Reinforcement Learning for Quantum Circuit Optimization: A Review

Akash Kundu[1,2,3], **Aritra Sarkar**[1,2,3], **Prayag Tiwari**[4], **Sebastian Feld**[1,2]

[1]Delft University of Technology, Delft, The Netherlands
[2]Quantum Computing Division, QuTech, The Netherlands
[3]Quantum Intelligence Alliance, Kolkata, India
[4]Halmstad University, Sweden

Quantum circuit performance on noisy intermediate-scale and early fault-tolerant devices is constrained by circuit depth, two qubit gate overhead, hardware connectivity and noise. Reinforcement learning (RL) offers a framework for automating circuit design and optimization. It treats circuit construction, compilation, routing and rewriting as sequential decision-making problems. In this review, we synthesize progress in reinforcement learning-based quantum circuit optimization (RL-QCO). We cover variational ansatz search, unitary synthesis, qubit mapping, ZX-calculus simplification and fault-tolerant resource reduction. We organize the field around four Markov decision process design choices. These are state representations, reward functions, action spaces and learning agents. This perspective reveals recurring trade-offs. These include expressivity versus trainability, sparse versus shaped rewards, gate-level versus abstract actions, and general purpose versus hardware-aware policies. We also assess benchmarking practices. Key issues include reproducibility, baselines, hyperparameter sensitivity, generalization across instances and hardware-in-the-loop evaluation. Finally, we outline emerging directions. These include reusable circuit priors, amortized reward evaluation, continual hardware adaptation and integration with fault-tolerant compilation. RL-QCO is positioned as a developing design paradigm for quantum software stacks, rather than a single algorithmic recipe.

Contents

Corresponding author: A.Kundu@tudelft.nl
Aritra.Sarkar@quantum-intelligence.net
Prayag.Tiwari@hh.se
S.Feld@tudelft.nl

1 Introduction

Quantum circuit design and optimization has emerged as a central bottleneck for realizing practical quantum algorithms on noisy intermediate-scale quantum (NISQ) devices [1, 2, 3]. For a fixed logical algorithm, performance on real hardware is dominated by low-level circuit properties such as depth, two-qubit gate count, connectivity overhead, and sensitivity to coherent and incoherent noise, rather than by asymptotic complexity alone [4, 5, 6, 7]. Current platforms typically offer only a modest number of qubits with restricted coupling graphs and non-negligible error rates, so even moderate-depth circuits can fall outside the regime where meaningful fidelities are achievable [8, 9, 10, 11]. As a result, a large fraction of the effort in deploying algorithms for chemistry, optimization, and simulation is spent on designing and refining parameterized quantum circuits (PQCs) and their compilations to specific architectures, motivating systematic methods that explicitly target circuit quality under realistic hardware and noise models [12, 13, 14, 15, 16, 17, 18, 19]. These methods span variational ansatz design, unitary compilation, qubit mapping and routing, circuit simplification, and early fault-tolerant resource optimization, all of which can be framed as quantum circuit optimization problems.

