Quantum Circuit Optimization as SAT

Irfansha Shaik

March 31, 2025

Aarhus University, Denmark
Kvantify, Denmark

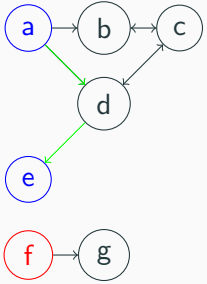
Graph Search Problems

Graph search is boring or is it?

Reachability

Two nodes in a graph are reachable if there exists a path between them.

From node a, node e is reachable but node f is not reachable

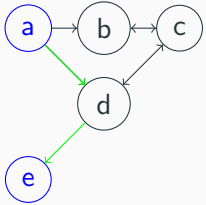


Graph search is boring or is it?

Shortest Path Problem

Compute the shortest sequence of nodes that reaches from source to target.

From a to e, the shortest path is of length 2 via node d.

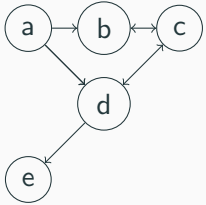


Graph search is boring or is it?

Travelling Salesman Problem

Visit all nodes uniquely and return to the source node

No solution exists!



Graph search problems are very important!

- Planning and scheduling
- Social Networks
- Quantum Circuit Synthesis (Aha!)
- ...
- Any NP-complete problem can be encoded as a graph search problem

How to solve graph problems?

- Breath First Search, Depth First Search
- Heuristic Algorithms
- Specialized Algorithms taking advantage of structure
- ...
- Domain Independent Solving

Domain Independent Solving

- Pick a **problem**
- Generate a **good encoding**
- Solve with a **Domain Independent Solver**
- Pray that it works

Domain Independent Solving

- Pick a **problem**
- Generate a **good encoding**
- Solve with a **Domain Independent Solver**
- ~~Pray that it works~~ Improve the encoding

How to encode graph reachability?

$\exists n_0, \ldots, n_k$

$n_0 = s \quad \wedge$

$n_k = t \quad \wedge$

$\bigwedge_{i=0}^{k-1} \text{Neighbour}(n_i, n_{i+1})$

$\exists$ a sequence of k+1 nodes, such that

n_0 is the source node

n_k is the target node

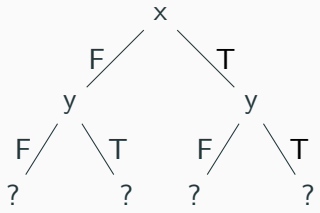
every n_i, n_{i+1} are neighbours

Propositional Satisfiability (NP-complete)

Given a boolean formula ϕ, Propositional Satisfiability (SAT) is finding the existence of an assignment to ϕ that makes it True.

$$\exists x \quad \exists y$$

$$x \leftrightarrow y$$



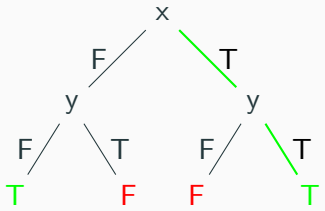
Propositional Satisfiability (NP-complete)

Given a boolean formula ϕ, Propositional Satisfiability (SAT) is finding the existence of an assignment to ϕ that makes it True.

$$\exists x \quad \exists y$$

$$x \leftrightarrow y$$

True



Key Idea!

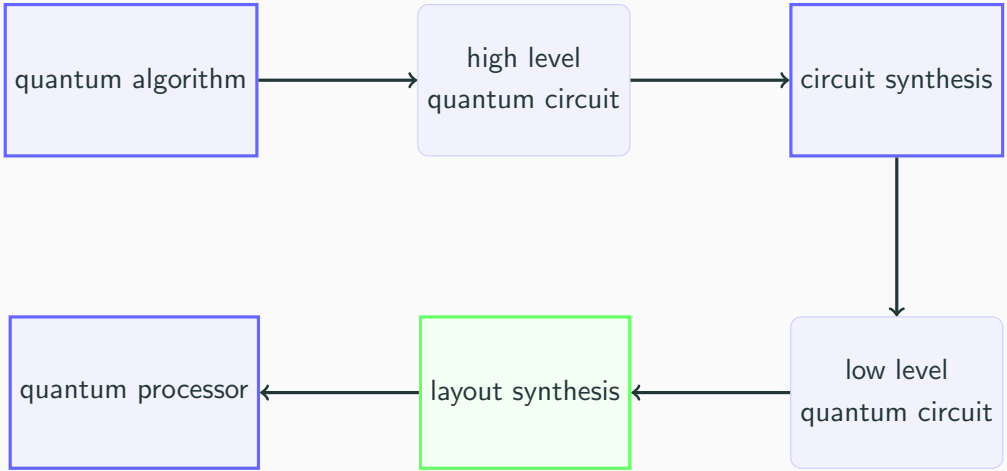
- Pick a **problem** (Perhaps Quantum Circuit Optimization?)
- Generate a good **SAT** encoding
- Solve with industrial SAT solvers
- ~~Pray that it works~~ Improve the encoding with domain specific knowledge

