
Equivalence Checking of Quantum Circuits

Christian Schilling
christianms@cs.aau.dk

December 9, 2024

Quantum Seminar @ Aalborg University

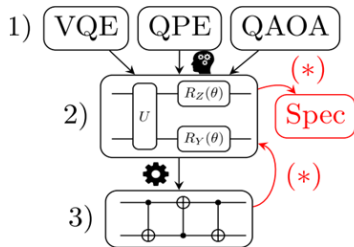


DEPARTMENT
OF COMPUTER
SCIENCE

Design automation in quantum computing at DEIS group

- 1) Application-driven algorithm
- 2) High-level circuit
- 3) Translation to basic gates

(*) Equivalent? Correct? Optimal?



Lukáš Holík



Kim Guldstrand Larsen



Christian Schilling



Max Tschaikowski



This talk: Based on joint work¹

Christian Bøgh Larsen



Simon Brun Olsen



Kim Guldstrand Larsen



VILLUM FONDEN



¹C. B. Larsen, S. B. Olsen, K. G. Larsen, and C. Schilling. “Contraction heuristics for tensor decision diagrams”. *Entropy* (2024).

Overview

Short overview of quantum computing

Equivalence checking of quantum circuits

Existing approaches to equivalence checking

Equivalence check based on tensor decision diagrams

Empirical evaluation

Conclusion and future work

Overview

Short overview of quantum computing

Equivalence checking of quantum circuits

Existing approaches to equivalence checking

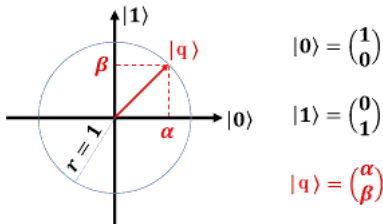
Equivalence check based on tensor decision diagrams

Empirical evaluation

Conclusion and future work

Single qubit

- Basis states $|0\rangle$ and $|1\rangle$
- Superposition $|q\rangle = \alpha|0\rangle + \beta|1\rangle$
where $\alpha, \beta \in \mathbb{C}$ s.t. $|\alpha|^2 + |\beta|^2 = 1$
- Written as vector: $|q\rangle \equiv \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$



Multiple qubits (unentangled)

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

Quantum gates and quantum circuits

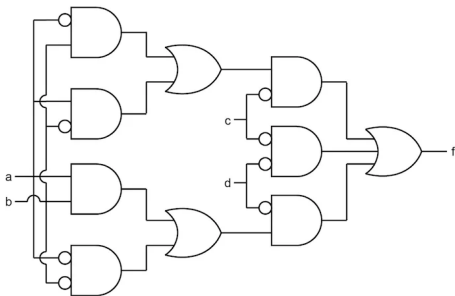
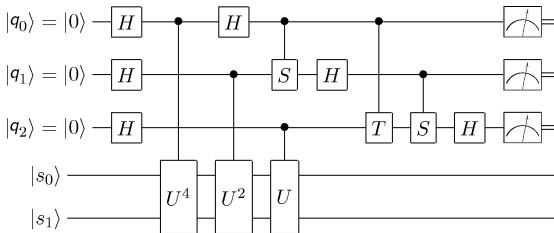
- Analogous to classical logic gates and circuits
- Quantum gates transform quantum states
Equivalent to (unitary) matrix multiplication with state vector

$$|q'\rangle = U |q\rangle$$

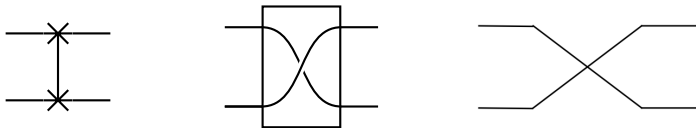
- Quantum circuit: Arrangement of (multiple) gates

$$|q'\rangle = U_m \cdot U_{m-1} \cdots U_2 \cdot U_1 |q\rangle$$

Quantum gates and quantum circuits



Example: SWAP gate



- Swapping of two qubits

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

Example: Controlled NOT gate

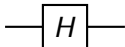


- Controlled negation of second qubit

$$\begin{aligned} CNOT(|q_0q_1\rangle) &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha_0\alpha_1 \\ \alpha_0\beta_1 \\ \beta_0\alpha_1 \\ \beta_0\beta_1 \end{bmatrix} = \begin{bmatrix} \alpha_0\alpha_1 \\ \alpha_0\beta_1 \\ \beta_0\beta_1 \\ \beta_0\alpha_1 \end{bmatrix} \\ &= |q_0\rangle \otimes |q_0 \oplus q_1\rangle \end{aligned}$$

where \oplus is addition modulo 2 (“XOR”)

Example: Hadamard gate



- Creates a superposition

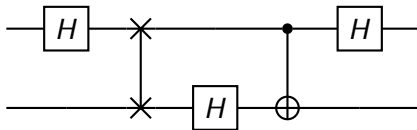
$$H(|q\rangle) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} \alpha + \beta \\ \alpha - \beta \end{bmatrix}$$

Examples

$$H(|0\rangle) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \equiv \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle$$

$$H(|1\rangle) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \equiv \frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle$$