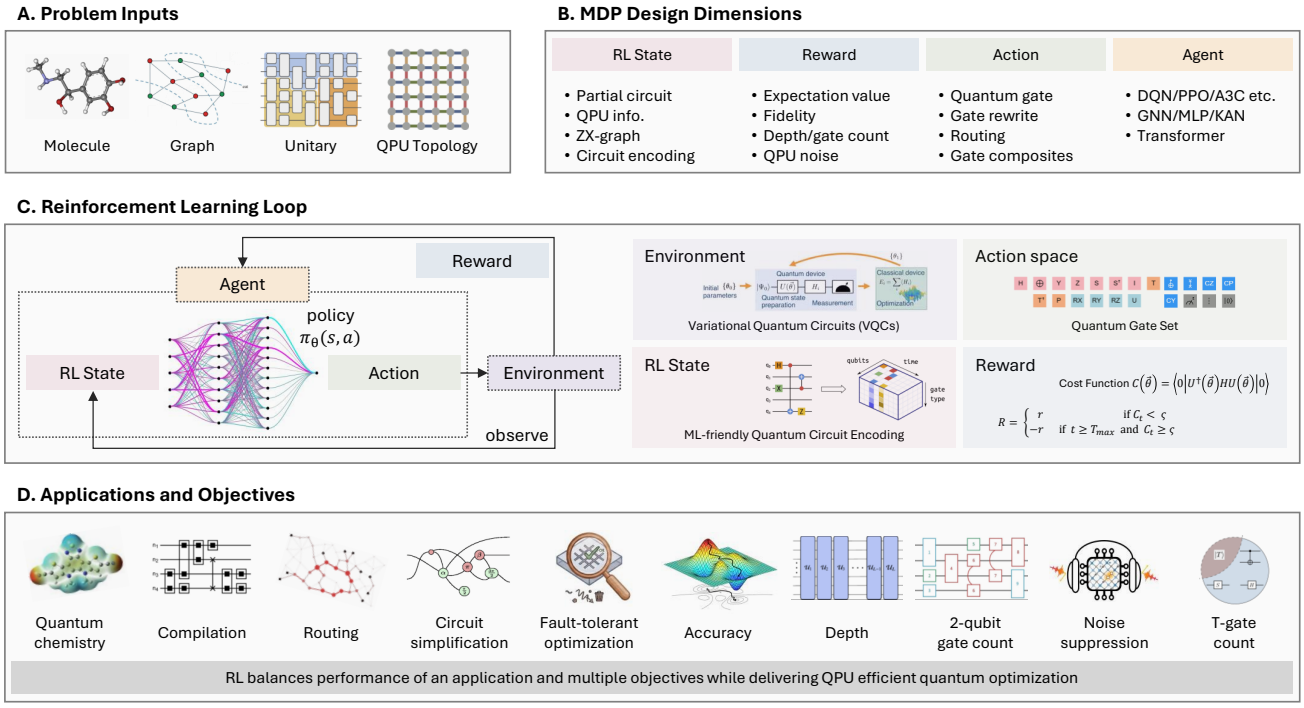

Figure 1 RL-QCO: Reinforcement learning framework for quantum circuit optimization. *A*. Problem inputs such as molecular Hamiltonians, combinatorial graphs, target unitaries and QPU coupling topologies define the circuit-optimization task. *B*. The four Markov decision process components, i.e., state, reward, action and agent admit different instantiations across RL-QCO methods and serve as the main organizing axes of this study. *C*. An RL agent interacts with a quantum circuit environment by observing a state s_t, selecting an action a_t (e.g. gate insertion, rewrite, routing move or macro operation) and receiving a reward r_t derived from task performance and circuit resources. *D*. Representative applications include circuit design for ground state prepration of quantum chemistry problems, compilation, routing, ZX-calculus simplification and fault-tolerant T-count optimization.

Hybrid quantum-classical schemes have emerged as a pragmatic response to these limitations. Among them, variational quantum algorithms (VQAs) have become a central approach for leveraging NISQ devices [20, 21]. In a typical VQA, a parameterized quantum circuit (PQC) $U(\boldsymbol{\theta})$ prepares a quantum state, a cost function which often the expectation value of a Hamiltonian H is evaluated as $C(\boldsymbol{\theta}) = \langle \mathbf{0}|U^\dagger(\boldsymbol{\theta})HU(\boldsymbol{\theta})|\mathbf{0}\rangle$, and a classical optimizer iteratively updates the parameters $\boldsymbol{\theta}$ to minimize this objective. The overall performance of such schemes is therefore tightly linked to the choice of *ansatz* for the PQC. In practice, circuit layouts are usually fixed in advance, either by following hardware-efficient templates [22]; in ref. [23] the authors argued that such structures are trainable on quantum hardware only in a narrow "Goldilocks" regime of shallow depth, local measurements, and area-law input states that are classically hard to simulate. Another way to construct PQCs is by incorporating problem-specific structure [24, 25, 26, 27, 28]; such ansätze can accurately approximate the ground state but after transpilation the depth and gate count often increase rapidly, making them hardware-inefficient on near-term devices. Although these designs have enabled progress in areas such as quantum chemistry, optimization, and high-energy-physics-inspired models, they often face a delicate trade-off: circuits that are kept shallow for noise resilience may lack expressivity [29, 30], whereas more expressive ansätze can suffer from excessive depth and poor trainability [31, 32, 33].

These limitations have motivated a shift from manually designed ansätze to quantum architecture search (QAS) [34, 35, 36, 37], which seeks to automate the discovery of PQC structures from a predefined pool of quantum gates and forms one important class of quantum circuit optimization problems. In neural architecture search (NAS) [38, 39, 40, 41], families of network topologies are optimized rather than fixing a specific architecture *a priori*, treating architecture design itself as a learning problem. In both settings, architecture design is cast as a search or optimization task over a combinatorial space of building blocks, guided by performance on a validation objective. However, QAS must additionally cope with intrinsically quantum aspects such as entanglement structure [42], non-Clifford resources [43], device noise [44], and restricted qubit connectivity [45], as well as the comparatively high cost of circuit evaluation on quantum hardware or simulators [46]. Closely related optimization problems arise beyond ansatz construction, including circuit compilation [47], hardware-aware routing, ZX-calculus-based simplification, and T-count/T-depth reduction in the early fault-tolerant regime, where the goal is to transform or schedule existing circuits under hardware and noise constraints rather than to invent new parameterized templates.

