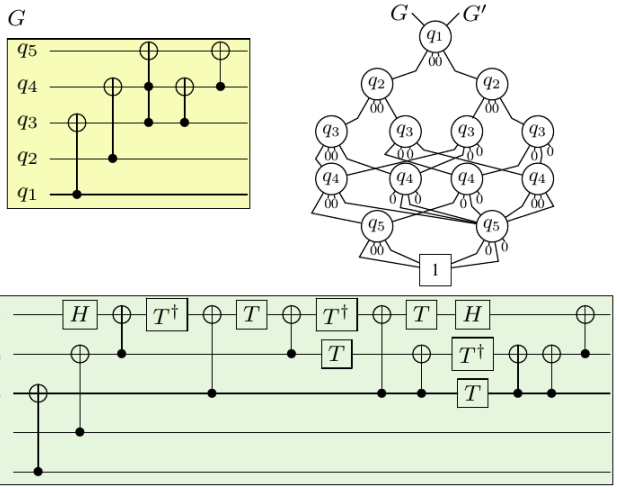
Quantum decision diagrams (QDDs)[1]

- Similar to BDDs, but represent complex numbers
- Available for both vectors and matrices
- Examples below: Mixed qudit systems (≥ 2 bases)



[1] P. Niemann, R. Wille, D. M. Miller, M. A. Thornton, and R. Drechsler. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* (2016).

Comparison of circuits after encoding[1]



[1] L. Burgholzer and R. Wille. *ASP-DAC*. 2020.

Alternative "reverse scheme" for equivalence[1]

$$C_1 \equiv C_2 \overset{\text{def}}{\iff} \exists \theta \colon U_1 = e^{i\theta} \cdot U_2$$

$$\iff \exists \theta \colon U_1 \cdot U_2^\dagger = e^{i\theta} \cdot I \overset{\text{def}}{\iff} C_1 C_2^{-1} \equiv C_I$$

- C_2^{-1} is the inverted C_2 (reversed and each gate inverted)
- Allows to combine both circuits

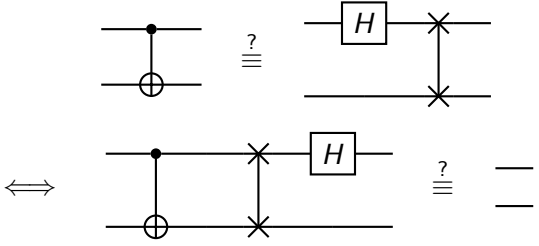
[1] G. F. Viamontes, I. L. Markov, and J. P. Hayes. *ICCAD*. 2007.

Alternative "reverse scheme" for equivalence[1]

$$C_1 \equiv C_2 \overset{\text{def}}{\iff} \exists\theta\colon U_1 = e^{i\theta} \cdot U_2$$

$$\iff \exists\theta\colon U_1 \cdot U_2^\dagger = e^{i\theta} \cdot I \overset{\text{def}}{\iff} C_1 C_2^{-1} \equiv C_I$$

- C_2^{-1} is the inverted C_2 (reversed and each gate inverted)
- Allows to combine both circuits



(Coincidentally, the swap and Hadamard gates are self-inverse)

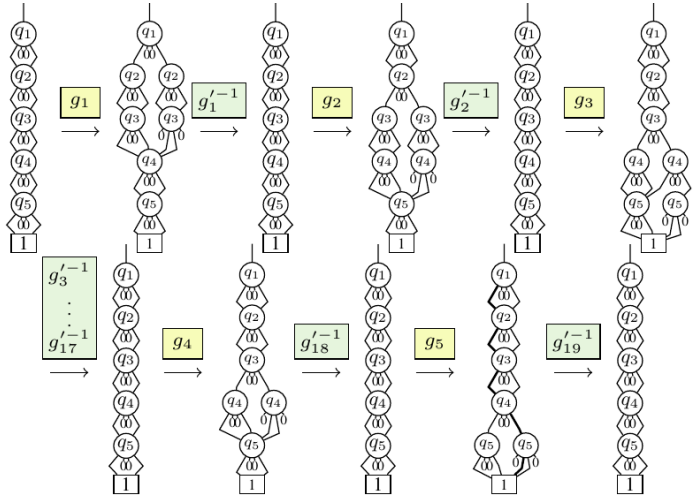
[1] G. F. Viamontes, I. L. Markov, and J. P. Hayes. *ICCAD*. 2007.