Example: A random circuit



Overview

Short overview of quantum computing

Equivalence checking of quantum circuits

Existing approaches to equivalence checking

Equivalence check based on tensor decision diagrams

Empirical evaluation

Conclusion and future work

Quantum circuit compilation

- When designing quantum algorithms (= circuits), it is useful to have many types of gates available
- Real quantum computers only support a few types of gates
- A **compiler** translates a high-level circuit with many gate types to a low-level circuit with few gate types

Quantum circuit compilation

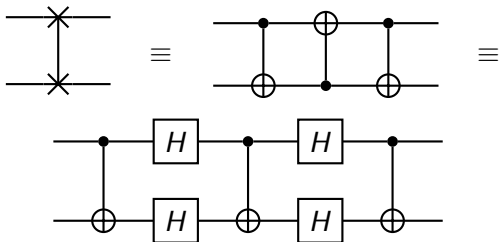
- When designing quantum algorithms (= circuits), it is useful to have many types of gates available
- Real quantum computers only support a few types of gates
- A **compiler** translates a high-level circuit with many gate types to a low-level circuit with few gate types
- Different types of gates incur different amounts of error
- Deeper circuits incur more errors
- A compiler also tries to optimize circuits to reduce errors

Quantum circuit compilation

- When designing quantum algorithms (= circuits), it is useful to have many types of gates available
- Real quantum computers only support a few types of gates
- A **compiler** translates a high-level circuit with many gate types to a low-level circuit with few gate types
- Different types of gates incur different amounts of error
- Deeper circuits incur more errors
- A compiler also tries to optimize circuits to reduce errors
- Important that the compiled circuits are **equivalent**

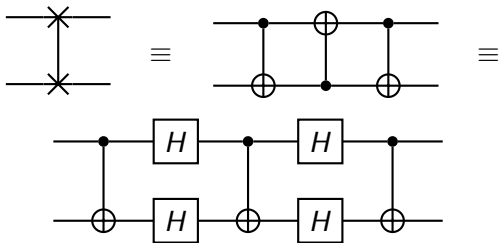
How to implement a SWAP gate?

- The SWAP gate is **equivalent** to three CNOT gates



How to implement a SWAP gate?

- The SWAP gate is **equivalent** to three CNOT gates



- We can easily prove this by comparing the matrices

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

Algorithm to check equivalence of quantum circuits

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

Algorithm to check equivalence of quantum circuits

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

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

Algorithm to check equivalence of quantum circuits

- Given, two circuits C_1, C_2
- Question: Are they equivalent ($C_1 \equiv C_2$)?
- Simple algorithm
 1. Compute matrix representations 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**
 n qubits $\rightsquigarrow 2^n \times 2^n$

Equivalence checking is hard

- Checking exact equivalence is NQP-complete¹
- Checking approximate equivalence is QMA-complete^{2,3}
- Problems in these complexity classes are widely believed to require exponential computations in the worst case

¹Y. Tanaka. “Exact non-identity check is NQP-complete”. *Int. J. Quantum Inf.* (2010).

²D. Janzing, P. Wocjan, and T. Beth. ““Non-identity-check” is QMA-complete”. *Int. J. Quantum Inf.* (2005).

³Z. Ji and X. Wu. *Non-identity check remains QMA-complete for short circuits*. 2009. arXiv: 0906.5416.

Overview

Short overview of quantum computing

Equivalence checking of quantum circuits

Existing approaches to equivalence checking

Equivalence check based on tensor decision diagrams

Empirical evaluation

Conclusion and future work

Alternative “reverse scheme” for equivalence¹

$$C_1 \equiv C_2 \iff \exists \theta: U_1 = e^{i\theta} \cdot U_2$$

$$\iff \exists \theta: U_1 \cdot U_2^\dagger = e^{i\theta} \cdot I \iff C_1 C_2^R \equiv C_I$$

- C^R is the “reversed” circuit
- Allows to combine both circuits

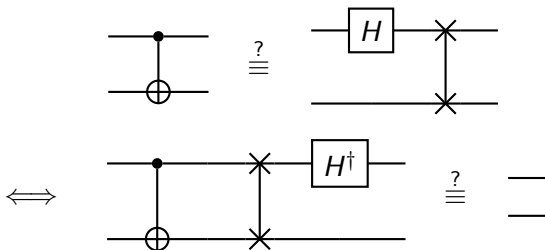
¹G. F. Viamontes, I. L. Markov, and J. P. Hayes. “Checking equivalence of quantum circuits and states”. *ICCAD*. 2007.

Alternative “reverse scheme” for equivalence¹

$$C_1 \equiv C_2 \iff \exists \theta: U_1 = e^{i\theta} \cdot U_2$$

$$\iff \exists \theta: U_1 \cdot U_2^\dagger = e^{i\theta} \cdot I \iff C_1 C_2^R \equiv C_I$$

- C^R is the “reversed” circuit
- Allows to combine both circuits



¹G. F. Viamontes, I. L. Markov, and J. P. Hayes. “Checking equivalence of quantum circuits and states”. *ICCAD*. 2007.

Approaches to equivalence checking

- ZX-calculus¹
- Encoding with decision diagrams²
- Tensor network contraction³
- Simulation-based approach for the Clifford group⁴
- Weighted model counting⁵

¹T. Peham, L. Burgholzer, and R. Wille. “Equivalence checking of quantum circuits with the ZX-calculus”. *IEEE J. Emerg. Sel. Topics Circuits Syst.* (2022).