In QAS, the goal is to identify arrangements of gates and corresponding parameters that minimize a target cost function while respecting hardware constraints, enabling task-specific and hardware-aware circuit architectures. QAS methods now span a broad range of applications, including molecular ground-state preparation [35, 48], QAOA-based optimization [13], circuit compilation [47] and hardware-efficient transpilation [49]. They employ diverse algorithmic tools such as adaptive construction [50], differentiable search [51], evolutionary strategies [52], simulated annealing [53], and surrogate performance predictors [54].

Within this broader landscape of quantum circuit optimization, reinforcement learning (RL) has emerged as a particularly natural framework, as it explicitly treats both circuit construction (e.g., ansatz design and synthesis) and circuit transformation (e.g., compilation, routing, and simplification) as sequential decision-making problems. In RL-based quantum circuit optimization (RL-QCO), an agent incrementally edits a circuit by selecting gates, qubit locations, rewrite rules, or higher-level composites of gates, using feedback from a quantum cost function as a reward signal to update its policy. This paradigm offers two key advantages for quantum circuit optimization. First, RL can operate with limited domain knowledge, exploring unconventional circuit structures that may be hard to anticipate by hand and adapting to tasks where the problem structure is only partially known. Second, RL can incorporate strong priors and physical constraints whenever these are available, for example through symmetry-aware action spaces, initialization from classically obtained states, or reward shaping based on expressivity, locality, or resource measures. In this way, RL serves not as a replacement for physical insight, but as a vehicle to encode, refine, and reuse such insight algorithmically across families of circuits and problem instances.

Despite rapid progress, RL-based quantum circuit optimization remains constrained by several intertwined challenges. Most existing studies demonstrate RL-QCO on relatively small systems, typically up to a few qubits in realistic noise models, due to the exponential growth of the action space and RL state, the need for long episodes to construct or transform deep circuits, and the high cost of querying quantum simulators or devices. Moreover, the design of effective state representations, reward functions, and action abstractions is highly problem dependent, and there is little consensus on best practices. As a result, the literature is fragmented across tasks (e.g., VQE, QAOA, compilation, routing, ZX-calculus simplification, and fault-tolerant resource optimization), environments, and RL algorithms, making it difficult to compare methods, identify common patterns, or assess scalability in a principled way.

Several recent overviews provide broader overviews of quantum architecture search and artificial intelligence for quantum technologies, but optimize for different objectives than the present work. Zhang *et al.* [55] overview artificial intelligence (AI) for science with an emphasis on cross-domain themes such as symmetry, equivariance, and transfer, treating reinforcement learning only tangentially. Alexeev *et al.* [56] summarizes AI across the full quantum computing stack, from hardware calibration to algorithms, and consider RL as one of several paradigms alongside differentiable programming and heuristic search. Duncan *et al.* [57] provide an excellent pedagogical treatment of shortcuts to adiabaticity, optimal control, and RL,

Table 1 Comparison of the present study with closely related overviews. ✓ = covered in depth; ● = partially covered or mentioned; ✗ = not covered. *MDP-axis* indicates whether the paper is organized around state/reward/action/agent design dimensions. RL: reinforcement learning, QAS: quantum architecture search, QC: quantum computing.

Dimension	**Zhang et al.** [55] (FTML, 2025)	**Alexeev, Yuri, et al.** [56] (Nature Comm., 2025)	**Duncan et al.** [57] (PRX Quantum, 2025)	**Bukov & Marquardt** [58] (ArXiv, 2025)	**Martyniuk et al.** [59] (IEEE QCE 2024)	**Our study**
Primary scope	AI for Science	AI for full QC stack	QC tutorial	RL for all quantum tech.	QAS	RL for circuit opt.
RL focus	✗	●	●	✓	●	✓
MDP-axis	✗	✗	✗	✗	✗	✓
Ansätze design	●	●	✗	●	✓	✓
Compilation & synthesis	✗	●	●	●	●	✓
Transpilation & routing	✗	●	✗	✗	●	✓
ZX-calculus simplification	✗	✗	✗	✗	✗	✓
Fault-tolerant regime	✗	●	✗	●	✗	✓
Reward engineering	✗	✗	✗	✗	✗	✓
Action space taxonomy	✗	✗	✗	✗	✗	✓
Hardware-aware RL	✗	●	✗	●	●	✓
Neural architectures	✗	✗	✗	✗	✗	✓
Benchmarking methodology	✗	✗	✗	✗	✗	✓
Analog / pulse-level RL	✗	●	✓	✓	✗	✗
RL for QEC decoding	✗	●	✗	✓	✗	✗