Reverse scheme "inside out" with QDDs[1]

- Motivation: Identity matrix has a small QDD encoding
- Start from identity matrix
- Multiply with some gates from first circuit
- Multiply with some gates from second circuit
- Repeat
- In the example (next slide), the QDD never gets above 9 nodes (compared to 13 nodes for each gate)

[1] L. Burgholzer and R. Wille. *ASP-DAC*. 2020.

Reverse scheme "inside out" with QDDs[1]

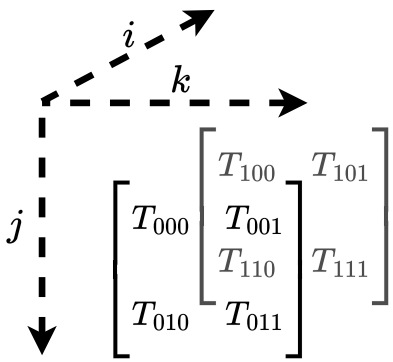


[1] L. Burgholzer and R. Wille. *ASP-DAC*. 2020.

Overview

Tensors

- Generalization of vectors / matrices to higher dimensions



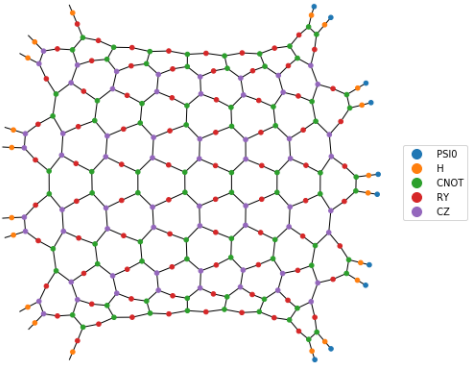
Tensors

- Generalization of vectors / matrices to higher dimensions
- High-level graphical representation as a node with edges

vector	v_j	
matrix	M_{ij}	
3-index tensor	T_{ijk}	

Tensor networks

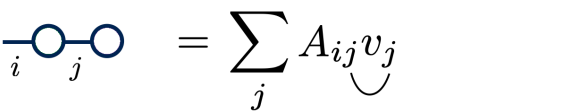
- Tensors can be arranged in a graph



Tensor networks

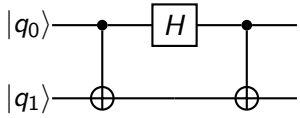
- Tensors can be arranged in a graph
- Nodes with shared edges can be **contracted** (= merged)

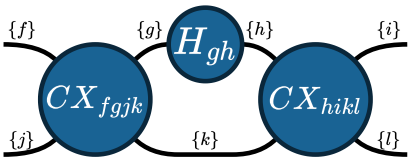
  Corresponds to matrix-vector and matrix-matrix multiplication for special cases



Example

1. Initial tensor network has one tensor for each gate
   Three choices for contraction $(\{g\}, \{h\}, \{k\})$

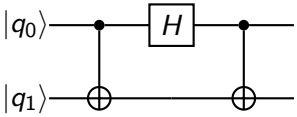


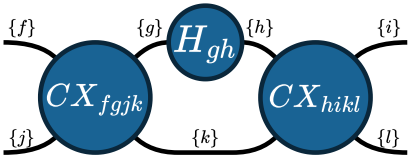


1.

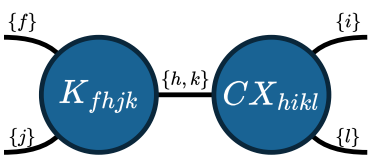
Example

1. Initial tensor network has one tensor for each gate
   Three choices for contraction $(\{g\}, \{h\}, \{k\})$
2. Contraction of CX_{fgjk} and H_{gh} via $\{g\}$





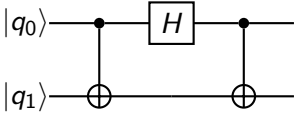
1.

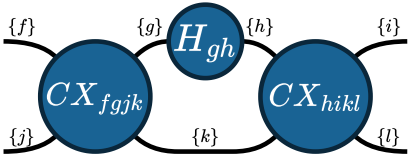


2.

