Towards **automated quantum circuit optimization** with graph-based **deep reinforcement learning**

Abhishek Abhishek

5th International Workshop on Quantum Compilation

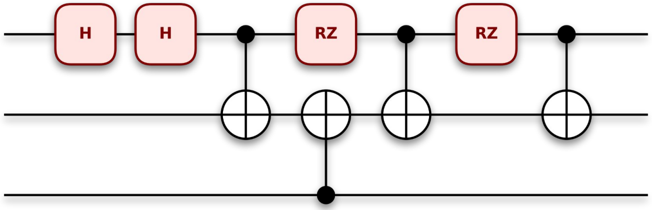
July 23, 2023

>QSAR QUANTUM SOFTWARE & ALGORITHMS RESEARCH

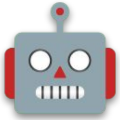
THE UNIVERSITY OF BRITISH COLUMBIA
Electrical and Computer Engineering
Faculty of Applied Science

Quantum Circuit Optimization with RL

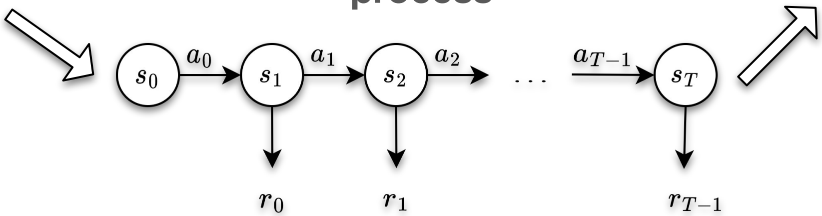
original circuit



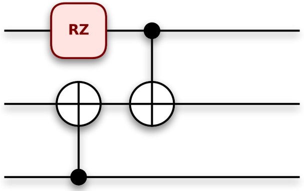
Reinforcement learning (RL) agent



Markov decision process



optimized equivalent circuit



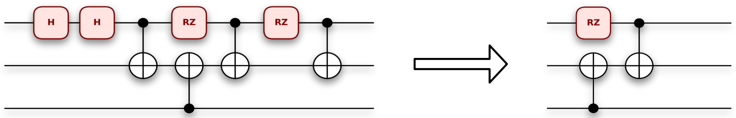
Quantum Circuit Optimization

goal: obtain a **more efficient representation** and reduce

- circuit depth
- total no. of gates
- T-gate count (fault tolerant)
- CNOT-gate count (near term)

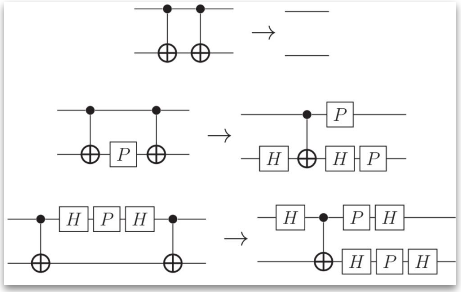
original circuit

optimized equivalent circuit



global optimization of arbitrary quantum circuits is difficult

peephole optimizations



transform passes, circuit matching etc.

phase polynomials

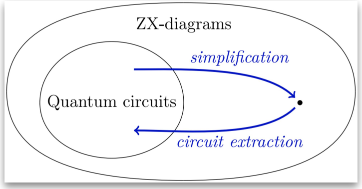
$$|\boldsymbol{x}\rangle \mapsto e^{2\pi i p(\boldsymbol{x})}|g(\boldsymbol{x})\rangle$$

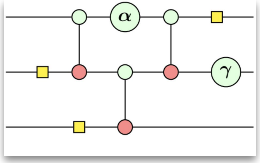
$$p(\boldsymbol{x}) = \sum_{i=1}^{2^n} \theta_i \; f_i(\boldsymbol{x})$$

$$g : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n \qquad f_i : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$$