but their RL coverage is focused on analog and pulse-level control rather than gate-level circuit optimization. In a recent paper Bukov and Marquardt [58] offer the broadest reinforcement learning-for quantum perspective to date, covering state preparation, feedback control, error correction, and metrology; circuit optimization is one chapter in this panorama, and the internal organization is not based on Markov decision process (MDP) design choices. Martyniuk *et al.* [59] provide an overview quantum architecture search across heuristic, evolutionary, differentiable, and learning-based methods, treating RL as one search paradigm among many and organizing primarily by search strategy and application domain rather than by RL-agent design.

In contrast, this article concentrates specifically on reinforcement learning for *quantum circuit optimization* and is, to our knowledge, the first RL-exclusive study whose internal structure is determined by MDP design choices rather than by application domain or hardware platform. As summarized in Table 1, we focus on gate-level and circuit-level optimization tasks (ansatz design, compilation and synthesis, transpilation and routing, ZX-calculus simplification, and early fault-tolerant T-count/T-depth reduction) and analyze RL-based methods along four core design axes: (1) *state representations*, including encodings of circuits, problem instances, and hardware knowledge; (2) *reward engineering*, capturing not only energy or fidelity but also circuit depth, gate count, noise robustness, and other resource metrics; (3) *action spaces*, from low-level gate placements to higher-level templates, compiler macros, and rewrite rules; and (4) *agent choices*, encompassing RL algorithms and neural network architectures. By organizing developments from roughly the last decade along these axes, and by explicitly discussing hardware-aware RL, neural architectures, and benchmarking methodology in this unified framework, we highlight shared structures, trade-offs, and open challenges, and outline opportunities for scalable, robust RL-QCO in both NISQ and fault-tolerant regimes.

The remainder of this work is structured around these four design axes. In Sec. 2 we formalize quantum circuit optimization as a constrained cost-minimization problem and introduce the reinforcement learning background needed to interpret RL-QCO methods. Secs. 4-7 then develop the core MDP components for RL-QCO state representations (Sec. 4), reward engineering (Sec. 5), action-space design (Sec. 6), and learning algorithms and agent architectures (Sec. 7); across ansatz design, compilation, routing, ZX-calculus simplification, and fault-tolerant resource optimization. Sec. 8 analyzes evaluation and benchmarking methodology for RL-QCO, including task metrics, resource-aware figures of merit, learning efficiency, and hardware-in-the-loop studies. Finally, Sec. 9 summarizes open challenges and future directions for scalable, robust reinforcement learning for quantum circuit optimization in both NISQ and fault-tolerant regimes.

2 Scope

In this section we describe the boundaries of the study. We first specify what we mean by quantum circuit optimization in the context of this work, including tasks such as variational quantum eigensolver [60], quantum approximate optimization algorithm [61], variational quantum state diagonalization [62] ansatz design, compilation, layout and simplification. We then clarify the focus on RL-based methods versus other automated design techniques, and finally we state the time window and selection criteria for included works.

2.1 What this study covers

This article summarizes reinforcement learning methods applied to *quantum circuit optimization* (RL-QCO), which we define as the use of RL agents to construct, transform or schedule quantum circuits with the objective of minimizing a cost that combines physical performance with hardware resource consumption. Concretely, we include the following tasks:

- **Circuit design for variational algorithms.** RL-based search over parameterized quantum circuit (PQC) structures for the variational quantum eigensolver (VQE) [60], the quantum approximate optimization algorithm (QAOA) [61] and related variational methods.
- **Circuit compilation and unitary synthesis.** RL approaches that approximate a target unitary or quantum state by a gate sequence drawn from a discrete gate set, including Clifford+T and native-gate synthesis.
- **Transpilation.** RL methods for qubit mapping, SWAP-based routing and gate scheduling onto device coupling graphs.
- **Circuit simplification and rewriting.** RL agents that iteratively apply equivalence-preserving rewrite rules, such as ZX-calculus rewrites, to reduce gate count or circuit depth.
- **Fault-tolerant resource optimization.** RL approaches targeting T-count or T-depth minimization relevant to magic-state distillation in the early fault-tolerant regime.

In all cases, the RL interaction loop is organized around four core MDP design dimensions: *state representation*, *reward engineering*, *action space* and *agent architecture*, which serve as the organizing axes of the paper from Sec. 4 to Sec. 7.