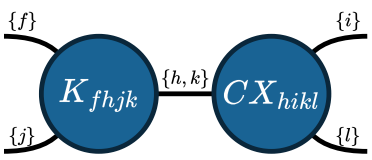
Example

1. Initial tensor network has one tensor for each gate
   Three choices for contraction $(\{g\}, \{h\}, \{k\})$
2. Contraction of CX_{fgjk} and H_{gh} via $\{g\}$
3. Contraction of remaining two tensors

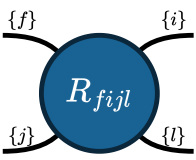




1.



2.



3.

Application: Quantum simulation on classical computer[2]

- Contract tensors in good order
- Different contraction heuristics to minimize floating-point operations, size, etc.[1]

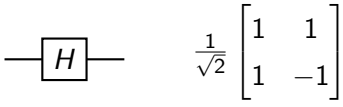
[1] J. Gray and S. Kourtis. *Quantum* (2021).
[2] I. L. Markov and Y. Shi. *SIAM J. Comput.* (2008).

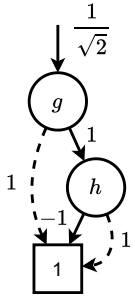
Overview

Tensor decision diagrams (TDDs)

- Alternative representation of tensors

Example: Hadamard gate with matrix, tensor, and TDD

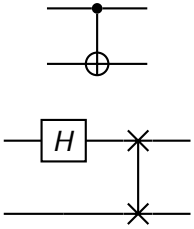




TDD-based algorithm for equivalence checking[1]

Algorithm combines reverse scheme, tensor networks, and TDDs

Given: Quantum circuits C_1, C_2



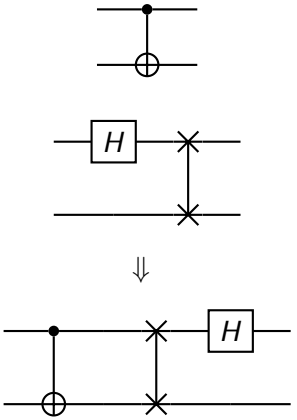
[1] C. B. Larsen, S. B. Olsen, K. G. Larsen, and C. Schilling. *Entropy* (2024).

TDD-based algorithm for equivalence checking[1]

Algorithm combines reverse scheme, tensor networks, and TDDs

Given: Quantum circuits C_1, C_2

1. Construct circuit $C_1 C_2^{-1}$



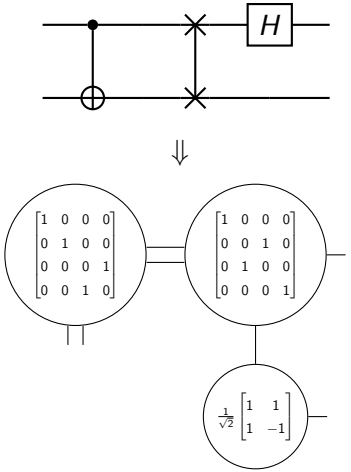
[1] C. B. Larsen, S. B. Olsen, K. G. Larsen, and C. Schilling. *Entropy* (2024).

TDD-based algorithm for equivalence checking[1]

Algorithm combines reverse scheme, tensor networks, and TDDs

Given: Quantum circuits C_1, C_2

1. Construct circuit $C_1 C_2^{-1}$
2. Convert $C_1 C_2^{-1}$ to tensor network



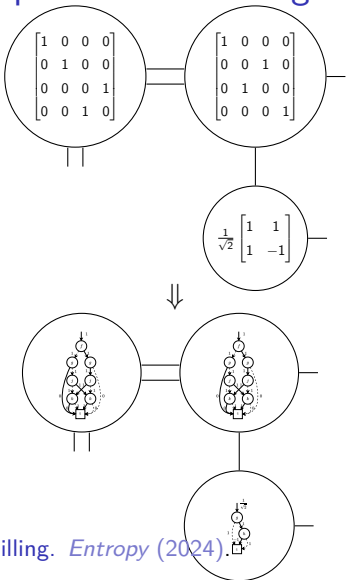
[1] C. B. Larsen, S. B. Olsen, K. G. Larsen, and C. Schilling. *Entropy* (2024).

