Observation and Action Encodings for Reinforcement Learning–Based Qubit Routing

A Controlled Ablation Study in qgym

Andac Durmaz[1]

Supervisor(s): S. Feld[1]**, A. Kundu**[1]**, M. Spaan**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: Andac Durmaz
Final project course: CSE3000 Research Project
Thesis committee: S. Feld, A. Kundu, M. Spaan, A. Lukina

Abstract

Running a quantum circuit on hardware with limited qubit connectivity requires inserting SWAP gates, each adding depth and exposing qubits to decoherence, so that two qubits that must interact become physically adjacent on the chip. Reinforcement learning (RL) is an increasingly used, adaptive alternative to hand-engineered routing heuristics, but how an RL agent's observation and action encodings should be designed has received little attention, even though environments such as qgym leave both choices to the user. We study these two encodings one at a time. The first is the observation reach, or how many upcoming gates the agent can see; the second is the action-space granularity, ranging from single SWAPs through a heuristically pruned set to multi-SWAP macro actions. Holding the reward, algorithm, hyper-parameters, and hardware fixed, we evaluate both on the coupling graphs of a 7-qubit and a 16-qubit IBM device. The observation reach has little effect on routing, and the full-circuit view fails on the larger device. Action-space granularity matters much more: macro actions solve substantially more circuits and route them with fewer SWAPs, and this advantage grows with device size. Injecting the same routing heuristic as a *soft* prior on the action distribution, rather than as a hard mask, preserves completeness while roughly halving the SWAP overhead. A disjoint held-out evaluation suggests these results reflect transferable routing skill rather than memorized training circuits.

1 Introduction

In the current Noisy Intermediate-Scale Quantum (NISQ) era, quantum hardware is limited by physical constraints such as restricted qubit connectivity and gate error rates [8]. On the superconducting devices targeted in this work, a fixed *coupling graph* defines which physical qubits are adjacent, and a two-qubit gate can act only on an adjacent pair.

When a quantum algorithm is written, it is expressed in terms of *virtual* qubits assuming all-to-all connectivity, where a two-qubit gate may be applied to any pair of qubits directly. Real hardware, however, offers no such freedom: the program must first be compiled to respect the connectivity of the target device. This compilation pipeline involves three main steps: *mapping* virtual qubits to physical ones, *routing* the computation so that interacting qubits are brought next to each other on the chip, and *scheduling* gate operations.

This work focuses on the *routing* step. When two qubits need to interact but are not physically adjacent, the compiler inserts SWAP gates to move them together (Figure 1). Each SWAP makes the circuit deeper and exposes the involved qubits to more decoherence. Routing must insert SWAPs until every two-qubit gate of the circuit can be executed on the device's coupling graph (formalized in Section 2). Among the several objectives studied in the literature, such as gate fidelity, runtime, or crosstalk, we adopt the common choice of minimizing the total number of inserted SWAPs, which in turn limits the added circuit depth. Even under this single objective the problem is NP-complete [13], with a search space that grows rapidly in circuit size.

Classical compilers tackle this problem with hand-engineered heuristics such as SABRE [5] and the routing layer of t|ket⟩ [1]. These methods are fast and produce strong baselines, but the rules they encode are fixed and adapt poorly when the hardware topology, gate set, or noise profile changes. Reinforcement learning (RL) has been proposed as a more adaptive alternative [7; 9; 6]: an agent interacts with the routing environment step by step and receives reward signals that gradually shape its policy toward better routing decisions. For qubit routing, each agent action corresponds to inserting a SWAP (or skipping when none is needed), and the reward reflects progress toward executing the remaining gates.

While prior RL formulations show that learned policies can match, and in some cases improve on, classical heuristics in specific settings [7], substantial freedom remains in *how* the interaction between agent and environment is designed. The qgym framework [17] consolidates much of this design space into a configurable routing environment, but allows the user to design the *observation* and *action* encodings. These two design choices, the observation space (what the agent sees) and the action space (what the agent is allowed to do), strongly shape how easily a policy can be learned, how sample-efficient training is, and how well the resulting policy generalizes to unseen circuits.

Research questions. This work investigates these encodings under tightly controlled conditions. The main research question is: *how do the observation-space and action-space encodings of an RL agent affect routing performance, learning stability, sample efficiency, and generalization in the qubit-routing pass?* We decompose it into two controlled ablation studies, each varying a single interface design choice while holding the rest of the experimental setup fixed:

- **SQ1 (observation).** How does the observation encoding, that is, how many upcoming gates of the circuit the agent can see (its lookahead *reach*), affect routing performance?
- **SQ2 (action).** How does the action-space granularity — atomic single-SWAP actions versus a heuristically pruned top-k action set versus high-level macro (multi-SWAP) actions — affect routing performance?

Within SQ2 we also explore a softer way to apply this heuristic: a *prior* that biases the policy toward good SWAPs rather than a hard mask that forbids the rest.

Contributions. The contributions of this paper are:

- A controlled ablation harness, built on qgym, that isolates the observation reach (SQ1) and the action-space granularity (SQ2) while holding the reward function, hyper-parameters, and hardware fixed across arms (using PPO for SQ1 and MaskablePPO for SQ2).
- An evaluation across two IBM coupling graphs of different scale (the 16-qubit Guadalupe and the 7-qubit

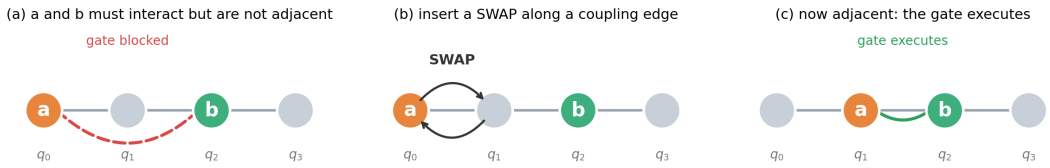

Figure 1: Qubit routing. A two-qubit gate needs qubits a and b, but they sit on non-adjacent physical qubits (a); a `SWAP` along a coupling edge brings them together (b), after which the gate executes (c).

Casablanca) to probe how the encoding effects interact with device size.

- A generalization protocol that separates memorization of training circuits from transferable routing skill, by training on a mixture of randomly generated and structured interaction circuits and evaluating on a *disjoint*, held-out benchmark suite.
- A soft-prior alternative to hard action masking that injects the routing heuristic as a bias on the action distribution rather than forbidding `SWAP`s, preserving completeness while reducing `SWAP` overhead.