2.2 What this study does not cover

Several related areas are deliberately excluded in order to maintain a focused treatment. Major excluded directions include:

- **Non-RL quantum architecture search (QAS).** Differentiable QAS [51], evolutionary strategies [52], Bayesian optimization [63], simulated annealing [53] and unsupervised learning [64] over PQC structures are outside scope. These methods are discussed comprehensively by Martyniuk *et al.* [59].
- **Continuous quantum control and pulse optimization.** RL methods that act on analogue drive amplitudes or pulse shapes rather than discrete gate sequences are excluded. This regime is covered in depth by Duncan *et al.* [57].
- **RL for quantum error-correction (QEC) decoding.** RL decoders for the surface code and related stabilizer codes operate on syndrome-measurement sequences rather than circuit-construction problems and are excluded. For example, ref. [65] discusses RL-based toric code decoders, while message-passing decoders for QLDPC codes are treated in ref. [66].
- **RL for quantum feedback and metrology.** Adaptive measurement protocols and quantum-parameter estimation via RL fall outside the circuit-optimization framing used here. We refer the reader to refs. [67, 68] for in-depth discussions of these topics.
- **Quantum reinforcement learning (QRL).** Methods in which the *agent itself* is implemented on a quantum computer, notably refs. [69, 70], are outside scope. This summarize concerns classical RL applied to quantum circuit tasks. Recent work on benchmarking quantum RL algorithms [71] highlights the distinct methodological questions that arise in that setting.
- **Classical circuit and logic synthesis.** RL applied to Boolean satisfiability, technology mapping or classical EDA is not covered. For such applications we refer the reader, for example, to ref. [72], which discusses SAT-based RL agents for MaxSAT.

2.3 Relation to existing works

Table 1 situates this study with respect to five closely related overview s on artificial intelligence (AI) and quantum computing [55, 56, 57, 58, 59]. As discussed in Sec. 1, these works optimize for complementary objectives such as cross-domain AI for science coverage, full-stack quantum-computing overviews, pedagogical treatments of analogue control and RL, or comprehensive catalogs of quantum architecture-search paradigms. In all cases, reinforcement learning for quantum circuit optimization (RL-QCO) appears as one component of a broader landscape. In contrast, the present article narrows the scope to RL-based circuit optimization and organizes methods by their Markov decision process design choices, providing a dedicated taxonomy of state representations, rewards, action spaces and agent architectures for RL-QCO.

2.4 Literature selection

We cover works published between approximately 2018 and mid 2026, spanning the period from the first applications of deep RL to quantum circuit tasks through to recent scalability and hardware-integration advances. Papers were included if they (i) use an RL agent whose policy, value function or model is trained by interaction with a quantum circuit environment, and (ii) target at least one of the circuit-optimization tasks defined in Sec. 2.1. Works that use RL only as a classical subroutine unrelated to circuit structure (for example, hyperparameter tuning) are excluded. Where multiple versions of a method exist (such as a conference paper followed by a journal extension), the most complete version is cited, and earlier versions are mentioned when the progression is instructive.

3 Background

This section consolidates the background needed to read the rest of the article from a unified standpoint. We first formalize quantum circuit design as a constrained optimization problem that subsumes parameterized quantum circuit design, compilation, transpilation, simplification, and resynthesis under a cost-and-constraint template. We then summarize reinforcement learning developments at the level needed to follow later design choices, thereby recasting circuit optimization as an MDP whose states, actions, and rewards instantiate the four design choices (state, reward, action, and agent) around which this study is organized.

3.1 Quantum circuit design and optimization

A quantum circuit is an ordered sequence of unitary gates drawn from a gate set $\mathcal{G}$, written $U = G_L \cdots G_2 G_1$, where L is the total number of gate applications. Gates that act on disjoint qubits in the same time step form a moment (or layer); the circuit depth $D(U)$ is the number of moments and the width $W(U)$ is the number of qubits actively used. Parameterized circuits $U(\theta) = \prod_\ell G_\ell(\theta_\ell)$ contain rotations whose angles $\theta \in \mathbb{R}^p$ are tuned by an outer optimizer [1, 21]. We further distinguish a logical circuit, defined over an abstract gate set with all-to-all connectivity, from a physical circuit that respects the coupling graph $G_{\text{hw}} = (V, E)$ of a target device, where V indexes physical qubits and E enumerates the pairs on which native two-qubit gates can be executed.