TDD-based algorithm for equivalence checking[1]

Algorithm combines reverse scheme, tensor networks, and TDDs

Given: Quantum circuits C_1, C_2

1. Construct circuit $C_1 C_2^{-1}$
2. Convert $C_1 C_2^{-1}$ to tensor network
3. Convert all tensors to TDDs



(TDDs on the right are only exemplary)

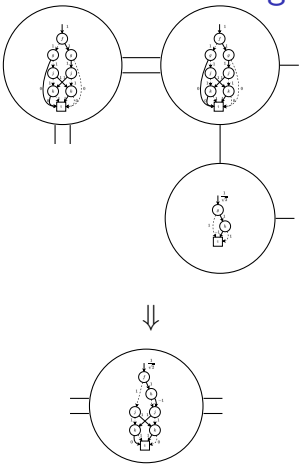
[1] C. B. Larsen, S. B. Olsen, K. G. Larsen, and C. Schilling. *Entropy* (2024).

TDD-based algorithm for equivalence checking[1]

Algorithm combines reverse scheme, tensor networks, and TDDs

Given: Quantum circuits C_1, C_2

1. Construct circuit $C_1 C_2^{-1}$
2. Convert $C_1 C_2^{-1}$ to tensor network
3. Convert all tensors to TDDs
4. Contract TDD network



(TDDs on the right are only exemplary)

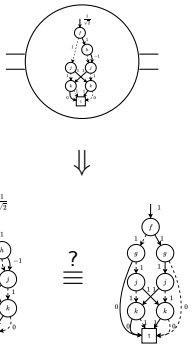
[1] C. B. Larsen, S. B. Olsen, K. G. Larsen, and C. Schilling. *Entropy* (2024).

TDD-based algorithm for equivalence checking[1]

Algorithm combines reverse scheme, tensor networks, and TDDs

Given: Quantum circuits C_1, C_2

1. Construct circuit $C_1 C_2^{-1}$
2. Convert $C_1 C_2^{-1}$ to tensor network
3. Convert all tensors to TDDs
4. Contract TDD network
5. Compare resulting TDD to identity TDD



(TDDs on the right are only exemplary)

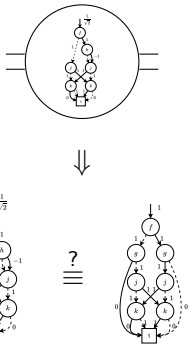
[1] C. B. Larsen, S. B. Olsen, K. G. Larsen, and C. Schilling. *Entropy* (2024).

TDD-based algorithm for equivalence checking[1]

Algorithm combines reverse scheme, tensor networks, and TDDs

Given: Quantum circuits C_1, C_2

1. Construct circuit $C_1 C_2^{-1}$
2. Convert $C_1 C_2^{-1}$ to tensor network
3. Convert all tensors to TDDs
4. **Contract TDD network** $\leftarrow$ how?
5. Compare resulting TDD to identity TDD

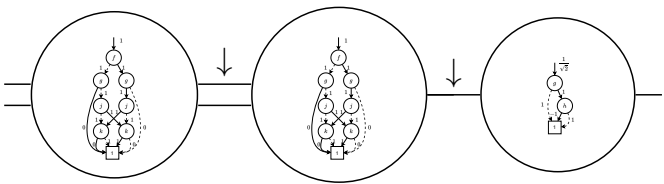


(TDDs on the right are only exemplary)

[1] C. B. Larsen, S. B. Olsen, K. G. Larsen, and C. Schilling. *Entropy* (2024).

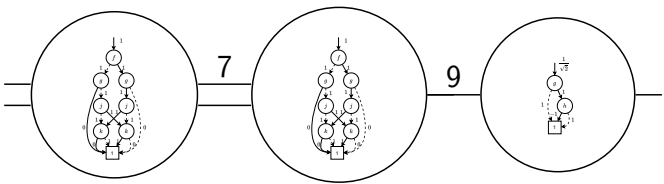
Lookahead heuristic for TDD contraction

- Greedy algorithm
- In each contraction step:
  1. Evaluate all possible contractions



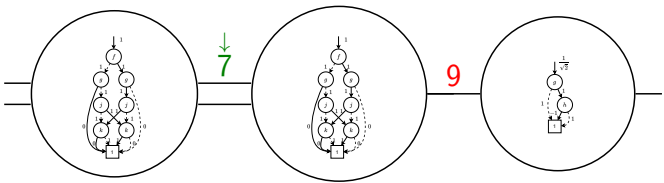
Lookahead heuristic for TDD contraction

- Greedy algorithm
- In each contraction step:
  1. Evaluate all possible contractions
  2. Measure size of resulting TDDs