Quantum Computing

- Quantum computing promises a new way of computing for some hard problems.
- Using quantum bits or qubits instead of classical bits.
- Instead of 0 or 1, a qubit can also be in a superposition of 0 and 1.

Insert some mysterious looking quantum processor

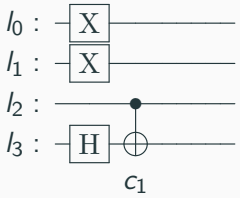
Quantum Compilation Pipeline



Example: Adder circuit

```
OPENQASM 2.0;
qreg q[4];
x q[0];
x q[1];
h q[3];
cx q[2], q[3]; // c1
...
```

Quantum gates in Open Quantum Assembly (OPENQASM) representation

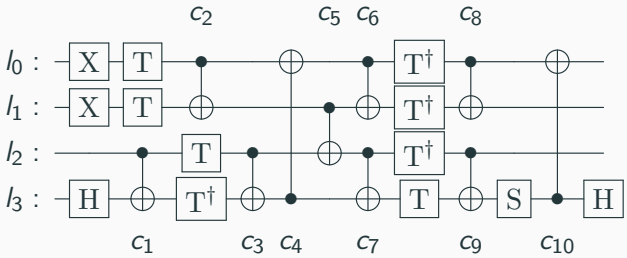


Graphical representation (reads left to right)

Example: Adder circuit

```
OPENQASM 2.0;
qreg q[4];
x q[0];
x q[1];
h q[3];
cx q[2], q[3]; // c1
...
```

Quantum gates in OPENQASM representation

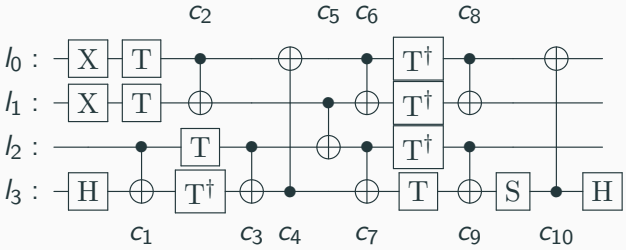


Graphical representation

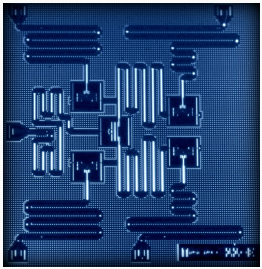
Quantum Layout Synthesis

Layout Synthesis

- Mapping a quantum circuit to some quantum architecture is Layout Synthesis.
- Mapping logical qubits to physical qubits such that all gates are executable respecting dependencies.



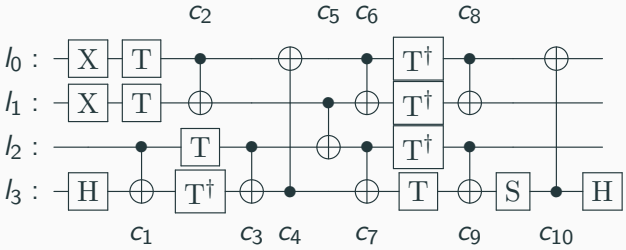
4-qubit Adder circuit



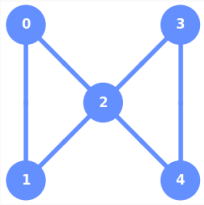
5-qubit IBM-QX5 platform

Layout Synthesis

- Mapping a quantum circuit to some quantum architecture is Layout Synthesis.
- Mapping logical qubits to physical qubits such that all gates are executable respecting dependencies.