2 Background and Related Work

2.1 The routing problem

A quantum circuit is compiled against a device whose connectivity is a graph $G = (V, E)$, where V is the set of physical qubits and an edge $(u, v) \in E$ means a two-qubit gate may be applied directly to qubits u and v. Following the qgym routing formulation [17], we abstract a circuit over a set of *virtual qubits* V_{virt} into its *interaction circuit*: the ordered list of two-qubit gate pairs $\langle (q_1, q_2), (q_3, q_4), \dots \rangle$. Single-qubit gates impose no connectivity constraint and are ignored in the routing process. A mapping $\pi : V_{\text{virt}} \to V$ assigns each virtual qubit to a physical one, and a two-qubit gate on virtual qubit pair (a, b) is executable only if $(\pi(a), \pi(b)) \in E$. When it is not, the agent inserts `SWAP` gates along edges of G until the pair becomes adjacent. The objective is to make every gate executable while inserting as few `SWAP`s as possible. The initial mapping is reset to the identity each episode (virtual qubit i on physical qubit i), so each interaction circuit defines a fully determined routing instance. Framed as a sequential decision process, this is a Markov decision process (MDP) [14]: the state is the current mapping together with the remaining interaction circuit, an action inserts a `SWAP` on an edge of G or *surpasses* the next gate, transitions are deterministic, and a per-step reward (defined below) charges for `SWAP`s while crediting executed gates.

2.2 The qgym routing environment

All experiments use the qgym `Routing` environment [17], a Gymnasium-compatible [16] simulator in which an episode routes one interaction circuit on a fixed coupling graph until all gates are executed. The environment exposes two encodings as configuration choices, which are exactly the variables we study (Section 3): the *observation space* and the *action space*.

Observation space. The default observation comprises the current mapping, a lookahead window over upcoming interaction gates, and a legality mask over candidate actions. Its key tunable parameter is `max_observation_reach`, the number of upcoming gates exposed to the agent: a short reach shows only the next few gates, a wide reach shows more of the circuit at the cost of a larger observation. The optional adjacency-matrix observation is disabled, since on a fixed topology it is constant and carries no learnable signal.

That observation does not expose the full MDP state: the mapping is complete, but only the next `max_observation_reach` gates are shown. Even so, it can be a *Markov state* (a sufficient statistic for optimal action) [14]: the mapping already summarizes every `SWAP` so far, so a memoryless policy can in principle act optimally, and a bounded reach withholds only part of the *future* circuit. The global reach is the limiting case, recovering the full MDP state. SQ1 therefore reduces to a sharp question: *how far ahead must the agent see to route well?*

Action space. In the atomic encoding, an action either performs a `SWAP` on one edge of G or issues the *surpass* action that executes the next gate (defined above). Alternative encodings either prune the SWAPs to the k that most reduce the total distance between qubits still waiting to interact, or aggregate several `SWAP`s into a single high-level move; these are formalized as the SQ2 arms in Section 3.

2.3 Reward function

To keep reward design from confounding the encoding study, all experiments use qgym's built-in `SwapQualityRewarder`. The per-step reward sums four event-based components: -5 for an illegal action, -1 per `SWAP`, $+1$ when a previously blocked two-qubit gate becomes executable (*surpass*), and $+0.8$ for a `SWAP` that makes more upcoming gates immediately executable (*good `SWAP`*). An action is illegal only when the agent surpasses a gate whose qubits are not yet adjacent; the SQ2 action masks (Section 3) make this unreachable, so the penalty is operative only for the unmasked SQ1 agent. The agent thus receives dense feedback every step rather than a single terminal signal, mitigating the sparse-reward problem typical of qubit routing [7].

2.4 Reinforcement-learning algorithm

The base algorithm is Proximal Policy Optimization (PPO) [12], used through Stable-Baselines3 [11]. SQ1 uses standard PPO; SQ2 uses `MaskablePPO` [2] because its variant-specific action spaces need an explicit feasibility mask. PPO is a stable, widely used default for Gymnasium

environments and has been applied to quantum-circuit compilation [9; 6]. Core hyper-parameters are matched across both sub-questions; only the encoding under study varies. The full table is in Section 4.

2.5 Related work

Classical routing heuristics. The qubit mapping and routing problem has been studied extensively from a compiler-engineering perspective. SABRE [5] introduced a bidirectional look-ahead heuristic (it routes the circuit forward and then in reverse, using each pass to refine the initial mapping for the next), now widely used as a baseline and shipped in several production compilers. The routing layer of t|ket⟩ [1] provides a similar hand-engineered solver. Beyond fast look-ahead heuristics, the problem has also been approached with informed graph search such as A^* [21], and formulated as an exact optimization problem, where constraint- and SMT-based solvers return a provably `SWAP`-minimal mapping [20; 15]; these exact methods, however, scale only to small circuits. SABRE and t|ket⟩ are fast and effective, but their static rules must be re-tuned by hand for each new device or noise model, motivating learning-based approaches.

Reinforcement learning for routing and compilation. Several works cast quantum-circuit compilation as a sequential decision-making problem. Pozzi et al. [7] apply RL directly to qubit routing, showing a learned policy can rival classical heuristics; but the observation and action encodings are fixed throughout, so the reported performance gap between the learned policy and classical baselines reflects a particular interface choice as much as the learning method itself. Quetschlich et al. [9] use RL to choose and order compiler-optimization passes, and Oancea et al. [6] apply deep RL to selecting good initial mappings under fixed gate-error rates. Van Veen et al. [18] study action-space engineering for RL-based routing in distributed quantum systems, showing that exposing multi-`SWAP` "macro" actions has a large effect on what the agent learns. Our SQ2 builds on this but isolates the granularity axis within a single, controlled environment.

Benchmarking environments and circuits. The qgym framework [17] provides a configurable Gymnasium environment for RL-based quantum compilation. It exposes the observation lookahead reach as a tunable parameter and supplies a default atomic action space; finer action-space control requires user-built wrappers, which is the design space this paper probes. For circuits, MQT Bench [10] provides a large, categorized collection of parameterized circuits, from which we draw disjoint slices for the structured training pool and the held-out evaluation suite.

Positioning. Unlike prior RL-routing studies, we treat the encodings themselves as the object of a controlled ablation, holding the reward, algorithm, hyper-parameters, hardware, and circuits fixed and varying only the observation reach (SQ1) or the action granularity (SQ2).