²L. Burgholzer and R. Wille. “Advanced equivalence checking for quantum circuits”. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* (2021).

³R. Orús. “Tensor networks for complex quantum systems”. *Nature Reviews Physics* (2019).

⁴D. Thanos, T. Coopmans, and A. Laarman. “Fast equivalence checking of quantum circuits of Clifford gates”. *ATVA*. 2023.

⁵J. Mei, T. Coopmans, M. M. Bonsangue, and A. Laarman. “Equivalence checking of quantum circuits by model counting”. *IJCAR*. 2024.

Approaches to equivalence checking

- ZX-calculus¹
- Encoding with decision diagrams² ← relevant later
- Tensor network contraction³ ← relevant later
- Simulation-based approach for the Clifford group⁴
- Weighted model counting⁵

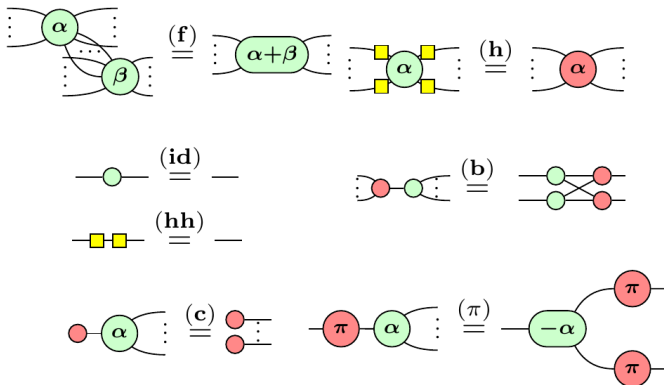
¹T. Peham, L. Burgholzer, and R. Wille. “Equivalence checking of quantum circuits with the ZX-calculus”. *IEEE J. Emerg. Sel. Topics Circuits Syst.* (2022).

²L. Burgholzer and R. Wille. “Advanced equivalence checking for quantum circuits”. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* (2021).

³R. Orús. “Tensor networks for complex quantum systems”. *Nature Reviews Physics* (2019).

⁴D. Thanos, T. Coopmans, and A. Laarman. “Fast equivalence checking of quantum circuits of Clifford gates”. *ATVA*. 2023.

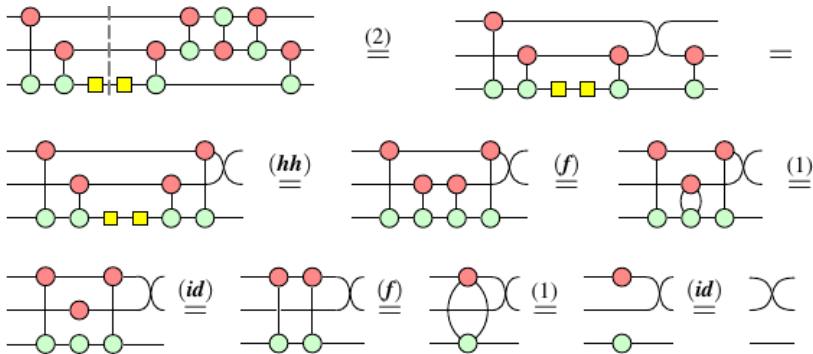
⁵J. Mei, T. Coopmans, M. M. Bonsangue, and A. Laarman. “Equivalence checking of quantum circuits by model counting”. *IJCAR*. 2024.

Based on ZX-calculus^{1,2}

Axioms

¹B. Coecke and R. Duncan. "Interacting quantum observables". *ICALP*. 2008.

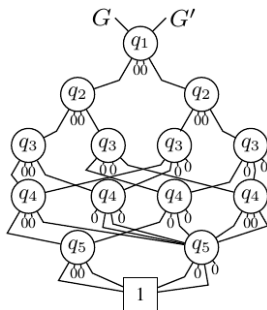
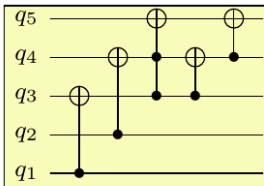
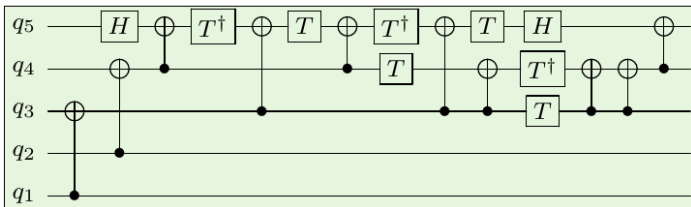
²T. Peham, L. Burgholzer, and R. Wille. "Equivalence checking of quantum circuits with the ZX-calculus". *IEEE J. Emerg. Sel. Topics Circuits Syst.* (2022).