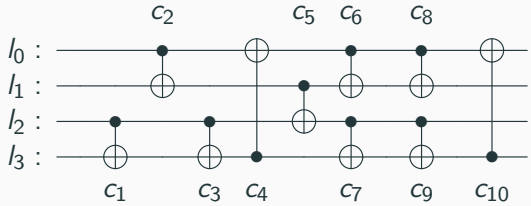


4-qubit Adder circuit

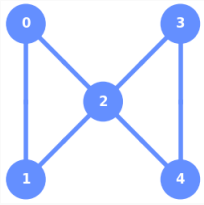


Coupling graph of IBM-QX5

- Mapping a quantum circuit to some quantum architecture is Layout Synthesis.
- Mapping logical qubits to physical qubits such that **all 2-qubit gates are applied on neighboring physical qubits**.

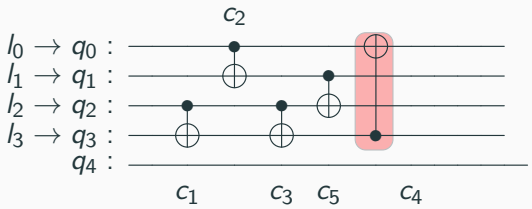


Adder circuit without single qubit gates

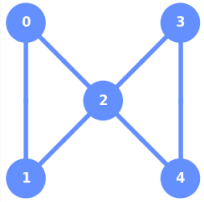


Coupling graph of IBM-QX5

Trying a simple initial mapping:

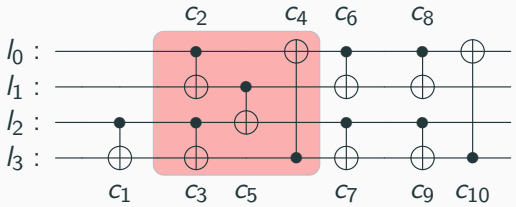


CNOT c_4 can not be scheduled

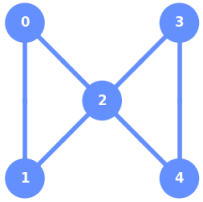


(0,3) not connected

No valid initial mapping exists.



CNOTs c2, c3, c4, and c5 form a square

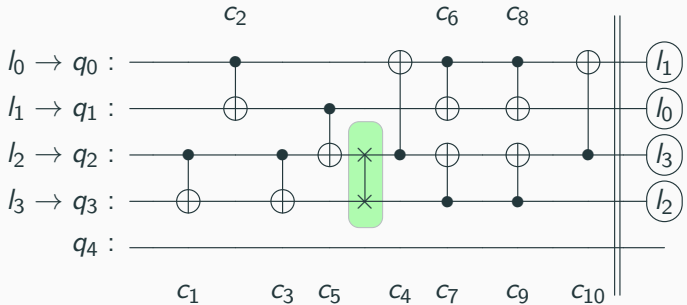


Has no square ☹

Example: Mapped Adder circuit

A SWAP gate : 

Use SWAP gates to move distant qubits closer ☺.



Mapped circuit with 1 extra SWAP

Challenge: Noise Minimization in NISQ processors

NISQ processors : Noisy Intermediate Scale Quantum processors

- Reducing noise is key to practical quantum computing in NISQ era.
- Noise can depend on depth, number of gates and error rates of qubits.
- We focus on minimizing the number of additional SWAP gates.

Our goal is to synthesize layout mappings with provably optimal SWAP counts.

Existing Work

Heuristic approaches

- Search based algorithms in major compilers like IBM qiskit and TKET.
- Several other approaches based on A*, temporal planning, and constraint programming etc.

Main Bottlenecks

- Extremely fast but adds many SWAPs (far from optimal).
- Not suitable for deeper circuits.

Existing Work

Exact Approaches, mainly based on SMT

- SWAP optimal : QMAP [2019]
- Depth optimal : OLSQ [2020] and OLSQ2 [2023]
- Near-optimal : TB-OLSQ [2020] and TB-OLSQ2 [2023]

Main Bottlenecks

- Time and memory bottlenecks when optimality is guaranteed.
- Near-optimal approaches, while more scalable, can still add many SWAPs.

Optimal Layout Synthesis for Quantum Circuits as Classical Planning

Irfansha Shaik, Jaco van de Pol (ICCAD-23)

- Two main actions: Schedule a CNOT or SWAP two qubits.
- Goal : Schedule all CNOTs on adjacent physical qubits respecting dependencies.
- An optimal plan corresponds to a mapping with optimal number of extra SWAPs.

Q-Synth v1

Optimal Layout Synthesis for Quantum Circuits as Classical Planning

Irfansha Shaik, Jaco van de Pol (ICCAD-23)

- Two main actions: Schedule a CNOT or SWAP two qubits.
- Goal : Schedule all CNOTs on adjacent physical qubits respecting dependencies.
- An optimal plan corresponds to a mapping with optimal number of extra SWAPs.