3 Methodology

The research question is answered through two controlled ablation studies. Each varies a single interface choice while holding the reward (Section 2), the RL algorithm, the policy architecture (a feedforward `MultiInput` network), the hyper-parameters, the hardware, and the circuits fixed, so any difference in routing performance is attributable to the encoding under study rather than to a confounder. We refer to each trained configuration (one observation reach for SQ1, one action space for SQ2) as an *arm*. This feedforward policy is deliberately simple, with no structural inductive bias toward the circuit or coupling graph; whether a structure-aware alternative such as an attention- or graph-based policy would change the picture is a question we return to in future work.

3.1 SQ1 — observation reach

SQ1 varies only `max_observation_reach`, the number of upcoming two-qubit gates the agent can see at each step. We sweep seven reaches, from a single upcoming gate to one that covers the whole circuit:

- **reach-1, reach-2, reach-3** — narrow windows that expose only the next one to three gates;
- **local** — a reach of 5 gates: a compact, reactive view of what is coming next;
- **reach-10 and reach-20** — intermediate reaches that progressively widen the window;
- **global** — a full-circuit reach, set to the longest circuit the agent can see during training (80 gates on Casablanca, 261 on Guadalupe), so it sees every remaining gate at once.

These values are set relative to the circuits: the structured training circuits have a median of 9 (Casablanca) and 15 (Guadalupe) two-qubit gates, so a reach of 1–5 exposes only part of a typical circuit, 10–20 spans most of one, and global reaches even the longest. SQ1 uses standard PPO. Because a wider reach produces longer, slower episodes, all SQ1 arms are trained for the same number of *episodes* (via the `StopTrainingOnMaxEpisodes` callback), rather than the same number of steps, so a slower-learning arm is not cut short by a step-based budget.

3.2 SQ2 — action-space granularity

SQ2 compares three action encodings, all trained with `MaskablePPO`:

- **atomic** — every individual `SWAP` on the coupling graph is its own action, plus the *surpass* action that executes the next gate. This is the most expressive but largest action space.
- **heuristic (top-k)** — at each step the agent is restricted to the k `SWAP`s that most reduce the total distance between qubits still needing to interact. We sweep $k \in \{1, 2, 3, 5, 7\}$ ($k = 3$ is the default arm), interpolating between a nearly greedy action set and the full atomic one.
- **macro** — following Van Veen et al. [18], a single action executes a pre-computed shortest-path chain of `SWAP`s that brings a chosen interacting pair together, so one action stands in for a whole sequence of individual `SWAP`s. Our action mask restricts this choice to the frontier's first

unresolved gate, so the agent decides only whether to surpass that gate or route its two qubits together (and along which path), keeping the action space compact and the routing target unambiguous.

Because the three encodings induce different episode lengths per decision, SQ2 arms are trained under a fixed training-step budget; the resulting differences in episode counts are reported and revisited in the discussion.

A soft heuristic prior. Motivated by a weakness of the heuristic encoding (hard masking can forbid a distance-increasing `SWAP` that a route requires), we also study a soft alternative. Instead of masking, the policy samples from

$$\pi(a \mid s) = \mathrm{softmax}\big(f_\theta(s)_a + \beta \cdot \mathrm{score}(a)\big),$$

where $f_\theta(s)_a$ is the network's logit for action a, $\mathrm{score}(a)$ is the same normalized distance-reduction the heuristic uses, and β sets the prior's strength. Every action keeps non-zero probability, so the policy can still choose a distance-increasing `SWAP` when a route needs one. It is trained with plain PPO (no mask), and $\beta = 0$ recovers the unbiased atomic policy. We sweep $\beta \in \{0, 1, 2, 4\}$ on the atomic action space.

3.3 Hardware: two device scales

To test whether the encoding effects depend on device size, we run every arm of both sub-questions on two IBM coupling graphs: the 16-qubit Guadalupe and the 7-qubit Casablanca (Figure 2). Both are simulated backends from Qiskit's fake-backend interface [3], reproducing the connectivity of the corresponding real IBM devices. Casablanca holds circuits of at most seven qubits, so a full circuit fills almost the entire chip and leaves few spare qubits; routing is then harder than on the 16-qubit device, where empty qubits give the agent room to move qubits around.

3.4 Training distributions and the generalization protocol

A learned router is only useful if it generalizes beyond its training circuits. To separate memorization of training circuits from a transferable policy, every agent is trained on a **mixed** distribution: at each episode it draws, with fixed probability, either a freshly sampled random interaction circuit or a circuit from a structured pool. The random half blocks memorization, since the agent rarely sees the same circuit twice, while the structured half exposes it to realistic algorithmic interaction patterns. Each arm is then evaluated two ways: *in-sample*, on the very circuits in its training pool (a per-instance ceiling), and *held-out*, on a disjoint suite of MQT Bench [10] circuit families not seen in training; the gap between the two separates memorized performance from transferable skill. Because the held-out circuits are also substantially *larger* than the training pool, this measures transfer to novel, harder circuits of similar families, not domain transfer to a different distribution or topology, so a high held-out rate reflects genuine routing skill. The protocol layers onto the same SQ1 and SQ2 arms, so both can be read in- and out-of-sample.

4 Experiments

4.1 Hardware and software

Experiments run in the qgym `Routing` environment [17] on top of Qiskit's [3] fake-backend interface, using Stable-Baselines3 [11] for PPO and `MaskablePPO`. Two coupling graphs are used: IBM Guadalupe (16 qubits) and IBM Casablanca (7 qubits). Training is CPU-only. On top of the `Routing` environment we add three wrappers: a custom *info-stripping* wrapper, then Gymnasium's `TimeLimit` (the per-episode step cap) and `Monitor` (per-episode logging). The info-stripping wrapper is a performance fix: qgym's per-step `info` dictionary holds the full, growing list of inserted `SWAP`s, which Stable-Baselines3 deep-copies every step, making long episodes progressively slower; since the agent never reads `info`, we replace it with an empty dictionary.