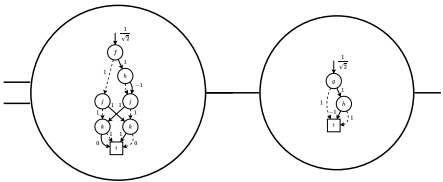
Lookahead heuristic for TDD contraction

- Greedy algorithm
- In each contraction step:
  1. Evaluate all possible contractions
  2. Measure size of resulting TDDs
  3. Execute a contraction with smallest resulting TDD



Lookahead heuristic for TDD contraction

- Greedy algorithm
- In each contraction step:
  1. Evaluate all possible contractions
  2. Measure size of resulting TDDs
  3. Execute a contraction with smallest resulting TDD



Lookahead heuristic for TDD contraction

- Greedy algorithm
- In each contraction step:
  1. Evaluate all possible contractions
  2. Measure size of resulting TDDs
  3. Execute a contraction with smallest resulting TDD
- Why should this scale?
  - Network is sparsely connected
  - Initial contractions (with many tensors) are cheap
  - Results can be stored for later iterations
- Still, step 1. is quite expensive

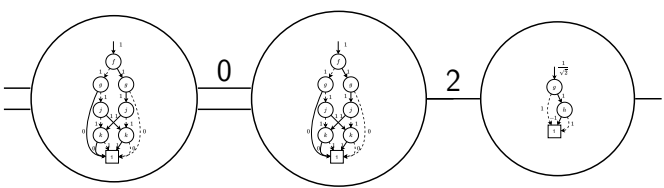
Counting heuristic for TDD contraction

- Goal: Imitate lookahead heuristic without expensive step 1
- Empirical observation: Lookahead heuristic prefers to distribute the contractions over the tensor network

1

Counting heuristic for TDD contraction

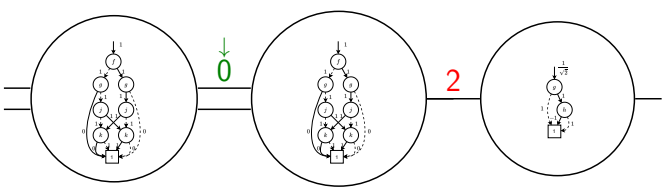
- Goal: Imitate lookahead heuristic without expensive step 1
- Empirical observation: Lookahead heuristic prefers to distribute the contractions over the tensor network
- In each contraction step:
  1. Select nodes with earliest participation in contraction[1]



[1] We set one edge to "2" to get an interesting example

Counting heuristic for TDD contraction

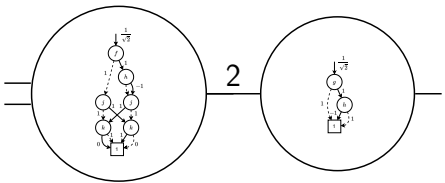
- Goal: Imitate lookahead heuristic without expensive step 1
- Empirical observation: Lookahead heuristic prefers to distribute the contractions over the tensor network
- In each contraction step:
  1. Select nodes with earliest participation in contraction[1]



[1] We set one edge to "2" to get an interesting example

Counting heuristic for TDD contraction

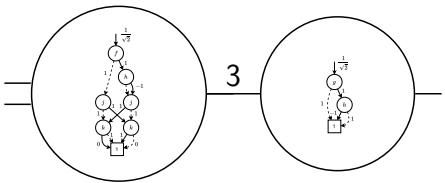
- Goal: Imitate lookahead heuristic without expensive step 1
- Empirical observation: Lookahead heuristic prefers to distribute the contractions over the tensor network
- In each contraction step:
  1. Select nodes with earliest participation in contraction[1]



[1] We set one edge to "2" to get an interesting example

Counting heuristic for TDD contraction

- Goal: Imitate lookahead heuristic without expensive step 1
- Empirical observation: Lookahead heuristic prefers to distribute the contractions over the tensor network
- In each contraction step:
  1. Select nodes with earliest participation in contraction[1]
  2. Update usage statistics of neighboring edges



[1] We set one edge to "2" to get an interesting example

Overview

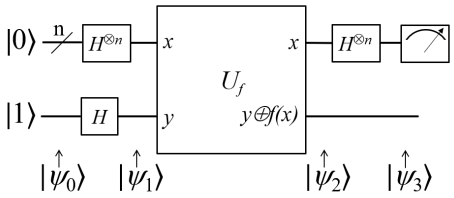
Quantum circuits in evaluation

- Circuits from MQT Bench[1] with varying number of qubits at two compilation levels (level 1 and 3 (out of 4)) with significantly different gate sets and layouts
  - Deutsch-Jozsa algorithm (DJ)
  - Greenberger-Horne-Zeilinger state preparation (GHZ)
  - Graph state preparation (GS)

[1] N. Quetschlich, L. Burgholzer, and R. Wille. *Quantum* (2023).