Looks like Graph search?

- Nodes define which gates are already scheduled.
- Edges define scheduling either a CNOT or a SWAP gate.
- Source node: no gates scheduled.
- Target node: all gates scheduled.

Q-Synth v1

Overview

- Significantly outperformed leading exact tools QMAP and OLSQ (at that time).
- Used off-the-shelf planners, FastDownward and Madagascar, for optimal synthesis.
- Outstanding domain submission award IPC-2023.

Q-Synth v1

Overview

- Significantly outperformed leading exact tools QMAP and OLSQ (at that time).
- Used off-the-shelf planners, FastDownward and Madagascar, for optimal synthesis.
- Outstanding domain submission award IPC-2023.

Challenges

- Optimal plan length is #CNOTs + #SWAPs.
- In our example, optimal plan length is $10 + 1 = 11$.
- In practical benchmarks, #CNOTs can be in 100s whereas #SWAPs can reach 20.
- Optimal planners suffer scalability issues for problems with long makespan.

Improvement in Q-Synth v2

Optimal Layout Synthesis for Deep Quantum Circuits on NISQ Processors with 100+ Qubits

Irfansha Shaik, Jaco van de Pol (SAT-24, best student paper award runner-up)

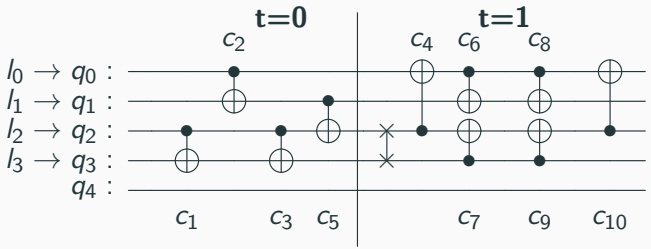
We encode optimal layout synthesis as SAT addressing the scalability bottleneck.

- Parallel plans to reduce makespan.
- Two-way CNOT constraints for better dependency propagation.
- Incremental solving to reuse learned clauses in various makespan runs.

Parallel plans

Logical to physical qubit mapping does not change between two consecutive swaps.

- In the time step 0, a group of CNOTs (≥ 0) can be scheduled.
- From time step 1, exactly 1 SWAP is scheduled followed by a group of CNOTs.



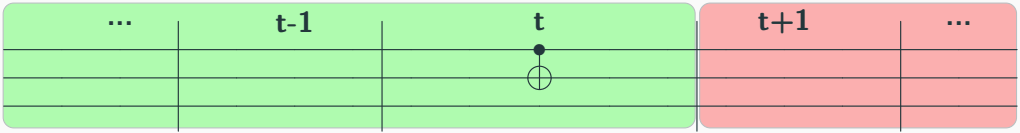
Our example circuit with 2 parallel steps instead of 11 sequential steps.

The order of CNOTs in a parallel step can be reconstructed from the original circuit.

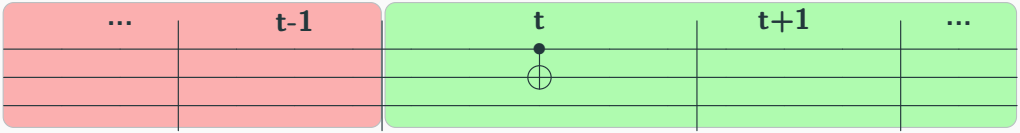
Two-Way CNOT constraints

CNOT dependencies must be respected when scheduling CNOT gates in parallel steps.

For all CNOTs, every predecessor is scheduled in the same or earlier steps but not later.



For all CNOTs, every successor is scheduled in the same or later steps but not earlier.



We encode using advanced, current, and delayed CNOT variables at every time step.
Encoding size scales linearly in #CNOTs while avoiding long clauses.

Incremental SAT Solving

A k-SWAP optimal synthesis requires refuting of mappings with up to $k-1$ SWAPs.

- We use incremental SAT solving to reuse the learned clauses refuting earlier steps.
- In each iteration, we assume that no CNOTs are delayed in the last time step.
- If no mapping found, we increase the makespan by 1 and update assumptions.

Experimental setup

Tools for Comparison
TB-OLSQ2: Near-Optimal
SABRE: Heuristic

Benchmarks
23 Standard (upto 54q/270cnots)
10 VQE (8q/upto 79cnots) **New!**

Platforms
54-qubit Sycamore
80-qubit Rigetti
127-qubit Eagle

Experiment 1

- 23 Standard benchmarks mapped to Sycamore, Rigetti, and Eagle platforms.
- Compared the number of additional SWAPs by tools TB-OLSQ2 and SABRE.

