Hardware-informed Reinforcement Learning for Quantum Gate Scheduling

Jakub Pietrzak

Supervisors: Sebastian Feld, Akash Kundu Responsible professor: Matthijs Spaan

TU Delft

1. Quantum gate scheduling

A quantum program consists of **a sequence of gates** to be **executed**. They should be scheduled in such a way to **minimize error probability**. In practice, a proxy metric of the **makespan** of the schedule is used - *shorter = better*

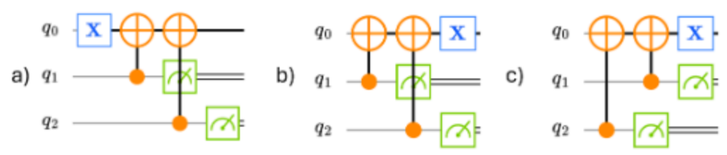


Figure 1. A scheduled circuit. We assume q_2 has shorter decoherence time than q_1 → it is less reliable. a) **makespan** = 4; b) Commutation-awareness → **makespan** = 3, but q_2 has an idle gap, a decoherence risk; c) Noise-awareness → move the idle gap to q_1, which is more reliable

But makespan ignores **decoherence times** and **crosstalk**, which influence the error probability.

2. Reinforcement Learning

Reinforcement Learning (RL) trains an agent to make sequential decisions by interacting with an environment:

State → Observation → Action → Reward → Next State

The agent learns the **policy**: $\pi(a|s)$, which expresses the probability of selecting action a in state s. The objective is to **maximize** cumulative **reward** over time:

$$\mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T(\tau)-1}\gamma^t r_t\right].$$

Notation: r_t is the reward received at timestep t. $\gamma \in [0,1]$ is the discount factor. $T(\tau)$ is the final timestep of the trajectory τ. $\mathbb{E}_\pi[\cdot]$ denotes the expected value over trajectories generated by following policy π.

RL Component	Scheduling environment
State	Current partial schedule, ready gates, qubit availability, dependencies, and hardware data
Observation space	Standard or extended
Action	Schedule a gate, or advance cycle
Reward	Standard or noise-aware
Policy	Learned strategy for choosing scheduling decisions from observed states

Table 1. RL formulation of quantum gate scheduling.

3. Motivation

- **Hardcoded heuristics** make non-RL solutions less adaptable
- Schedulers are optimized for the **proxy metric** instead of the closer objective, the **log-Estimated Success Probability (log-ESP)**
- **Hardware data** is often omitted: realistic gate execution times, crosstalk errors, qubit fidelities
- **Commutation rules** are often overlooked

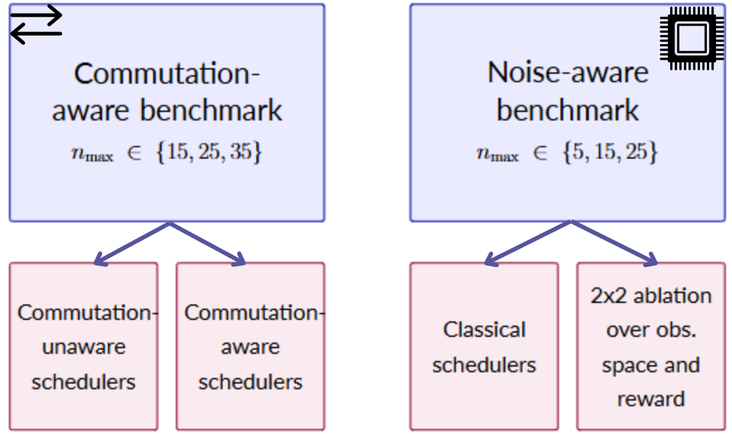
Research question: Does injecting domain knowledge into the scheduling agent increase the schedule fidelity? Can an RL agent produce better schedules than classical schedulers?

$$\mathcal{L}(S) = \sum_{g\in\mathcal{G}} \log F_i + \sum_{(i,j)\in\mathcal{X}(S)} \log C_{i,j} + \sum_{q\in\mathcal{Q}} \sum_{\Delta t\in\mathcal{I}_q(S)} \log F_{\text{idle}}(q,\Delta t)$$

log-ESP — gate fidelities — crosstalk — decoherence (idle gaps)

4. Methodology

We conduct 2 experiments with 3 max-gate settings ($n_{\max}$):



For classical schedulers we use *ASAP, ALAP*. For RL, we use *MaskablePPO*.

RL Feature	Variant
Observation Space	$\mathcal{O}_{\text{std}}$ (all relevant *noise-agnostic* state info) $\mathcal{O}_{\text{ext}} = \mathcal{O}_{\text{std}}$ + hardware noise data
Reward Function	$r_{\text{std},t} = -1$ per cycle $r_{\text{noise},t} = \begin{cases} \frac{\mathcal{L}(\hat{S}_{111})}{n}, & \text{if } n > 0, \\ -1, & \text{if } n = 0. \end{cases}$ for a partial schedule of n gates

Table 2. RL environment features we do ablation over to find the best way to inject noise-awareness

5. Results & Discussion

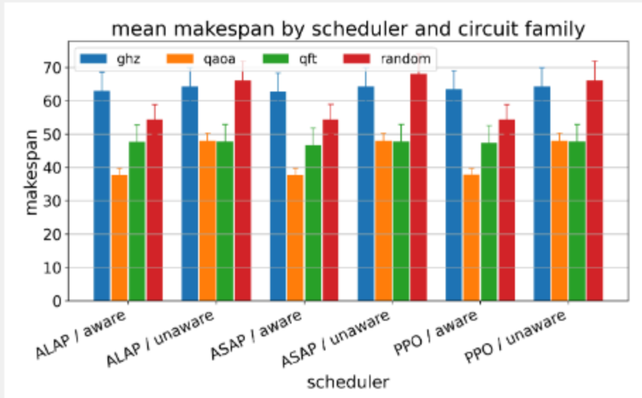


Figure 2. Commutation-aware benchmark; mean makespan by scheduler and circuit family

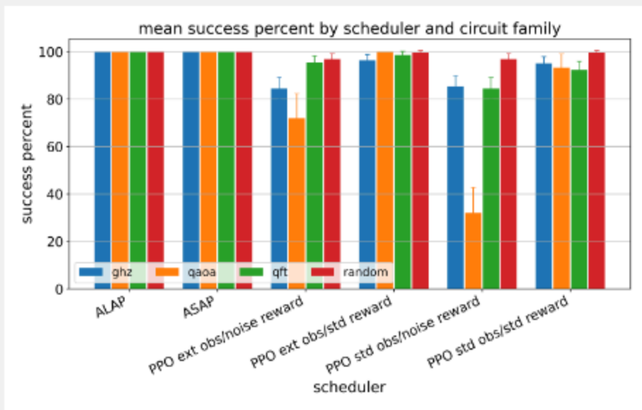


Figure 3. Noise-aware benchmark; success rate by scheduler and circuit family

- **Commutation-awareness reduces** makespan by 20% for **QAOA** and **Random** circuits, while giving little benefit for GHZ and QFT circuits.
- **PPO** is **able to compete** with classical schedulers both in makespan and log-ESP. **Extended observation space** looks especially **promising**, but more research is needed.
- **Noise-aware reward** is **unreliable**, produces much **smaller** number of **valid schedules**.

j.k.pietrzak@student.tudelft.nl