Quantum circuit optimization spans several distinct but related tasks, each cast as a search over circuits $U \in C$ for a constraint set C that fixes the available gate set, the allowed connectivity, and a depth or gate-count budget. (i) Ansätze design for variational quantum algorithms such as the variational quantum eigensolver [60] and the quantum approximate optimization algorithm [61] chooses a parameterized template $U(\theta)$ that is expressive enough to represent the target state yet shallow enough for NISQ hardware [22]. (ii) Unitary or state compilation approximates a target unitary $U_\star$ or state $|\psi_\star\rangle$ by a circuit over a discrete universal gate set, classically guaranteed by the Solovay-Kitaev decomposition [73, 74]. (iii) Transpilation maps a logical circuit to a physical one via three coupled subproblems: qubit mapping (initial logical-to-physical assignment), routing (insertion of SWAPs to satisfy connectivity), and scheduling (assigning gates to time slots) [75]. (iv) Simplification and gate cancellation rewrite a circuit while preserving its action, removing redundant rotations, commuted CNOT pairs, or trivial identities. (v) Clifford+T resynthesis targets fault-tolerant cost models by minimizing the T-count and T-depth, since non-Clifford operations dominate the cost of magic-state distillation [76]. (vi) Noise- and error-aware optimization co-designs the circuit with the device by selecting paths through low-error qubits and applying ZX-calculus rewrites that simplify the diagram before resynthesis [77].

All of the tasks discussed above can be cast as the constrained optimization

$$U^\star \in \arg\min_{U \in C} J(U), \tag{1}$$

where C encodes the admissible circuits and $J(\cdot)$ is a possibly multi-objective cost. In a resource constrained setting, one writes $J(U) = \alpha\mathcal{E}(U) + \beta\mathcal{R}(U)$, with $\mathcal{E}(U)$ a physical objective such as the ground state energy or infidelity and $\mathcal{R}(U)$ a resource term penalizing depth, two-qubit gate count, active volume, or estimated noise. Such a formulation is more pronounced in the NISQ and early fault-tolerant (EFT) era. Fault-tolerant compilation, in contrast, typically fixes an exact-equivalence constraint $U \equiv U_\star$ and minimizes a discrete resource cost, most prominently the T-count [78]. Variational ansätze design, transpilation and resynthesis are thus instances of the same template differing only in the choice of $\mathcal{E}$, $\mathcal{R}$ and C, which is the unifying view we adopt in this article.

The constraint set C is shaped primarily by the target device. Native gate sets restrict the elementary operations to a small alphabet (e.g.,$\{\sqrt{X}, \texttt{RZ}, \text{ECR}\}$ on superconducting hardware or $\{\texttt{RX}, \texttt{RY}, \texttt{RZZ}\}$ on trapped ions); connectivity restrictions forbid two-qubit gates outside $E \subseteq V \times V$; gate-time budgets are imposed by qubit coherence times T_1, T_2, which together with measured per-gate errors yield decoherence-induced feasibility windows; readout errors and crosstalk further constrain

admissible scheduling [79]. $J(U)$ can be evaluated at various abstraction levels like noiseless state vector simulation, density-matrix-based simulation under a noise model, and execution on calibrated quantum hardware.

The most common task-specific figure of merit for variational algorithms is the expectation value $\langle H \rangle_{U(\theta)} = \langle 0|U^\dagger(\theta) H U(\theta)|0\rangle$ of a problem Hamiltonian H, while compilation tasks measure the discrepancy from a target through the state fidelity $F(|\psi\rangle, |\phi\rangle) = |\langle\psi|\phi\rangle|^2$ or the average gate fidelity of the implemented channel. Resource metrics quantify cost on hardware, for example, the total gate count $|U|$, the depth $D(U)$, the two-qubit gate count (CNOT or ECR), the SWAP overhead introduced by routing, and, in the fault-tolerant regime, the T-count and T-depth. A device-aware proxy that is widely used during training is the estimated success probability $P_{\text{succ}}(U) \approx \prod_{g\in U}(1-\varepsilon_g)$, where ε_g is the calibrated error of gate g [79]. These metrics are coupled by intricate trade-offs. Deeper, more expressive ansätze cover a larger region of state space but suffer from barren plateaus and accumulated noise [80], while shallow, hardware-friendly designs may under-fit the target [42, 81]. The notion of equivalence between two circuits also depends on the task. Exact unitary equivalence (up to a global phase) is required in fault-tolerant resynthesis, approximate equivalence at fidelity threshold is standard in NISQ compilation, and observable equivalence on a fixed input $|\psi_0\rangle$ ($\langle O\rangle_{U_1|\psi_0\rangle} = \langle O\rangle_{U_2|\psi_0\rangle}$) is sufficient when the circuit is only queried through a specific measurement (e.g., shadow observables). However, even for modest n and depth budget, the size of C scales as $|\mathcal{G}|^{\Omega(nD)}$, and the underlying decision problems are NP-hard already for qubit mapping on standard 2D topologies [82]. This complexity motivates the learning-based search strategies studied in the rest of the article.