Experiment 2

- 10 VQE benchmarks mapped to Sycamore, Rigetti, and Eagle platforms.
- Compared #SWAPs and circuit depth with the tool TB-OLSQ2.

For both experiments, we give 12000s time and 8 GB memory limit.

Results: Experiment 1 (Standard)

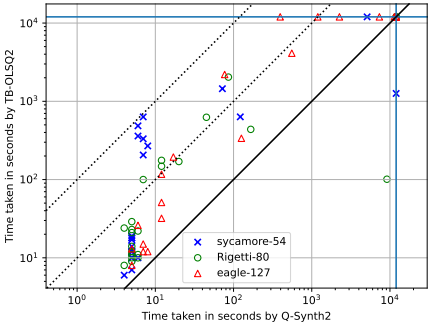
- Q-Synth2 outperforms TB-OLSQ2 (up to 100x)
- Optimally mapped circuits (up to 16-qubits) on to 54, 80, 127-qubit platforms for the first time.
- SABRE, while fast, adds too many SWAPs

Summary:

Tool:	**QS2**	TO2	SB
total solved (69)	61	56	69
optimally solved (69)	61	47*	10*

#SWAPs on 127-qubit Eagle Platform:

Circuit(q/cx) / Tool:	**QS2**	TO2	SB
queko(16/87)	4	timeout	36



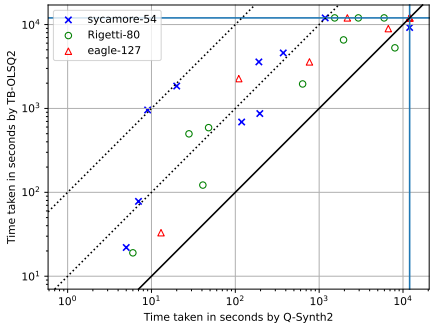
Results: Experiment 2 (VQE)

- Q-Synth2 outperforms TB-OLSQ2 (upto 100x)
- Optimally mapped 8-qubit VQE circuits with up to 17 SWAPs for the first time.

Summary:

Tool:	**QS2**	TO2
total solved (30)	24	20
optimally solved (30)	24	9*

Q-Synth2 sometimes also reports better depths.



Layout Synthesis Extensions in Q-Synth v4

Subarchitectures (BSc project, Kostiantyn V. Milkevych)

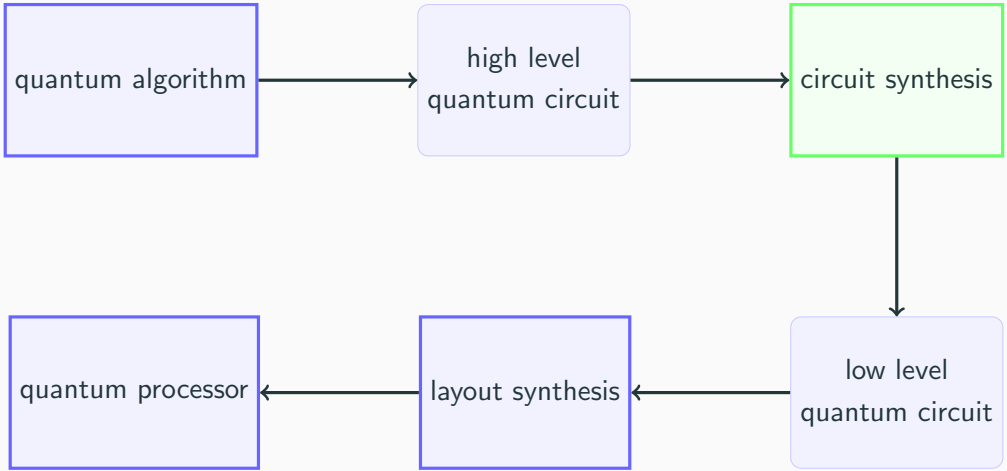
- Mapping only on maximal subarchitectures.
- Scales better on larger Quantum platforms.

Depth Optimal Synthesis (MSc project, Anders B. Clausen, Anna B. Jakobsen)

- Adopted SAT encoding for depth optimization.
- Significantly outperformed existing approaches.

Quantum Circuit Synthesis

Quantum Compilation Pipeline



Challenges

- Synthesis of an n-qubit circuit requires synthesis on $2^n \times 2^n$ complex matrix.
- Not practical even for a couple of qubits.
- Challenges in error still persist.