4.2 Circuits

Two circuit sources are used:

- **Random interaction circuits** — used by the mixed training distribution, with a fresh circuit drawn at the start of every episode. Each circuit has a uniformly random number of two-qubit gates (between 1 and 80), where every gate connects a uniformly random pair of distinct qubits on the device.
- **MQT Bench** [10] — generated at MQT Bench's *target-independent* level. At this level each circuit is expressed using only single- and two-qubit gates (any larger gate, such as a Toffoli, has been decomposed into these smaller ones), but the circuit is not yet mapped to any device's connectivity, so the routing problem is still open. These circuits are split into two disjoint folders, a structured training pool and a held-out evaluation suite spanning random, real-algorithm, and synthetic families, with no circuit appearing in both.

4.3 Classical baseline

We compare the RL agents against SABRE [5], the classical heuristic shipped with Qiskit. A circuit is *solved* when the router makes every one of its two-qubit gates executable within the evaluation step budget; the *solve rate* is the fraction of the evaluation suite that is solved. SABRE is a complete router that solves every circuit, while an RL agent may get stuck and fail some circuits. Comparing average `SWAP` counts over different circuit sets would therefore be misleading, so we compare `SWAP` counts only on circuits that both SABRE and the RL agent solve.

We use SABRE-trivial: the identity initial mapping (virtual qubit i on physical qubit i, the same starting point the RL agents use; Section 2) followed by SabreSwap, so any difference reflects routing effort alone. We do not compare against SABRE-full, which first runs SabreLayout to optimize the initial mapping. The RL agents never optimize layout, so that comparison would conflate routing quality with layout choice. SABRE is stochastic, so each circuit is routed under five transpiler seeds and we report the *minimum* `SWAP` count, matching the best-of-five protocol used for the RL agents.

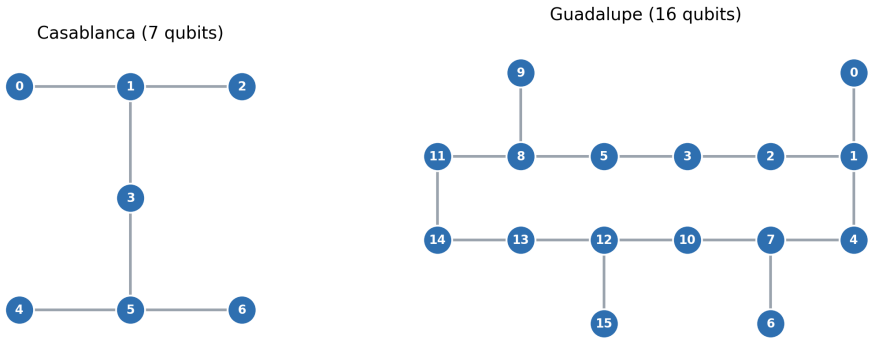

Figure 2: Coupling graphs of the two IBM devices, obtained from Qiskit's fake-backend interface: nodes are physical qubits and edges are the qubit pairs that may interact directly. Casablanca (left, 7 qubits) is the Falcon layout; Guadalupe (right, 16 qubits) is a heavy-hex lattice with longer routing distances.

4.4 Episode bounds and evaluation cap

Training episodes are truncated by the Gymnasium `TimeLimit` wrapper, after 600 steps on Casablanca and 1,500 on the larger Guadalupe. The caps are scaled to device size: Guadalupe has more qubits and a larger graph diameter, so routing a circuit there requires more `SWAP`s, and its cap is correspondingly higher. Both caps sit well above what a competent policy needs to route a typical training circuit (median 9 and 15 two-qubit gates on Casablanca and Guadalupe respectively), so the cap mainly terminates episodes in which the policy is stuck rather than cutting short genuine progress. Evaluation uses a fixed 3,000-step cap, applied identically across sub-questions. For most circuits this far exceeds what a competent policy needs, so a failure indicates a stuck policy. It is, however, a *fixed* budget: six of the 39 held-out Guadalupe circuits (the `grover` and `qwalk` families at 10–16 qubits) contain between 3,666 and 234,300 two-qubit gates, more than the cap permits *any* router to clear, since each gate requires at least one step. The maximum achievable Guadalupe solve rate is therefore $33/39 = 84.6\%$, exactly the score macro attains; we verified separately that, with an evaluation cap scaled to circuit length, macro routes those long circuits to completion. A reported Guadalupe failure thus means *either* a stuck policy *or* a circuit longer than the fixed budget, and macro solves every circuit the budget admits.

4.5 Training budget and seeds

SQ1 arms are trained for an equal number of episodes per arm (`StopTrainingOnMaxEpisodes`); SQ2 arms are trained for a fixed number of environment steps. Casablanca configurations are repeated across ten random seeds (1–10); Guadalupe, which is far costlier per seed (a single global-reach arm trains for roughly fifteen hours), is repeated across three. Because the learning-stability metric is the standard deviation of solve rate across seeds, we prioritise seed count on the device where it is affordable. Shared hyper-parameters are listed in Table 1.

We did not tune hyper-parameters per arm. The configuration follows the PPO example shipped with qgym [17]:

Setting	Value
Algorithm (SQ1 / SQ2)	PPO / MaskablePPO
Policy	MultiInput
Discount γ	0.99 (SB3 default)
Learning rate	$3 \cdot 10^{-4}$ (SB3 default)
Batch size	256 (default 64)
n_{steps}	2048 (SB3 default)
Epochs per update	10 (SB3 default)
Entropy coefficient	0.01 (default 0)
PPO clip range	0.2 (SB3 default)
GAE λ	0.95 (SB3 default)
Value-function coefficient	0.5 (SB3 default)
Max gradient norm	0.5 (SB3 default)
Parallel environments	6
Train episode cap (steps)	600 (Casa.) / 1,500 (Guad.)
Eval step cap	3,000
Seeds	1–10 (Casa.) / 1–3 (Guad.)

Table 1: Hyper-parameters shared across both sub-questions. Only the variant-specific encoding under study (observation reach for SQ1; action space for SQ2) differs between arms.

every value in Table 1 is the Stable-Baselines3 default except two: the minibatch is enlarged from 64 to 256 for steadier gradients, and a small entropy bonus (0.01) is added to encourage exploration. SQ1 uses PPO and SQ2 MaskablePPO, which share these settings (the action mask aside). Keeping every value fixed across arms is deliberate: per-arm tuning would reintroduce the confounder we are removing, so we report performance under one fixed configuration rather than peak performance. Because every arm shares the same untuned setup the *comparison* stays fair, and given the large effect sizes (Section 5) it is unlikely tuning would reverse the ranking.