The optimization problem $\min_{U\in C} J(U)$ can be cast as a finite-horizon Markov decision process whose actions modify the current circuit (insertion, deletion, or rewriting of gates), whose states encode the partial circuit together with optionally observable or hardware information, and whose rewards are derived from $-J(U)$ or from its incremental change after each action [83]. Secs. 4 and 5 unfold this MDP view by overviewing state encodings and reward designs explored in RL-QCO research.

3.2 Reinforcement learning

A Markov decision process (MDP) provides the standard mathematical formulation for RL [83]. An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s'|s,a)$ is the transition kernel, $r(s,a)$ is the reward function, $\gamma \in [0,1)$ is the discount factor, and ρ_0 is the initial-state distribution. An agent interacts with the environment by sampling $s_0 \sim \rho_0$ and, at each step t, drawing an action a_t from a policy, receiving $r_t = r(s_t, a_t)$, and transitioning to $s_{t+1} \sim P(\cdot|s_t, a_t)$. The induced trajectory is $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$. Tasks are termed episodic when τ terminates after a finite horizon and continuing otherwise. The discounted return $G_t = \sum_{k\geq 0} \gamma^k r_{t+k}$ summarizes long-term performance, and the Markov property states that $P(s_{t+1}|s_t, a_t)$ is independent of the past given (s_t, a_t). When the agent only has access to a partial observation o_t rather than s_t, the formalism generalises to a partially observable MDP (POMDP) [84], a regime that typically applies to RL-QCO whenever limited circuit evaluation shot budgets prevent full-state access.

A policy $\pi(a|s)$ specifies how actions are chosen and may be stochastic or deterministic; the latter is the limit $\pi(a|s) = \delta(a - \mu(s))$ for some deterministic map $\mu : \mathcal{S} \to \mathcal{A}$. Two value functions characterzse π, the state-value $V^\pi(s) = \mathbb{E}_\pi[G_t \mid s_t = s]$ and the action-value $Q^\pi(s,a) = \mathbb{E}_\pi[G_t|s_t = s, a_t = a]$. These two are related through the advantage $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$, which measures how much better an action is than the policy's average behaviour. Both satisfy the Bellman expectation equations, e.g., $V^\pi(s) = \mathbb{E}_{a\sim\pi, s'\sim P}[r(s,a) + \gamma V^\pi(s')]$, while the optimal value functions $V^\star, Q^\star$ obey the Bellman optimality equations $V^\star(s) = \max_a \mathbb{E}_{s'}[r(s,a) + \gamma\, V^\star(s')]$. An optimal policy is thus any $\pi^\star \in \arg\max_\pi V^\pi$. For finite MDPs, at least one deterministic optimal policy is guaranteed to exist [85]. The discount factor γ controls the credit-assignment horizon. Small γ favours short-horizon decisions, whereas γ close to 1 is needed to propagate sparse terminal rewards back through long episodes.

RL algorithms that are relevant to quantum circuit optimization can be categorized into four families. (i) Value-based methods learn Q directly through bootstrapped Bellman updates. Tabular Q-learning [86] is provably convergent in the finite case, and its deep variant DQN scales to large state spaces by combining a neural Q-network with experience replay and a slowly updated target network [87], with refinements such as Double DQN [88] and prioritized replay [89]. (ii) Policy-gradient methods directly optimize a parameterized policy π_θ via the policy-gradient theorem $\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) A^\pi(s,a)]$, with REINFORCE [90] as the canonical example. (iii) Actor-critic methods combine a policy (actor) with a learned value baseline (critic) to lower variance. These methods span synchronous and asynchronous variants A2C/A3C [91], the trust-region updates of TRPO and the clipped surrogate objective of PPO [92], and entropy-regularised continuous-control methods such as SAC [93]. (iv) Model-based and search-augmented methods learn or assume a transition model and plan within it, ranging from Dyna-style integration of planning and learning [94] to AlphaZero and MuZero, which interleave Monte Carlo tree search with deep value/policy networks [95]. Value-based methods are typically chosen for discrete, low-dimensional action spaces; policy-gradient and actor-critic methods are preferred when actions are continuous or stochastic policies are desired; while model-based or search-augmented methods are useful when a simulator is cheap, samples are expensive, or long-horizon planning over combinatorial structures is required.