Deutsch-Jozsa algorithm (DJ)[1,2]

- Given $f\colon \{0,1\}^n \to \{0,1\}$ with promise that it is either
  - constant (100% "0" or 100% "1") or
  - balanced (50% "0" and 50% "1")
- Task: Determine which of the two cases it is
- Demonstrates exponential speed-up (requires a single shot)

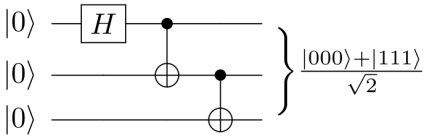


[1] D. Deutsch and R. Jozsa. *Proc. R. Soc. A* (1992).

[2] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. *Proc. R. Soc. A* (1998).

Greenberger-Horne-Zeilinger state preparation (GHZ)[1]

- The **GHZ state** generalizes the Bell state
- For 3 qubits: $\frac{|000\rangle + |111\rangle}{\sqrt{2}}$
- For k qubits: $\frac{|0\rangle^{\otimes k} + |1\rangle^{\otimes k}}{\sqrt{2}}$
- Used in quantum communication and cryptography protocols



[1] D. M. Greenberger, M. A. Horne, and A. Zeilinger. *Bell's theorem, quantum theory and conceptions of the universe*. 1989.

Graph state preparation (GS)[1]

- A **graph state** is a state that can be represented by a graph
- Each vertex corresponds to a qubit
- $|G\rangle = \prod_{(u,v)\in E} CZ^{(u,v)} |+\rangle^{\otimes |V|}$

  where $CZ^{(u,v)}$ is the corresponding controlled-Z gate
- Used, e.g., in quantum error-correcting codes

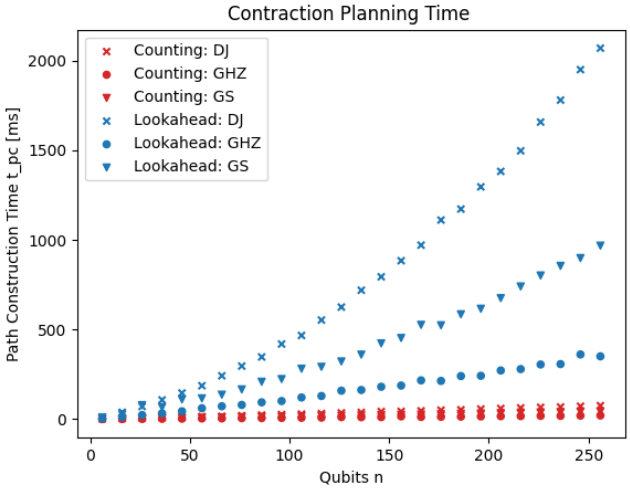
[1] M. Hein, J. Eisert, and H. J. Briegel. *Physical Review A—Atomic, Molecular, and Optical Physics* (2004).

Types of experiments

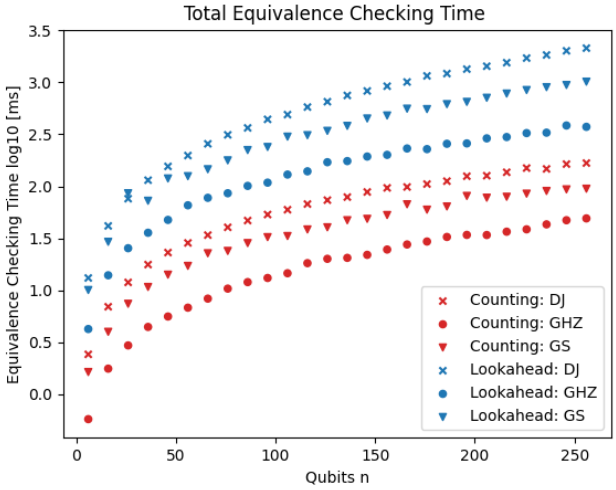
1. Compare proposed heuristics to each other
2. Compare proposed heuristics to heuristics from cotengra[1]
3. Compare proposed heuristics to QCEC[2] (uses decision diagrams)

[1] J. Gray and S. Kourtis. *Quantum* (2021).
[2] L. Burgholzer and R. Wille. *Softw. Impacts* (2021).