Based on ZX-calculus^{1,2}

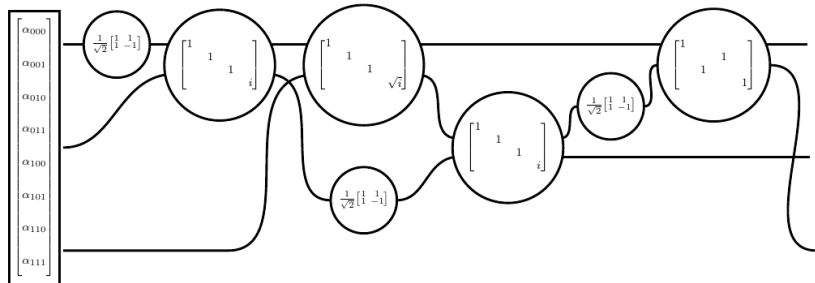
Example proof

¹B. Coecke and R. Duncan. "Interacting quantum observables". *ICALP*. 2008.

²T. Peham, L. Burgholzer, and R. Wille. "Equivalence checking of quantum circuits with the ZX-calculus". *IEEE J. Emerg. Sel. Topics Circuits Syst.* (2022).

Based on decision diagrams¹ G  G' 

¹L. Burgholzer and R. Wille. “Advanced equivalence checking for quantum circuits”. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* (2021).

Based on tensor network contraction^{1,2}

¹X. Hong, Y. Feng, S. Li, and M. Ying. “Equivalence checking of dynamic quantum circuits”. *ICCAD*. ed. by T. Mitra, E. F. Y. Young, and J. Xiong. 2022.

²L. Burgholzer, A. Ploier, and R. Wille. “Simulation paths for quantum circuit simulation with decision diagrams - what to learn from tensor networks, and what not”. *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 $j \in \{1, 2, \dots, n\}$, we have $UZ_jU^\dagger = VZ_jV^\dagger$; and*
2. *for all $j \in \{1, 2, \dots, n\}$, we have $UX_jU^\dagger = VX_jV^\dagger$.*

Here, as before, we have denoted $Z_j = I \otimes \dots \otimes I \otimes Z \otimes I \otimes \dots \otimes I$, i.e. an n -fold tensor product of identity gates I with the Pauli Z gate at the j -th position. Analogously, $X_j = I \otimes \dots \otimes I \otimes X \otimes I \otimes \dots \otimes I$ 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 encoding as a **weighted model counting** problem²

¹D. Thanos, T. Coopmans, and A. Laarman. “Fast equivalence checking of quantum circuits of Clifford gates”. *ATVA*. 2023.

²J. Mei, T. Coopmans, M. M. Bonsangue, and A. Laarman. “Equivalence checking of quantum circuits by model counting”. *IJCAR*. 2024.

Overview

Short overview of quantum computing

Equivalence checking of quantum circuits

Existing approaches to equivalence checking

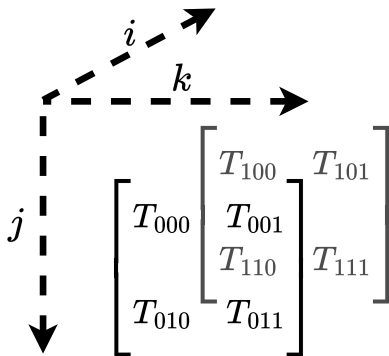
Equivalence check based on tensor decision diagrams

Empirical evaluation

Conclusion and future work




Tensors

- Generalization of vectors / matrices to higher dimensions



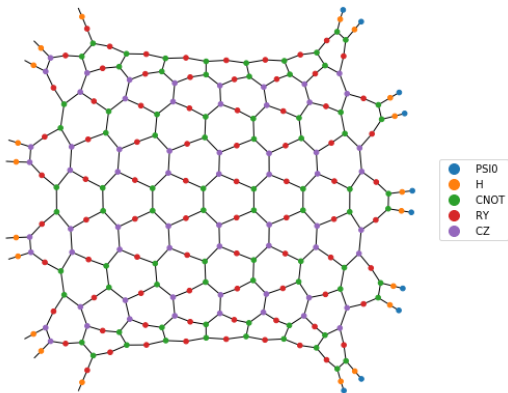
Tensors

- 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
- Shared edges can be **contracted**
Corresponds to matrix-vector and matrix-matrix multiplication for special cases

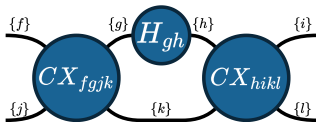
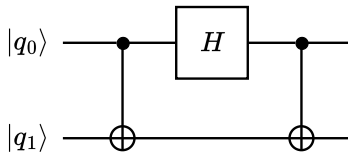
$$\begin{array}{c} \text{---} \bullet \text{---} \bullet \\ i \quad j \end{array} = \sum_j A_{ij} \underbrace{v_j}$$

$$\begin{array}{c} \text{---} \bullet \text{---} \bullet \text{---} \\ i \quad j \quad k \end{array} = \sum_j A_{ij} \underbrace{B_{jk}} = AB$$

Example

1. Initial tensor network has one tensor for each gate

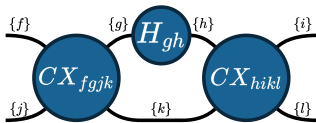
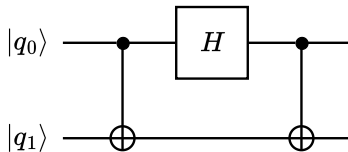
Three choices for contraction ($\{g\}$, $\{h\}$, $\{k\}$)



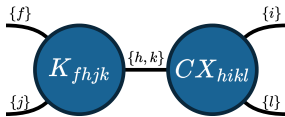
1.