Budgets and coverage. Because the two sub-questions use different training budgets (Section 3), their absolute numbers are read separately, not compared directly. The Guadalupe runs (seeds 1–3) serve as a scale check that the Casablanca findings hold at 16 qubits.

4.6 Metrics

Each trained agent is evaluated with stochastic, best-of-5 rollouts on the held-out suite: rather than taking the deterministic argmax, the policy is sampled five times per circuit, and a circuit counts as *solved*, as defined in Section 4.3, if at least one of the five rollouts routes every two-qubit gate before the step cap. The policy-quality metrics are: *solve rate*, the fraction of held-out circuits solved; average `SWAP`s on solved circuits; average episode length; and wall-clock training time. Two additional metrics characterize training dynamics. *Sample efficiency* is the number of *episodes* until the rolling 100-episode mean reward first reaches 90% of its initial-to-final improvement. *Learning stability* is the standard deviation of solve rate across seeds for each arm. Lower is better for both. Per-circuit results are persisted so individual circuits can be compared across variants, sub-questions, devices, and seeds.

Statistical analysis. With as few as three seeds (Guadalupe), a seed-level rank test is underpowered: at three seeds a two-sided Mann–Whitney test cannot fall below $p = 0.10$ regardless of effect size. We therefore use seeds for variance (the σ above) and test differences between encodings *per circuit* on the held-out suite, which provides 19 (Casablanca) and 39 (Guadalupe) paired observations: McNemar's exact test for solve/fail outcomes and the Wilcoxon signed-rank test for `SWAP` overhead on jointly-solved circuits, with Holm correction within each family. Bootstrap 95% confidence intervals on solve rate resample circuits.

5 Results

We report Casablanca (7 qubits) across ten seeds and Guadalupe (16 qubits) across three seeds. All numbers are held-out: each agent is evaluated with stochastic best-of-five rollouts on the disjoint test families (Section 3.4). `SWAP` overheads are reported relative to SABRE-trivial, the like-for-like baseline that starts from the same identity mapping as the RL agents (Section 4). Throughout, we treat `local` and `atomic` as the default configurations against which the other encodings are compared. Seeds report *variance* (the standard deviation σ of solve rate); differences between encodings are tested *per circuit* on the held-out suite (McNemar for solve/fail, Wilcoxon for `SWAP` overhead, Holm-corrected), as detailed in Section 4.

5.1 SQ1: observation reach

Widening the observation reach does not improve completeness. On Casablanca every reach solves essentially the same fraction, about 89.5%: the narrowest reach-1 solves one extra circuit (94.2% on average) and the global reach one fewer (88.9%), and these differences are not statistically significant (McNemar, one to two discordant circuits). So reach is judged on efficiency (Table 2), where two things stand out. First, learning is stable across seeds for the mid-range reaches ($\sigma = 0$ from reach-2 through reach-20); only the narrowest reach-1 and the full-circuit global reach vary (2.8 and 1.6 pp), reflecting the single circuit they inconsistently solve. Second, the narrowest reaches are the most `SWAP`-efficient ($1.6\times$ SABRE at reach 1–3 against local's $2.5\times$) but take many more episodes to converge, while the wider reaches converge faster yet cost more wall-clock time (Table 2). So a wider observation buys faster convergence, a narrower one buys routing efficiency; neither improves completeness. The global reach is strictly dominated: it matches the others on completeness at best, is among the least stable, and is the most expensive to train.

Reach	Solve%	σ (pp)	Sample-eff. (episodes)	SWAP× (median)	Steps	Train
Casablanca (10 seeds)						
reach-1	94.2	2.8	2582	1.6	372	17 m
reach-2	89.5	0.0	1302	1.6	139	22 m
reach-3	89.5	0.0	775	1.6	145	21 m
local (5)	89.5	0.0	746	2.5	173	24 m
reach-10	89.5	0.0	586	2.2	192	30 m
reach-20	89.5	0.0	557	1.6	193	31 m
global	88.9	1.6	571	2.9	224	55 m
Guadalupe (3 seeds; global seed 1)						
local (5)	46.2	0.0	–	9.7	452	223 m
reach-10	41.0	2.1	–	13.6	327	277 m
reach-20	41.0	2.1	–	15.0	483	255 m
global	0.0	–	–	–	–	902 m

Table 2: SQ1 observation reach. Completeness is statistically indistinguishable across reaches, so the comparison is one of efficiency. *Sample-eff.* is the number of episodes until the rolling 100-episode mean reward reaches 90% of its initial-to-final improvement; σ is the standard deviation of solve rate across seeds. SWAP× is the median overhead relative to SABRE-trivial on solved circuits (best-of-seeds).

On Guadalupe the picture is consistent and the cost of a wide observation becomes severe. Local, reach-10, and reach-20 cluster at 41–46% (local leads on both completeness and efficiency), while global collapses to 0% after roughly fifteen hours of training, unable to make use of an observation that spans 261 upcoming gates. The Guadalupe reach-1/2/3 arms are omitted: under the equal-*episode* SQ1 budget a single Guadalupe reach arm takes four to five hours, making the low-reach sweep there computationally infeasible. Taken together, the observation reach is not a productive design lever: a single upcoming gate already suffices, and widening the window never improves completeness and, on the larger device, only hurts efficiency.

5.2 SQ2: action-space granularity

Action-space granularity is the main factor (Table 3). The macro encoding solves every Casablanca circuit (100%) and 84.6% of Guadalupe's, above atomic (89.5% / 42.7%); the heuristic top-k arms fall between the two, rising with k up to k=5 (77.4% on Casablanca) before levelling off, and all remain below atomic and macro. On Guadalupe, macro's advantage over both atomic and local is statistically significant on the per-circuit tests, in completeness (McNemar, 14–15 discordant circuits, $p < 0.001$) and in `SWAP` overhead (Wilcoxon, $p \leq 0.004$). On Casablanca the completeness gap between macro and atomic is two circuits and is *not* significant ($p = 0.50$), but macro's `SWAP`-efficiency advantage is ($p = 0.04$). The extremes are also the most stable: atomic and macro have $\sigma = 0$ across seeds on both devices, while