{NOT, CNOT, Rz} circuits

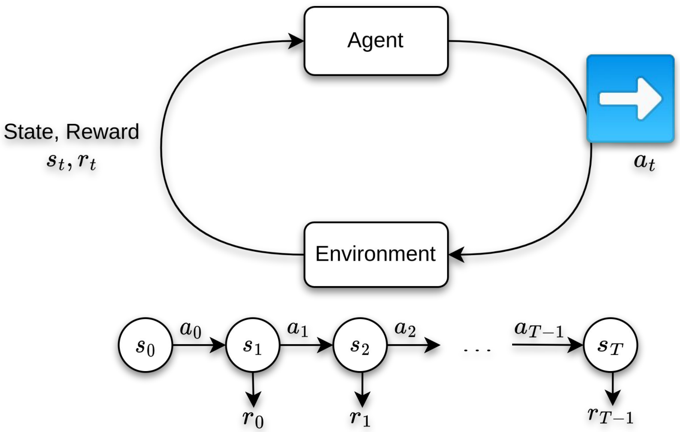
ZX-calculus





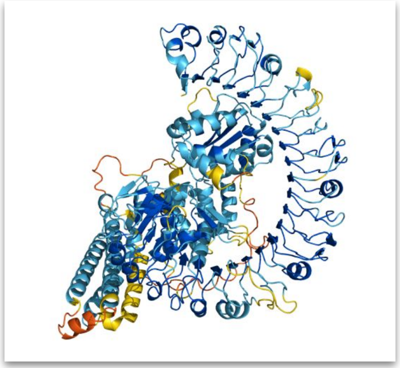
Reinforcement Learning

RL goal: autonomously discover strategies for **complex decision-making problems**



Chess, Shogi, and Go [1]
e.g. AlphaGo

Protein Folding [2]
e.g. AlphaFold



RL agents achieve superhuman performance in a lot of these tasks!

Compilation [3]

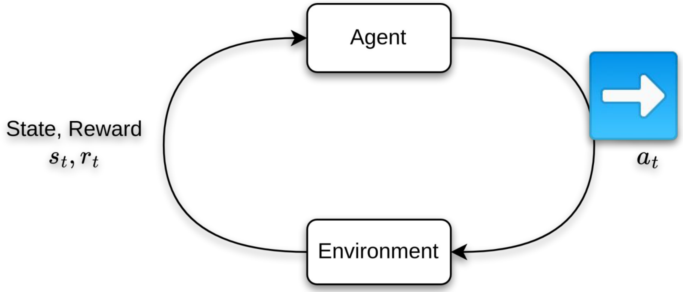
CompilerGym

[1] David Silver et al. ,A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science 362, 1140-1144 (2018).
[2] Jumper, J. et al. Highly accurate protein structure prediction with AlphaFold. Nature 596, 583–589 (2021).
[3] Cummins, Chris, et al. "Compilergym: Robust, performant compiler optimization environments for ai research." 2022 IEEE/ACM International Symposium on Code Generation and Optimization (CGO). IEEE, 2022.

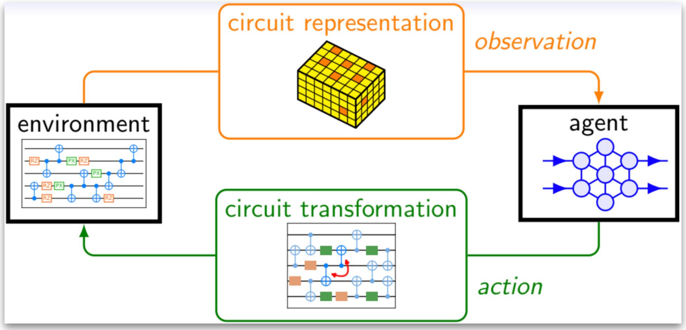
RL for Quantum Compilation

RL being explored for various quantum compilation tasks such as:

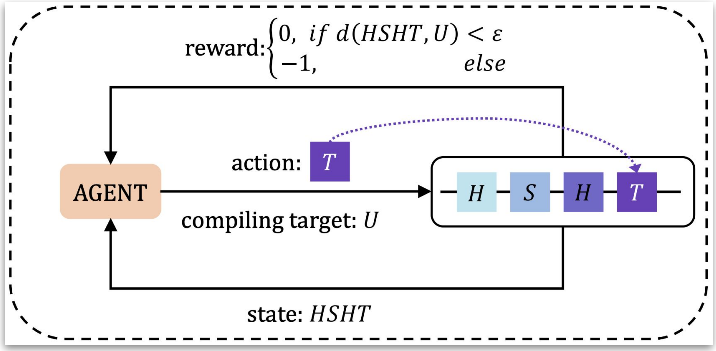
- circuit optimization
- unitary synthesis
- qubit placement and routing



circuit optimization [1]



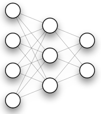
unitary synthesis [2]



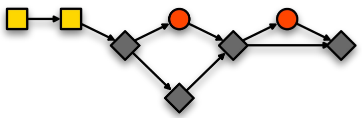
[1] T. F¨osel, M. Y. Niu, F. Marquardt, and L. Li, Quantum circuit optimization with deep reinforcement learning, arXiv preprint arXiv:2103.07585 (2021).
[2] Chen et al., Efficient and practical quantum compiler towards multi-qubit systems with deep reinforcement learning, arXiV: 2204.06904