Example

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



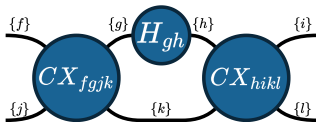
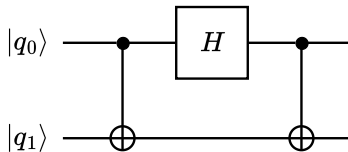
1.



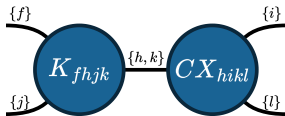
2.

Example

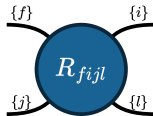
1. Initial tensor network has one tensor for each gate
Three choices for contraction ($\{g\}$, $\{h\}$, $\{k\}$)
2. Contraction of CX_{fgjk} and H_{gh} via $\{g\}$
3. Contraction of remaining two tensors



1.



2.



3.

Application: Quantum simulation on classical computer²

- Contract tensors in smart orders
- Different contraction heuristics to minimize floating-point operations, size, etc.¹

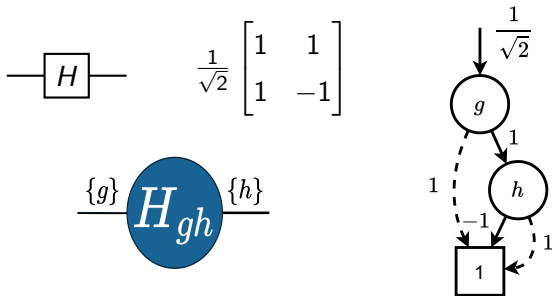
¹J. Gray and S. Kourtis. “Hyper-optimized tensor network contraction”. *Quantum* (2021).

²I. L. Markov and Y. Shi. “Simulating quantum computation by contracting tensor networks”. *SIAM J. Comput.* (2008).

Tensor decision diagrams (TDDs)

- Alternative, unique representation of a tensor
- Only informal introduction here

Example: Hadamard gate, tensor, and TDD

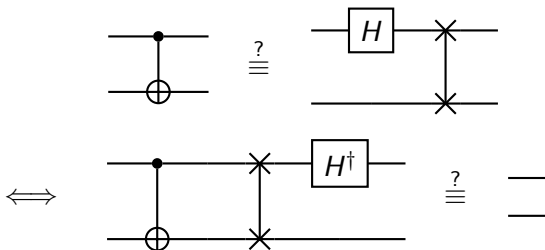


Alternative “reverse scheme” for equivalence¹

$$C_1 \equiv C_2 \iff \exists \theta: U_1 = e^{i\theta} \cdot U_2$$

$$\iff \exists \theta: U_1 \cdot U_2^\dagger = e^{i\theta} \cdot I \iff C_1 C_2^R \equiv C_I$$

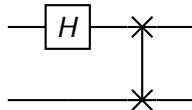
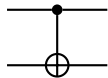
- C^R is the “reversed” circuit
- Allows to combine both circuits



¹G. F. Viamontes, I. L. Markov, and J. P. Hayes. “Checking equivalence of quantum circuits and states”. *ICCAD*. 2007.

Algorithm

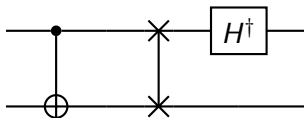
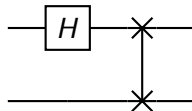
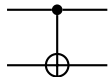
Given: Quantum circuits C_1 , C_2



Algorithm

Given: Quantum circuits C_1, C_2

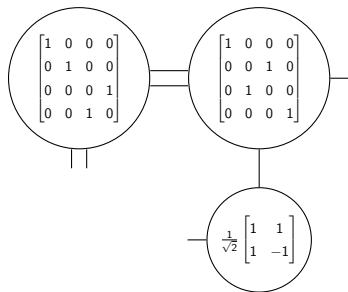
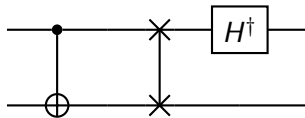
1. Construct combined circuit $C_1 C_2^\dagger$



Algorithm

Given: Quantum circuits C_1, C_2

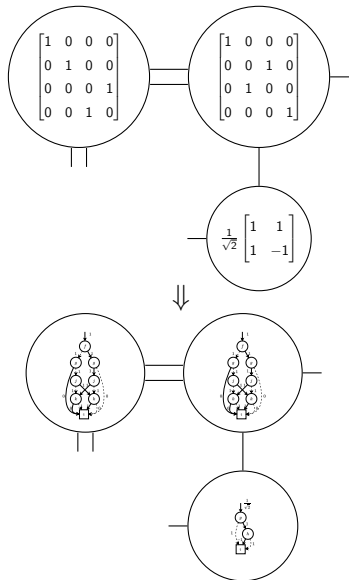
1. Construct combined circuit $C_1 C_2^\dagger$
2. Convert $C_1 C_2^\dagger$ to tensor network



Algorithm

Given: Quantum circuits C_1, C_2

1. Construct combined circuit $C_1 C_2^\dagger$
2. Convert $C_1 C_2^\dagger$ to tensor network
3. Convert all tensors to TDDs