Variant	Solve%	σ (pp)	Sample-eff. (episodes)	SWAP× (median)	Train
Casablanca (10 seeds)					
atomic	89.5	0.0	441	3.0	12 m
k=1	35.3	4.1	244	1.0	14 m
k=2	60.5	2.6	237	1.2	14 m
k=3	67.4	3.2	484	1.2	14 m
k=5	77.4	3.4	364	1.5	13 m
k=7	74.7	4.6	297	1.2	13 m
macro	100.0	0.0	1044	2.0	14 m
Guadalupe (3 seeds)					
atomic	42.7	3.2	614	24.0	49 m
k=1	5.1	0.0	220	1.0	44 m
k=2	15.4	0.0	189	1.4	43 m
k=3	23.1	3.6	451	2.0	43 m
k=5	22.2	1.2	433	3.2	45 m
k=7	26.5	2.4	608	4.1	46 m
macro	84.6	0.0	1264	1.9	45 m

Table 3: SQ2 action granularity (held-out). Macro maximizes completeness; the extremes (atomic, macro) are perfectly stable while the heuristic arms carry the seed variance. *Sample-eff.* is as in Table 2. SWAP× is the median overhead relative to SABRE-trivial on each arm's solved circuits, and should be read together with solve% and the matched-set numbers in Section 5.2.

the intermediate heuristic arms carry essentially all of the seed variance (σ up to 4.6 pp). Macro's completeness does carry a sample-efficiency cost: it is the slowest arm to converge (1,044 episodes on Casablanca against atomic's 441), echoing the SQ1 pattern that the most capable arms converge slowest. Training cost is flat across the SQ2 arms (they share a fixed step budget), so the less complete arms are not cheaper to produce, only less capable.

Because each arm solves a different set of circuits, a raw efficiency comparison would be confounded, so we also compare `SWAP` overhead on the *same* circuits, those solved by all of atomic, macro, and local (17 on each device). On this matched set macro is the most efficient on both devices, and the gap widens sharply with scale: on Casablanca macro routes them at 2.0× SABRE against atomic's 3.0× (local 2.5×), and on Guadalupe macro needs only 1.8× where atomic requires 21.4× (local 9.6×). Macro's advantage is thus not an artifact of which circuits it solves; it holds, and grows, when the circuit set is held fixed.

5.3 Interaction of observation reach and action granularity

SQ1 and SQ2 each vary one encoding at a time; to check that they do not interact, we re-ran all three action spaces at every observation reach on Casablanca. The two axes are essentially independent. Macro solves every circuit (100%) at every bounded reach and dips only at the full-circuit global view (96.5%); atomic is flat at 89.5% regardless of reach; and the heuristic not only stays below both but *degrades* as the window widens (73.7% at reach-1 down to 56.1% at global). The action-space ranking (macro > atomic > heuristic) therefore holds at every reach, and a wider observation never lifts any action space above its narrow-reach score. Widening the window is thus no substitute for a better action space: granularity dominates, and the only consistent effect of more observation is to make the pruned heuristic worse. Guadalupe shows the same pattern (macro flat at 84.6%, atomic 38–43%, heuristic 20–28% across reaches). The non-baseline reaches use three seeds (only reach-5 has ten), so these numbers are a directional check that the two axes do not interact—indicative, not definitive.

5.4 A soft heuristic prior versus hard masking

The heuristic top-k arms reveal a weakness of hard masking: by *forbidding* every `SWAP` outside the k most distance-reducing ones, they occasionally rule out a distance-increasing `SWAP` that a route requires, and so lose completeness (Table 3). A natural alternative is to keep the heuristic signal but apply it *softly*: instead of masking, we add a β-weighted bias toward the same distance-reducing `SWAP`s to the policy's logits, so every action keeps non-zero probability (Section 3; plain PPO). Sweeping β on Casablanca, this matches atomic's completeness (89.5% at every β, against the hard heuristic's 77.4% at best) while roughly halving the `SWAP` overhead: average `SWAP`s on solved circuits fall from 200 at β=0 (no bias) to 102 at β=2, saturating thereafter. On the jointly-solved circuits this reduction is significant (Wilcoxon, $p = 0.001$ versus atomic). The bias is what drives the gain, since β=0 reproduces atomic. On Guadalupe the effect points the same way (matched median 94 `SWAP`s for the soft prior versus 230 for atomic) but is not significant on the smaller matched set ($p = 0.08$). The soft prior improves routing efficiency, not training speed: like the narrow-reach arms in SQ1, it takes more episodes to converge. We treat this as a promising direction rather than a fully validated method, since it is evaluated on the atomic action space and primarily on the smaller device.

5.5 Generalization: in-sample versus held-out

On both devices, held-out solve rate is lower than in-sample, but the variant ranking is preserved and macro generalizes best (Table 4). On Casablanca macro transfers *without loss* (100% in- and out-of-sample), while atomic and the local/global reaches drop about ten points and the heuristic drops more; on Guadalupe the same ordering holds with larger drops. The drop reflects two distinct factors: genuine novelty (the test families are disjoint from training) and a difference in difficulty (the held-out circuits are larger than the training pool, a median of 113 versus 15 two-qubit gates on Guadalupe). The gap is therefore a conservative estimate of generalization, not a pure measure of memorization.

5.6 Effect of device scale

The Casablanca conclusions carry over to Guadalupe, but their magnitude grows sharply with device size. Macro's completeness lead over atomic widens from about ten percentage points at 7 qubits (not significant) to roughly forty at 16 ($p < 0.001$); its matched-set `SWAP`-efficiency lead grows from 2.0× versus 3.0× to 1.8× versus 21.4×; and the global reach degrades from a one-point deficit (88.9%) to total failure (0%) as its observation grows from 80 to 261 gates. Action-space granularity therefore matters *more* as the device scales up, not less. (These Guadalupe magnitudes rest

	Casablanca		Guadalupe	
Arm	In-sample	Held-out	In-sample	Held-out
macro	100%	100%	100%	85%
atomic	100%	90%	70%	46%
k=3	91%	67%	38%	21%
local	100%	90%	71%	46%
global	100%	89%	0%	0%

Table 4: In-sample (training pool) versus held-out solve rate. The ranking is preserved on both devices, and macro transfers best, losing nothing on Casablanca. Held-out is over all seeds (ten Casablanca, three Guadalupe); Casablanca in-sample is likewise over ten seeds, while Guadalupe in-sample is seed 1 only.

on three seeds, a scale check rather than tightly-estimated effects, though the per-circuit significance does not depend on seed count.)