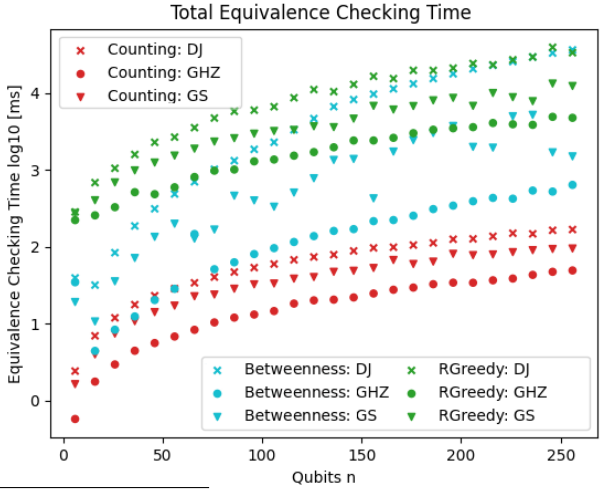
Comparison of proposed heuristics



Comparison of proposed heuristics

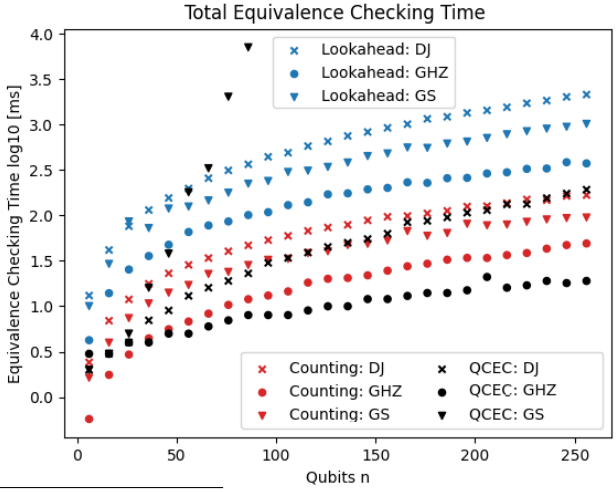


Comparison to cotengra[1]



[1] J. Gray and S. Kourtis. *Quantum* (2021).

Comparison to QCEC[1]



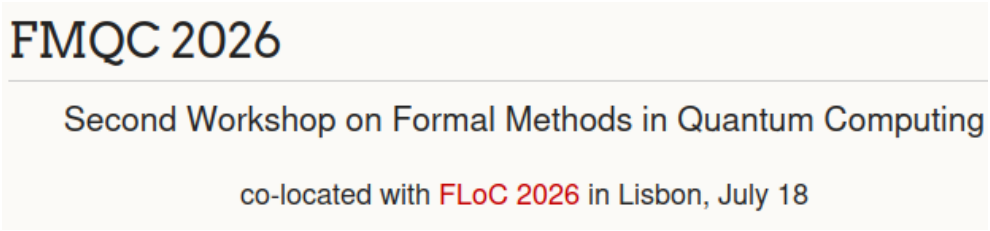
[1] L. Burgholzer and R. Wille. *Softw. Impacts* (2021).

Overview

Conclusion

- Algorithm for equivalence checking of quantum circuits:
  - Integration of "reverse scheme" and TDD networks
  - Lookahead heuristic (greedy)
  - Counting heuristic (cheap approximation)