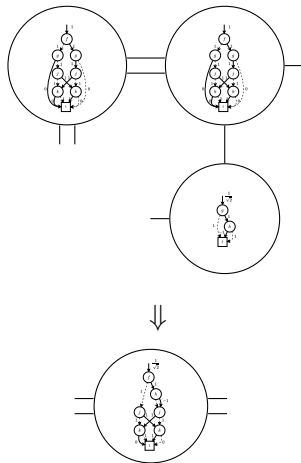


(TDDs on the right are only exemplary)

Algorithm

Given: Quantum circuits C_1, C_2

1. Construct combined circuit $C_1 C_2^\dagger$
2. Convert $C_1 C_2^\dagger$ to tensor network
3. Convert all tensors to TDDs
4. Contract TDD network

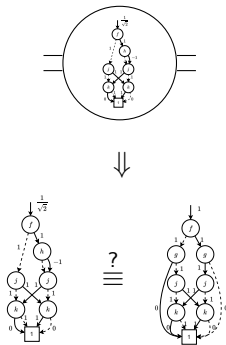


(TDDs on the right are only exemplary)

Algorithm

Given: Quantum circuits C_1, C_2

1. Construct combined circuit $C_1 C_2^\dagger$
2. Convert $C_1 C_2^\dagger$ to tensor network
3. Convert all tensors to TDDs
4. Contract TDD network
5. Compare TDD to identity TDD

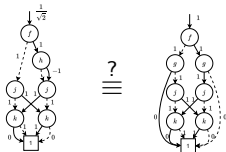
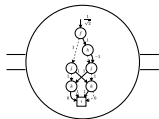


(TDDs on the right are only exemplary)

Algorithm

Given: Quantum circuits C_1, C_2

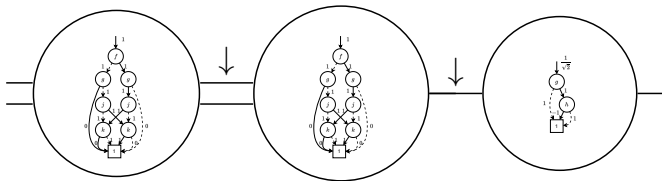
1. Construct combined circuit $C_1 C_2^\dagger$
2. Convert $C_1 C_2^\dagger$ to tensor network
3. Convert all tensors to TDDs
4. **Contract TDD network** ← how?
5. Compare TDD to identity TDD



(TDDs on the right are only exemplary)

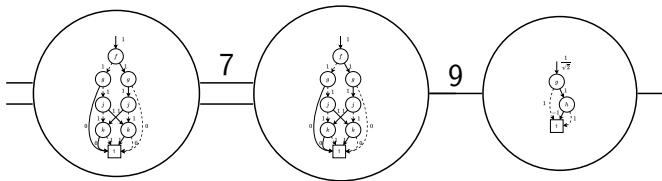
Lookahead heuristic for TDD contraction

- Greedy algorithm (finds a local optimum)
- In each contraction step:
 1. Evaluate all possible contractions



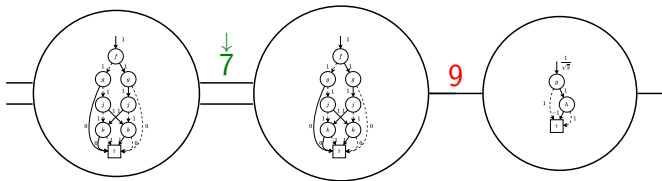
Lookahead heuristic for TDD contraction

- Greedy algorithm (finds a local optimum)
- In each contraction step:
 1. Evaluate all possible contractions
 2. Measure size of resulting TDDs



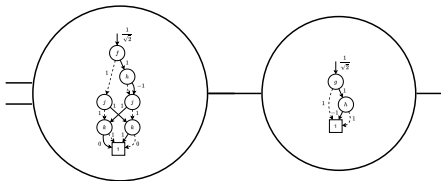
Lookahead heuristic for TDD contraction

- Greedy algorithm (finds a local optimum)
- In each contraction step:
 1. Evaluate all possible contractions
 2. Measure size of resulting TDDs
 3. Execute contraction with smallest output



Lookahead heuristic for TDD contraction

- Greedy algorithm (finds a local optimum)
- In each contraction step:
 1. Evaluate all possible contractions
 2. Measure size of resulting TDDs
 3. Execute contraction with smallest output



Lookahead heuristic for TDD contraction

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

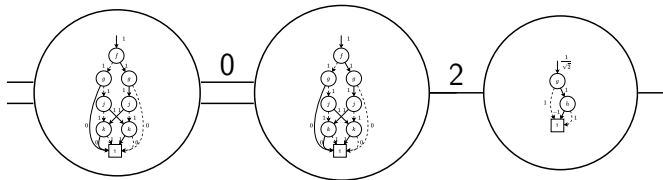
Counting heuristic for TDD contraction

- Goal: Imitate lookahead heuristic without executing step 1
- Empirical observation: Lookahead heuristic prefers to distribute