6 Discussion

6.1 Observation reach (SQ1)

The central SQ1 finding is that enlarging the observation reach does not improve routing, and at the extreme it harms it. Because the SQ1 arms share an equal-episode budget, this cannot be blamed on wider reaches being under-trained: every arm sees the same number of training circuits, yet the wider ones gain nothing in completeness.

We can test the natural explanation, that the routing decision is largely local, directly with the reach sweep. Since the mapping already makes the process memoryless (Section 2), reach is purely *forward* visibility, so if the decision were local a narrow window should match a wide one. It does: a single-gate observation is no worse than any wider reach, and the heuristic even degrades as the window grows. We cannot, however, fully separate "the decision is local" from "a flat MLP cannot exploit the wider observation even where it would help." The contrast with SABRE is telling: SABRE plans over the *entire* remaining circuit (the full Markov state) and routes with fewer `SWAP`s than any of our arms, so that information is demonstrably useful; yet handing our agent the same full MDP state (the global reach) yields no gain and never closes the gap to SABRE. The bottleneck is therefore representational, not informational.

The global reach makes this concrete. Its observation grows with circuit length, so on Guadalupe a median training circuit fills only a few of the 261 slots: the input is mostly padding and the signal sparse and positionally shifting. The policy collapses to a degenerate `SWAP`-cycling regime that rarely surpasses (an entropy bonus and stochastic evaluation only partly mitigate it), giving 0%. A sequence- or graph-structured policy, a richer observation, or a longer-horizon reward could plausibly exploit the global reach where a flat MLP under a myopic reward cannot; we develop these in future work.

6.2 Action granularity (SQ2)

SQ2 shows the opposite. The action encoding, unlike the observation, matters a great deal. The atomic space requires the agent to assemble every multi-`SWAP` route from single steps, which is a long-horizon credit-assignment and exploration problem, whereas the macro space collapses each routing operation into a single action along a precomputed shortest path, shortening the effective decision horizon by an order of magnitude. We read macro's dominance as a consequence of this horizon reduction rather than of any extra information: it is given the same circuits and the same reward, only a coarser way to act. The heuristic top-k arms are revealing in the same way. Pruning the menu to the k distance-reducing swaps shrinks the action space but leaves the single-step structure, and so the long horizon, intact, while its distance filter can withhold the occasional distance-increasing swap a route requires. These arms therefore trade away completeness without easing the underlying difficulty, and approach atomic as k grows. This diagnosis is confirmed by the soft-prior variant (Section 5.4): biasing the policy toward the same distance-reducing `SWAP`s *without forbidding* the others recovers atomic's completeness while roughly halving its `SWAP` overhead, so it is the hard *forbidding* of actions, not the heuristic signal itself, that costs completeness. The fixed step budget has one further consequence: because a macro episode resolves a circuit in fewer decisions, the macro agent completes more episodes within the fixed budget. This is a property of the encoding rather than a confound, since the step budget equalizes environment interaction, which is what costs compute.

6.3 Generalization

That the variant ranking survives from in-sample to held-out, with macro dropping the least, confirms the encoding effects are genuine rather than memorized. Disentangling true generalization from the held-out circuits' larger size would, however, require a held-out suite matched to the training circuits' size.

6.4 Device scale

A single theme connects the two devices: the encoding effects are governed by problem dimensionality, so they intensify as the device grows. The action-side difficulty, assembling routes from single `SWAP`s, worsens with more qubits and longer routes, which is why macro's lead over atomic widens sharply at 16 qubits; the observation-side failure scales for the same reason, as the global reach's input triples in size and its performance falls from competitive to degenerate. The seven-qubit device is small enough to mask both weaknesses; the sixteen-qubit device exposes them, suggesting the conclusions would sharpen further on larger hardware, though confirming that is beyond our compute budget.

6.5 Threats to validity

Several limitations bound these conclusions. We sample each axis at finitely many points (seven reaches, five values of k), so we describe trends but cannot pinpoint the optimal reach or k. All experiments use one reward function: fixing it isolates the encoding effect, but a different reward could shift the balance between completeness and `SWAP` economy.

The two sub-questions use different training budgets, so their absolute numbers are read separately. Compute is CPU-only, so Guadalupe, at hours per arm, is limited to three seeds

(one for the global reach); its variance is a scale check, not a fully seeded study. Crucially, our headline comparisons do not rest on seed counts: they are tested per circuit (19 on Casablanca, 39 on Guadalupe), so adding seeds would only tighten the low-seed arms, a compute limit rather than a methodological one.

The initial mapping is always the identity, so the agents never optimize layout, and part of the gap to SABRE comes from this rather than routing quality. The observation is a fixed, flat encoding, and we did not test a structure-aware policy; the SQ1 null result therefore bounds what *this* representation extracts, not what a richer one might. Finally, each device has a single fixed topology, so our generalization is across circuit families and sizes, not across hardware.

7 Responsible Research

This work involves no human subjects or personal data, so the main responsibility considerations are reproducibility, honest evaluation, and transparent use of compute and third-party software.

Reproducibility. Every training run is seeded (ten seeds on Casablanca, three on Guadalupe), and the hyper-parameters are held identical across all arms. We deliberately do not tune them per variant, so that any difference in routing performance is attributable to the encoding under study rather than to tuning (Section 4, Table 1). The values are the Stable-Baselines3 defaults except for two documented choices, a larger batch size and a small entropy bonus. Each run writes its output to a file whose name encodes the full configuration (sub-question, variant, device, seed, and evaluation suite), so that no two runs can overwrite each other's results. The source code, the benchmark-generation scripts, and the per-circuit result files are released,[1] so the full pipeline of training, evaluation, and the figures in this paper can be reproduced.

Honest evaluation. A learned router can look strong simply by being tested on the circuits it was trained on. To avoid this, training and evaluation use *disjoint* circuit families from MQT Bench: no evaluation circuit, and no evaluation family, appears in the training pool (Section 3.4), so reported solve rates are measured on genuinely unseen circuits. Each policy is also sampled five times per circuit (stochastic, best-of-five) rather than run once deterministically; we state this explicitly so the numbers are not mistaken for single-shot performance.

Use of third-party assets. The experiments build on qgym [17], Stable-Baselines3 [11], Qiskit [3], and MQT Bench [10], each used within its open-source license and cited at first use. All results use a corrected version of qgym's `Routing` environment, which otherwise applies `SWAP` actions to non-adjacent physical qubits; the patch is included in our released code.