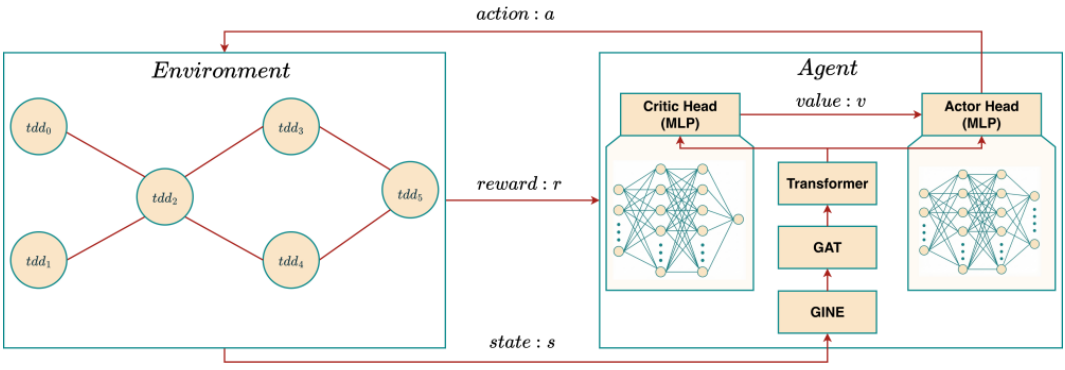
Upcoming events



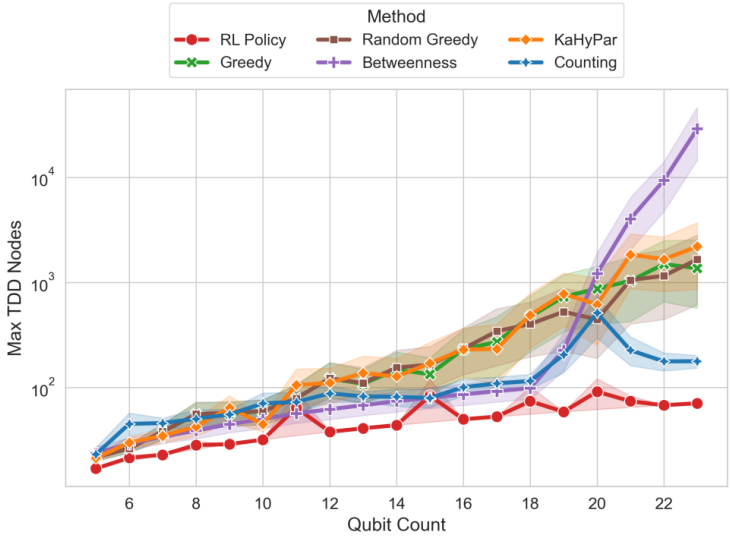
We are hiring

- 1 PhD student and 4 postdoc positions
  - Basic Research Center on quantum communication
  - Grand Solutions project on quantum software stack
  - EU Quantum Center of Excellence
- Open call for 2 postdoc positions (deadline May 25)

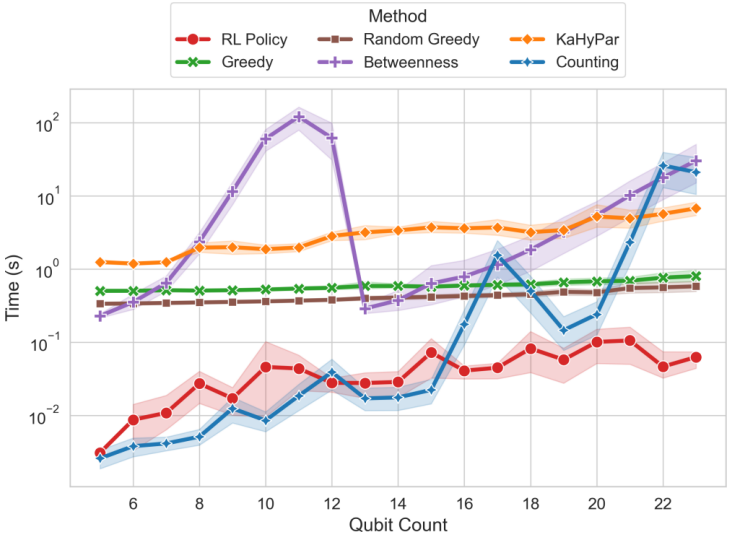
Learning contraction orders (WIP)



Learning contraction orders (WIP)



Learning contraction orders (WIP)



Learning contraction orders (WIP)

Qubits	Method	Peak Size	Time (s)	Status
30	Hybrid heuristic/RL (Ours)	**220**	**0.221**	Success
	KaHyPar	32,748	3.996	Success
	Greedy	33,249	24.571	Success
	Random Greedy	165,373	25.828	Success
	Counting / Betweenness	–	–	TO
38	Hybrid heuristic/RL (Ours)	**127**	**0.243**	Success
	KaHyPar	2,204,566	99.321	Success
	Greedy / Random Greedy / Counting	–	–	TO
50	Hybrid heuristic/RL (Ours)	**15,289**	**8.200**	Success
	All Baselines	–	–	TO