¹We set one edge to “2” to get an interesting example

Counting heuristic for TDD contraction

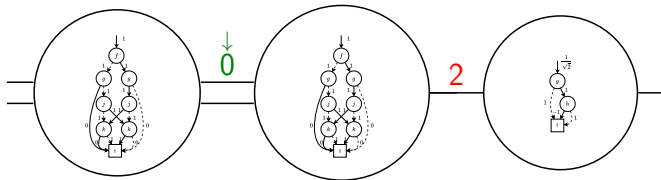
- Goal: Imitate lookahead heuristic without executing step 1
- Empirical observation: Lookahead heuristic prefers to distribute
- In each contraction step:
 1. Select a pair of nodes with longest non-usage¹



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

Counting heuristic for TDD contraction

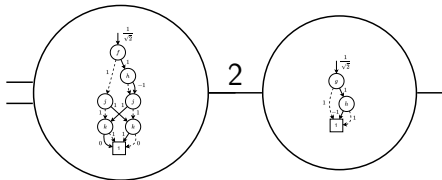
- Goal: Imitate lookahead heuristic without executing step 1
- Empirical observation: Lookahead heuristic prefers to distribute
- In each contraction step:
 1. Select a pair of nodes with longest non-usage¹



¹We set one edge to “2” to get an interesting example

Counting heuristic for TDD contraction

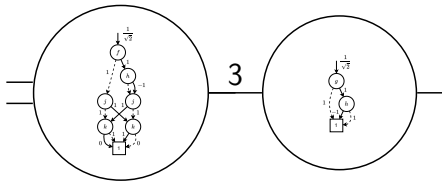
- Goal: Imitate lookahead heuristic without executing step 1
- Empirical observation: Lookahead heuristic prefers to distribute
- In each contraction step:
 1. Select a pair of nodes with longest non-usage¹



¹We set one edge to “2” to get an interesting example

Counting heuristic for TDD contraction

- Goal: Imitate lookahead heuristic without executing step 1
- Empirical observation: Lookahead heuristic prefers to distribute
- In each contraction step:
 1. Select a pair of nodes with longest non-usage¹
 2. Update usage statistics (avoided in implementation)



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

Overview

Short overview of quantum computing

Equivalence checking of quantum circuits

Existing approaches to equivalence checking

Equivalence check based on tensor decision diagrams

Empirical evaluation

Conclusion and future work

Quantum circuits in evaluation

- Circuits from MQT Bench¹ 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)
 - Quantum Fourier transformation (entangled qubits) (QFTE)
 - Real amplitudes ansatz with random parameters (RAR)
 - W state preparation (WS)

¹N. Quetschlich, L. Burgholzer, and R. Wille. “MQT Bench: Benchmarking software and design automation tools for quantum computing”. *Quantum* (2023).

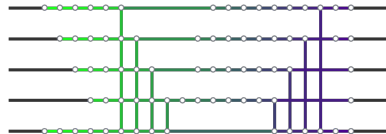
Types of experiments

1. Compare heuristics to each other
2. Compare heuristics to heuristics from cotengra¹
3. Compare heuristics to QCEC² (decision diagrams)

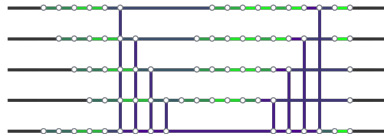
¹J. Gray and S. Kourtis. “Hyper-optimized tensor network contraction”. *Quantum* (2021).

²L. Burgholzer and R. Wille. “QCEC: A JKQ tool for quantum circuit equivalence checking”. *Softw. Impacts* (2021).

Different contraction order heuristics (DJ circuit)