Broader impact. Reducing the number of `SWAP`s a circuit requires lowers its depth and the errors that accumulate with it, which can make near-term quantum computation more reliable. The work is foundational and methodological, and we see no direct dual-use or misuse concern.

[1] https://github.com/andacdurmaz2/rl-for-quantum-circuit

Compute footprint. All training is CPU-only, and we report the wall-clock training time of every arm alongside the results so the cost of the study is transparent. The most expensive single arm is the full-circuit (global) observation on Guadalupe, at roughly 15 hours. We flag this cost precisely because it yields a degenerate policy, which makes it a cautionary data point rather than a useful configuration.

AI usage. During this research, an LLM model was used as a tool to correct grammatical errors in the paper and paraphrase long or unclear sections. While writing the code and debugging the compilation errors for the experiment, AI assistance was also utilized.

8 Conclusions and Future Work

8.1 Conclusions

We studied how an RL router's observation and action encodings affect its routing, training, and generalization, through two controlled ablations on qgym: observation reach (SQ1) and action granularity (SQ2). Both run on the coupling graphs of a 7- and a 16-qubit IBM device, simulated via Qiskit's fake-backend interface rather than physical hardware. The two encodings behave very differently. Widening the observation reach does not help: a single upcoming gate already suffices, and the full-circuit view fails on the larger device. This is not because the extra information is useless; SABRE exploits the whole circuit to route better than any of our agents. Rather, a flat policy under a myopic reward cannot use it, so the bottleneck is representational, not informational. Action granularity matters far more: macro solves many more circuits than atomic or heuristic and routes them with fewer `SWAP`s, both effects growing with device size. The variant ranking holds in- and out-of-sample, so these are properties of the encodings, not memorized circuits.

In practice, effort spent on the action space is better rewarded than effort spent widening the observation, and *how* heuristic knowledge is injected matters: a soft prior preserves completeness while markedly cutting `SWAP` overhead, whereas a hard mask over the same heuristic loses completeness. More broadly, the paper offers a controlled, encoding-focused comparison of RL-based routing and a held-out protocol that separates transferable routing skill from memorization.

8.2 Future Work

Several directions follow. The clearest, from our SQ1 result, is *how* the agent processes the observation: because a flat MLP could not exploit even the global reach that SABRE uses to advantage, a structure-aware policy (attention over the upcoming gates [19] or a graph neural network over the device's coupling graph and the circuit's interaction graph [4]), paired with a longer-horizon reward, is the natural way to turn a wider observation into better routing. On the action side, the soft heuristic prior is a first step beyond hard masking (Section 5.4); confirming it on larger devices and pairing it with lookahead-aware candidate heuristics is open. Further extensions follow from our limitations: a fuller joint sweep of the two axes (Section 5.3), more device topologies (which would

re-activate the adjacency observation that is constant here), and an equalized cross-sub-question budget.

References

[1] Alexander Cowtan, Silas Dilkes, Ross Duncan, Alexandre Krajenbrink, Will Simmons, and Seyon Sivarajah. On the qubit routing problem. In *14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019)*, volume 135 of *LIPIcs*, pages 5:1–5:32, 2019.

[2] Shengyi Huang and Santiago Ontañón. A closer look at invalid action masking in policy gradient algorithms. In *The International FLAIRS Conference Proceedings*, volume 35, 2022.

[3] Ali Javadi-Abhari, Matthew Treinish, Kevin Krsulich, Christopher J. Wood, Jake Lishman, Julien Gacon, Simon Martiel, Paul D. Nation, Lev S. Bishop, Andrew W. Cross, Blake R. Johnson, and Jay M. Gambetta. Quantum computing with Qiskit. arXiv:2405.08810 [quant-ph], 2024.

[4] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[5] Gushu Li, Yufei Ding, and Yuan Xie. Tackling the qubit mapping problem for NISQ-era quantum devices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '19)*, pages 1001–1014, 2019.

[6] Rares Adrian Oancea, Stan van der Linde, Willem de Kok, Matthia Sabatelli, and Sebastian Feld. Optimizing initial qubit mappings under fixed gate error rates using deep reinforcement learning. In *Innovations for Community Services (I4CS 2025)*, volume 2513 of *Communications in Computer and Information Science*, pages 189–208. Springer, 2025.

[7] Matteo G. Pozzi, Steven J. Herbert, Akash Sengupta, and Robert D. Mullins. Using reinforcement learning to perform qubit routing in quantum compilers. *ACM Transactions on Quantum Computing*, 3(2):1–25, 2022.

[8] John Preskill. Quantum computing in the NISQ era and beyond. Quantum 2, 79 (2018); arXiv:1801.00862 [quant-ph], 2018.

[9] Nils Quetschlich, Lukas Burgholzer, and Robert Wille. Compiler optimization for quantum computing using reinforcement learning. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2023.

[10] Nils Quetschlich, Lukas Burgholzer, and Robert Wille. MQT Bench: Benchmarking software and design automation tools for quantum computing. *Quantum*, 7:1062, 2023.

[11] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv:1707.06347 [cs.LG], 2017.

[13] Marcos Yukio Siraichi, Vinícius Fernandes dos Santos, Caroline Collange, and Fernando Magno Quintão Pereira. Qubit allocation. In *Proceedings of the 2018 International Symposium on Code Generation and Optimization (CGO)*, pages 113–125. ACM, 2018.

[14] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2018.

[15] Bochen Tan and Jason Cong. Optimal layout synthesis for quantum computing. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 137:1–137:9. IEEE, 2020.

[16] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. arXiv:2407.17032 [cs.LG], 2024.

[17] Stan van der Linde, Willem de Kok, Tariq Bontekoe, and Sebastian Feld. qgym: A Gym for training and benchmarking RL-based quantum compilation. arXiv:2308.02536 [quant-ph], 2023.

[18] Joost van Veen, Luise Prielinger, and Sebastian Feld. Rethinking how to act: Action-space engineering for reinforcement learning-based circuit routing in distributed quantum systems. arXiv:2605.02389 [quant-ph], 2026.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[20] Robert Wille, Lukas Burgholzer, and Alwin Zulehner. Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations. In *Proceedings of the 56th Annual Design Automation Conference (DAC)*, page 142. ACM, 2019.

[21] Alwin Zulehner, Alexandru Paler, and Robert Wille. An efficient methodology for mapping quantum circuits to the IBM QX architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(7):1226–1236, 2019.