Our Framework: Graph-based RL for QCO

graph neural network (GNN) agent

s_t  $\pi_\theta(.\,|s_t)$ policy network

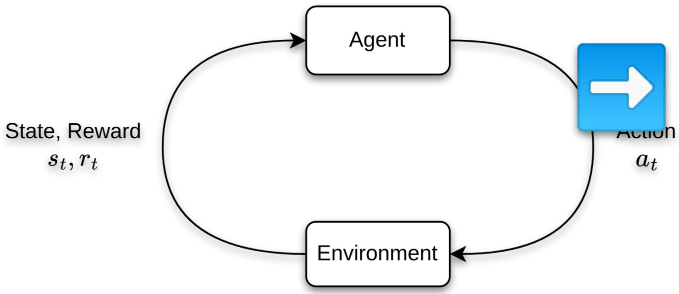
state s_t DAG representation

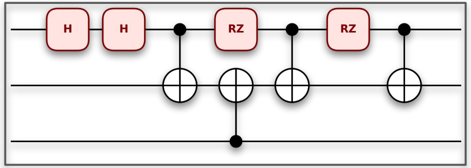


reward r_t

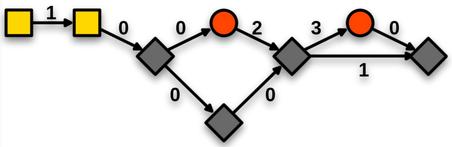
- reduction in gate count
- reduction in circuit depth

model-free RL -> we can use any optimization objective without major changes to the framework





$a_t \sim \pi_\theta(.\,|s_t)$




```
TRANSFORM_MAP = {
 0: "do_nothing",
 1: cancel_inverses,
 2: commute_controlled,
 3: merge_rotations
}
```

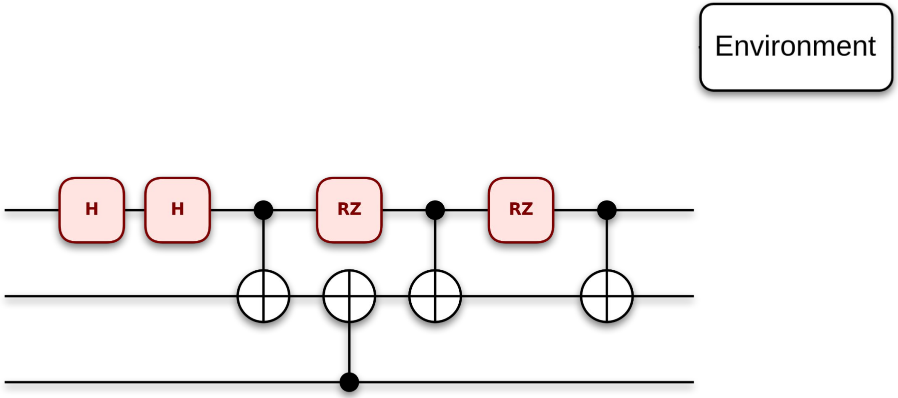
Environment

properties of the environment:

👉 state s_t : quantum circuit at a given step t

👉 fully observable

👉 deterministic transitions $s_{t+1} = f(s_t, a_t)$



in our current framework 🚧👷,

✅ gate set = {H, S, CNOT}, T, Rz and Rx -> can be replaced with any universal gate set