From left to right



Lookahead



Counting



cotengra's Betweenness

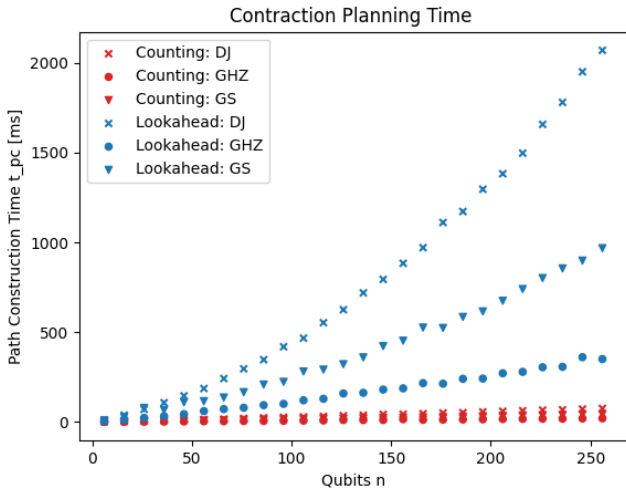


cotengra's RandomGreedy

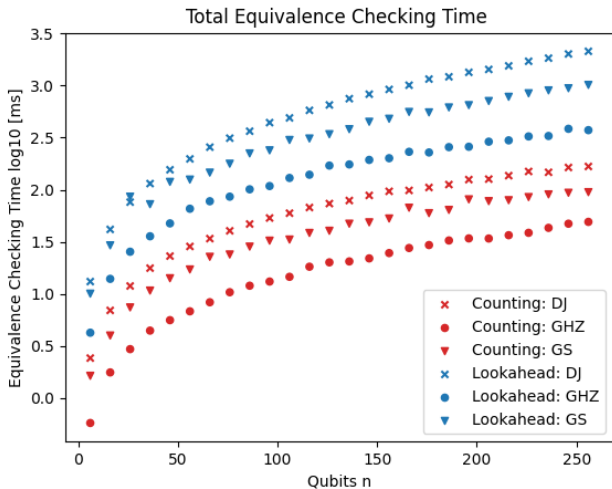


QCEC's proportional

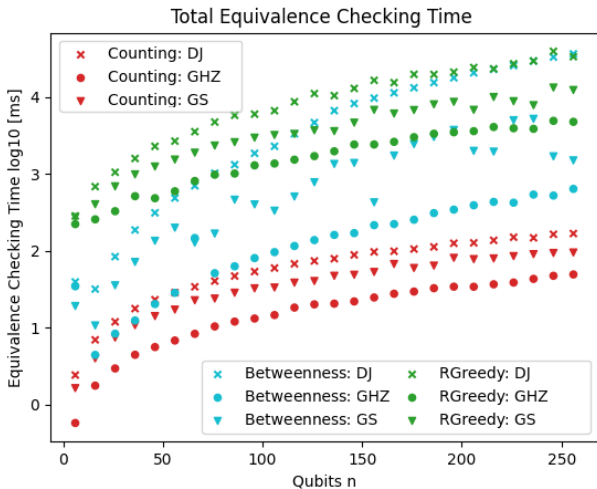
Comparison of own heuristics



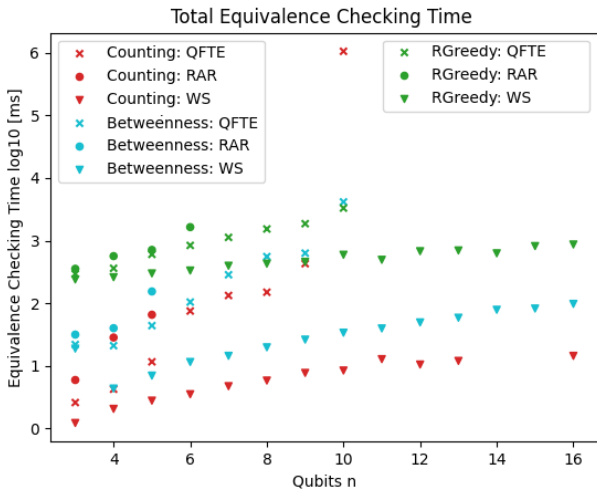
Comparison of own heuristics



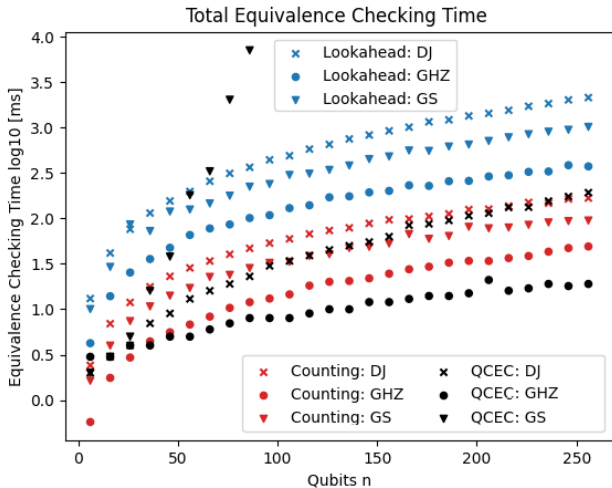
Comparison to cotengra



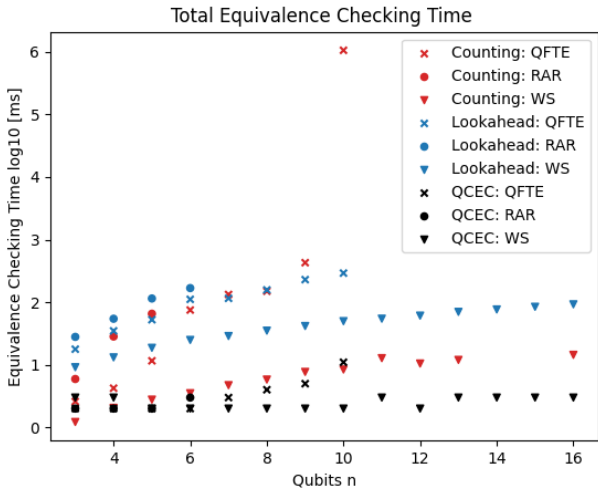
Comparison to cotengra



Comparison to QCEC



Comparison to QCEC



Overview

Short overview of quantum computing

Equivalence checking of quantum circuits

Existing approaches to equivalence checking

Equivalence check based on tensor decision diagrams

Empirical evaluation

Conclusion and future work

Conclusion

- Integration of “reverse scheme” and TDD networks
- Lookahead heuristic (greedy)
- Counting heuristic (cheap approximation)
- Evaluation:
 - Outperforms cotengra’s heuristics
 - Often keeps up with QCEC

Future work

- Evaluate beyond equivalence checking
- Exploit parallelization (CPU, GPU)
- Find other heuristics
 - Generalize tensor network heuristics to TDD networks
 - Employ machine learning
- New PhD project, starting soon