Peephole optimization: an alternative

- Synthesize/optimize subcircuits which can be efficiently represented.
- Focus on gate count and depth of error prone CNOT gates.

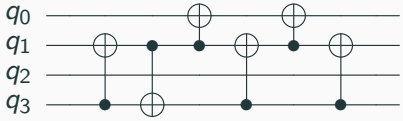
CNOT Circuits

CNOT gate

A Conditional-NOT gate which XORs two qubits.

CNOT circuit

A circuit with only CNOT gates.



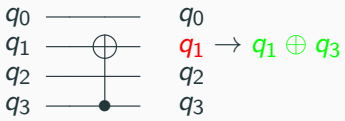
An Example CNOT Circuit

We start with an empty circuit and an empty parity matrix

$$\begin{array}{c} \begin{array}{cccc} q_0 & q_1 & q_2 & q_3 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

An Example CNOT Circuit

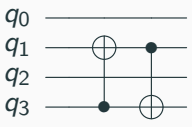
Applying CNOT on q_3 and q_1



$$\begin{array}{c} \begin{array}{cccc} q_0 & q_1 & q_2 & q_3 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

An Example CNOT Circuit

Applying CNOT on q_1 and q_3

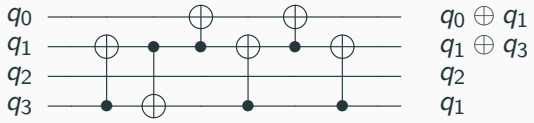


$$q_0$$
$$q_1 \oplus q_3$$
$$q_2$$
$$q_3 \rightarrow q_1 \oplus q_3 \oplus q_3 \rightarrow q_1$$

$$\begin{array}{c} \begin{array}{cccc} q_0 & q_1 & q_2 & q_3 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

An Example CNOT Circuit

Applying all CNOTs

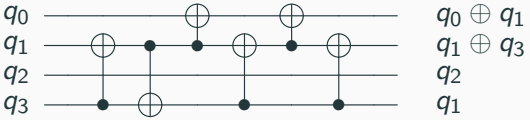


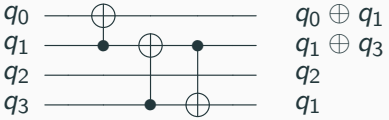
$$\begin{matrix} q_0 & q_1 & q_2 & q_3 \end{matrix}$$
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Optimal CNOT synthesis

- Synthesizing CNOT circuit corresponds to synthesizing parity matrix.
- Nodes are defined by parity matrices
- Edges are defined by CNOT gates
- Source node: Identity matrix
- Target node: Parity matrix of initial circuit

Equivalent Example Circuits





Optimal Layout-Aware CNOT Circuit Synthesis with Qubit Permutation

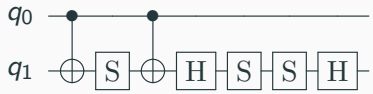
Irfansha Shaik, Jaco van de Pol (ECAI-24)

We encode optimal CNOT synthesis as Planning, SAT and QBF.

- Both CNOT count and Depth can be optimized.
- Handled layout restrictions and allows qubit permutation.
- Reduction of up to 56% in CNOTs and 46% in circuit depth.
- Reduction of up to 17% CNOTs, 19% CNOT depth in optimally mapped circuits.

Clifford Synthesis

Any Circuit with H, S, and CNOT gates is a Clifford circuit.



$$\left(\begin{array}{cc|cc|c} 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{array}\right)$$

- Nodes represent stabilizer matrices.
- Edges represent one of H, S, and CNOT gates.
- Synthesis is similar to CNOT synthesis.

CNOT-Optimal Clifford Synthesis as SAT

Irfansha Shaik, Jaco van de Pol

- Optimization of only CNOT gates instead of all gates.
- Handled layout restrictions and allows qubit permutation.
- Significantly outperformed existing exact approaches.
- Better circuits compared to industrial compiler like Qiskit and TKET.

Final thoughts

Impact

- Small steps towards filling the hole left by heuristics approaches.
- Being used at Kvantify for circuit optimization pipeline.
- Recently enabled VQE computation on IonQ's Aria platform.

Outlook

- Circuit synthesis of other interesting sub-classes.
- Optimizations on algorithm level.
- On going (MSc and BSc) student projects.

Thank You

Q-Synth: A Quantum Synthesizer

https://github.com/irfansha/Q-Synth