✅ one circuit processed at a time

Circuit DAG representation

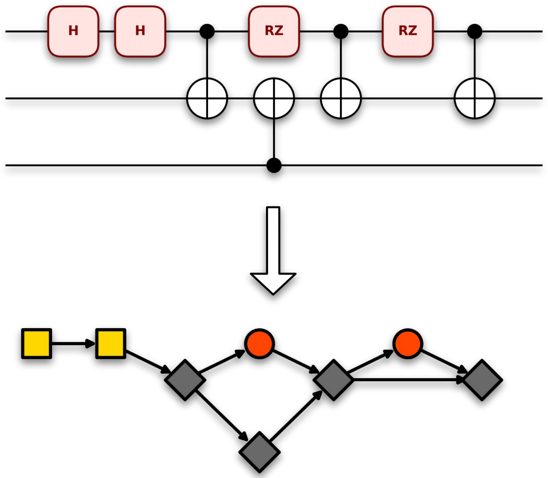
State, Reward
s_t, r_t

$U \Longleftrightarrow G = (V, E)$

vertices: gate operations $V = \{H, H, CNOT, RX, \dots\}$

edges: qubit dependencies among the gates

$(v_i, v_j) \in E \Rightarrow v_j$ acts on a qubit in sequence after v_i

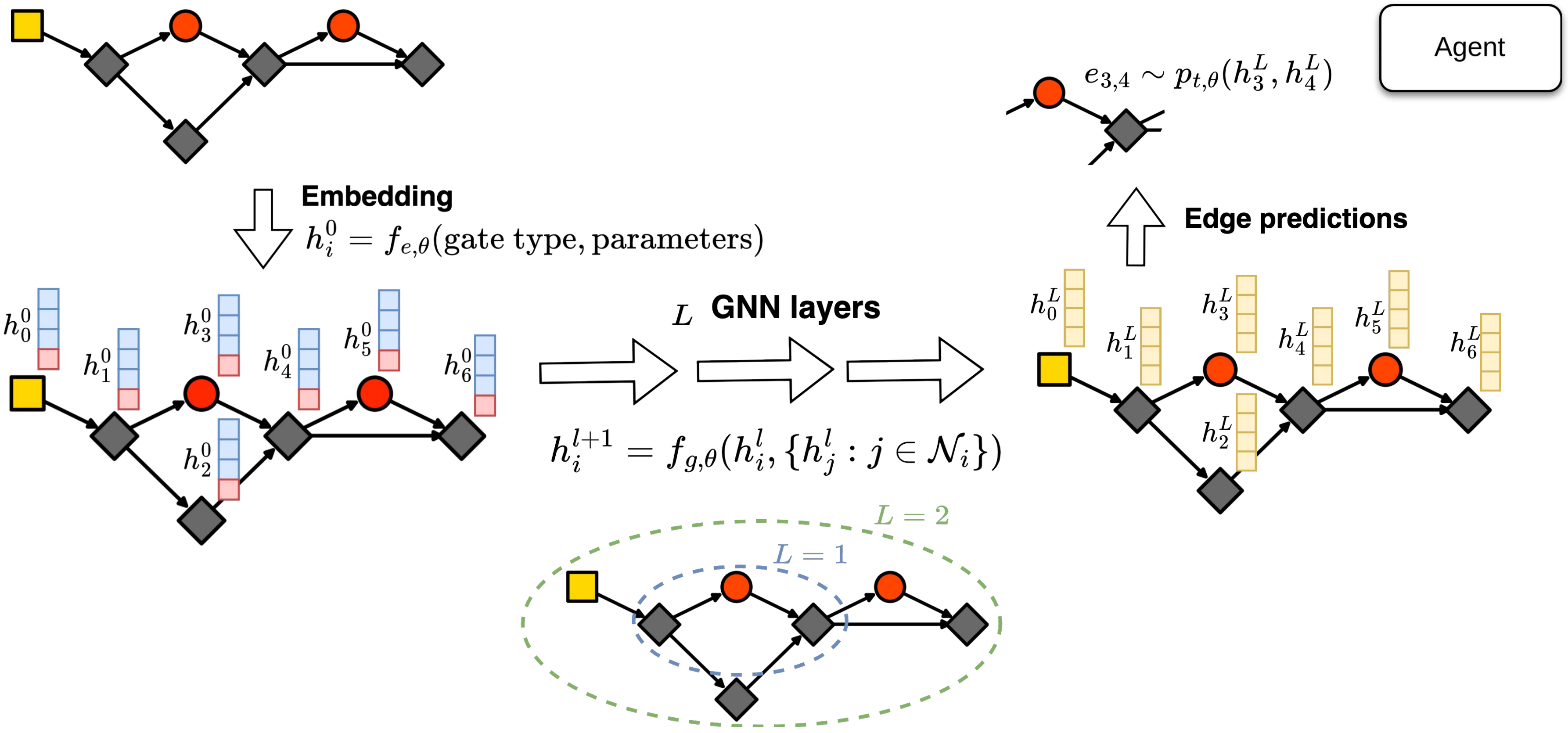


reward r_t as DAG properties

👍 circuit depth = length of the longest path in the DAG

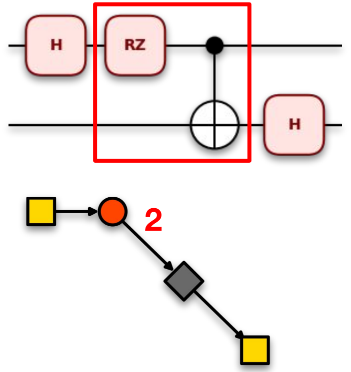
👍 gate count = no. of vertices $|V|$

Graph Neural Network (GNN) RL agent

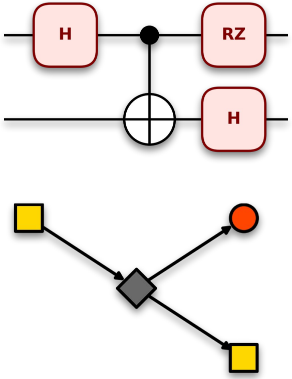


Edge representation of circuit transformations

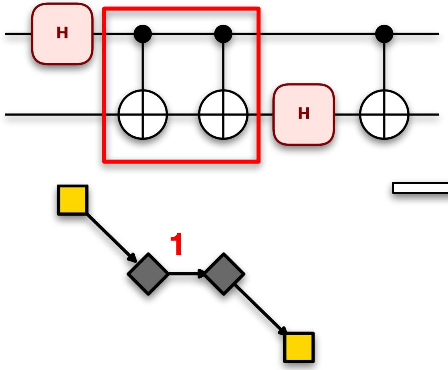
Action a_t



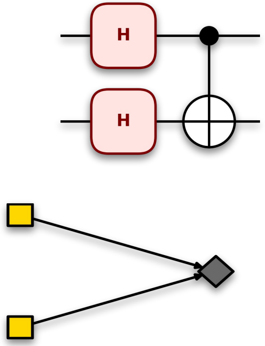
`commute_controlled(edge, graph)`



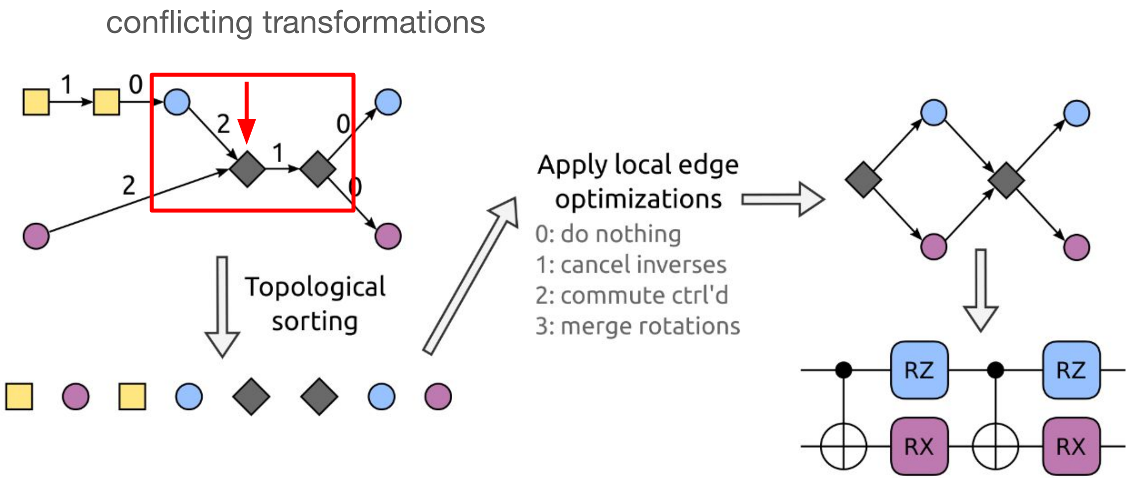

```
TRANSFORM_MAP = {
 0: "do_nothing",
 1: cancel_inverses,
 2: commute_controlled,
 3: merge_rotations
}
```



`cancel_inverses(edge, graph)`



Applying edge transformations



key idea: at any step, only consider the edge to the right of a node

Work-in-progress software framework 👷🚧🏗️

PENNYLANE

quantum circuit libraries, circuit unitary verification

PyG

GNN agent

RACCOON: Reinforcement **A**gents for quantum **C**ir**C**uit **O**ptimizati**ON**

NetworkX
Network Analysis in Python

circuit DAG transformation

Gymnasium

RL environment

Ongoing and Future work

- **Benchmarking** on different circuit libraries
  - Fault-tolerant: Reversible circuits, Hamiltonian simulation
  - Near-term: Variational circuits (e.g. QAOA, VQE)
- **Open-sourcing** the RL4QCO framework
- **Extending** graph-based RL to other compilation tasks such as **circuit cutting**

Collaborators

David Wierichs

Nathan Killoran

Olivia Di Matteo

Funding

NSERC CRSNG