Effective learning requires a careful trade-off between exploiting the current best estimate of π and exploring under-sampled regions of $(\mathcal{S}, \mathcal{A})$. Standard mechanisms include ϵ-greedy and Boltzmann/softmax exploration in value-based methods, and entropy regularization in policy-gradient and actor-critic methods, where a bonus $-\beta\mathcal{H}[\pi_\theta(\cdot|s)]$ is added to the objective to prevent premature collapse onto a deterministic policy. When rewards are very sparse, *intrinsic-motivation* signals are useful, augmenting the external task reward with auxiliary bonuses that encourage exploration of novel or informative states [96, 97]. These range from count-based bonuses [98] to prediction-error curiosity [99]. Curriculum learning and progressive task schedules further help by exposing the agent to easier instances before harder ones [100]. In RL-QCO, exploration is particularly difficult because the action space scales combinatorially with circuit width and depth, every reward query may require an expensive simulation or hardware shots, and the cost landscape contains barren plateaus of vanishing gradients that masquerade as local optima.

Beyond small problems, value and policy functions are no longer represented as tables but as parameterized approximators. Deep RL replaces these lookup tables with neural networks tailored to the structure of the observation. These include multilayer perceptrons (MLP) for fixed-length feature vectors [35], convolutional networks (CNN) for grid-like qubit/moment tensors [36], graph neural networks (GNN) for ZX-diagrams or coupling graphs [101], and transformers for long token sequences over gate alphabets [81]. While function approximation enables generalization, combining it with bootstrapping and off-policy learning produces the well-known deadly triad [83], which can cause divergence even on benign problems. Practical RL-QCO pipelines therefore rely on a recurring toolbox of stabilisers. These include target networks and replay buffers, reward shaping that aligns intermediate signals with the final objective [102], action masking to forbid illegal or trivial gate placements [103], and observation/return normalisation to keep features and gradients on comparable scales.

Reporting RL performance requires distinguishing training-time and test-time behaviour. Training metrics typically include the average return, success rate, and sample complexity, i.e., the number of environment interactions needed to reach a target threshold, while test-time metrics emphasise zero-/few-shot generalisation to unseen problem instances, larger qubit counts, or noise models not encountered during training. Common pitfalls in benchmarking include high variance across random seeds and strong sensitivity to hyperparameters [104], which signifies variability over multiple seeds on standardised environments. Gym/Gymnasium-style APIs [105, 106] have become the de facto interface, and the RL-QCO community has begun to adopt them through dedicated benchmarks such as qcd-gym [107] and qgym [108], which allows circuit construction, transpilation, and routing tasks using a unified interface.

3.3 Reinforcement learning framework for quantum circuit optimization

The interaction loop underlying RL-based quantum circuit optimization is illustrated in Fig. 1(b). At each decision step t, the environment exposes a state s_t that encodes information about the current circuit and, optionally, the problem instance and hardware. In ansatz-design and synthesis settings, s_t typically describes a partial circuit together with current cost estimates, whereas in compilation, routing or ZX-simplification tasks it may encode a transformed version of an input circuit, a residual operator, or a circuit-equivalent graph representation. Hardware-aware variants further augment the state with device characteristics such as the coupling graph or qubit- and gate-specific error rates.

Given s_t, the RL agent samples an action a_t from its policy $\pi(a_t \mid s_t)$. Actions correspond to local modifications of the circuit or its schedule, for example inserting or deleting gates, applying equivalence-preserving rewrite rules, performing routing moves such as SWAP insertion, or invoking higher-level compiler macros and gate composites. The environment applies a_t, updates the circuit and any associated classical quantities (for example optimized parameters or accumulated depth), and transitions to the next state s_{t+1}, thereby defining the Markovian dynamics of the RL-QCO task. Episodes terminate once a stopping criterion is met, such as reaching a maximum circuit depth, satisfying a fidelity or cost threshold, or exhausting a fixed action budget.

The reward signal r_t measures progress towards the optimization objective. In many RL-QCO formulations, rewards are derived from a task-specific cost functional that combines a physical performance term (such as energy expectation value or state/operation fidelity) with resource penalties reflecting depth, two-qubit gate count, routing overhead, estimated noise or fault-tolerant cost. Rewards may be issued only at the end of an episode, based on the quality of the final circuit, or shaped to provide intermediate feedback that correlates local modifications with eventual performance. The choice of state, action space, transition structure and reward function together specifies a concrete Markov decision process for quantum circuit optimization and forms the basis for the design taxonomy developed in the following sections.

4 State representation

In reinforcement learning, the state specifies the information available to the agent when choosing actions, and its design is especially critical in quantum circuit optimization. For RL-based quantum circuit optimization (RL-QCO), the state

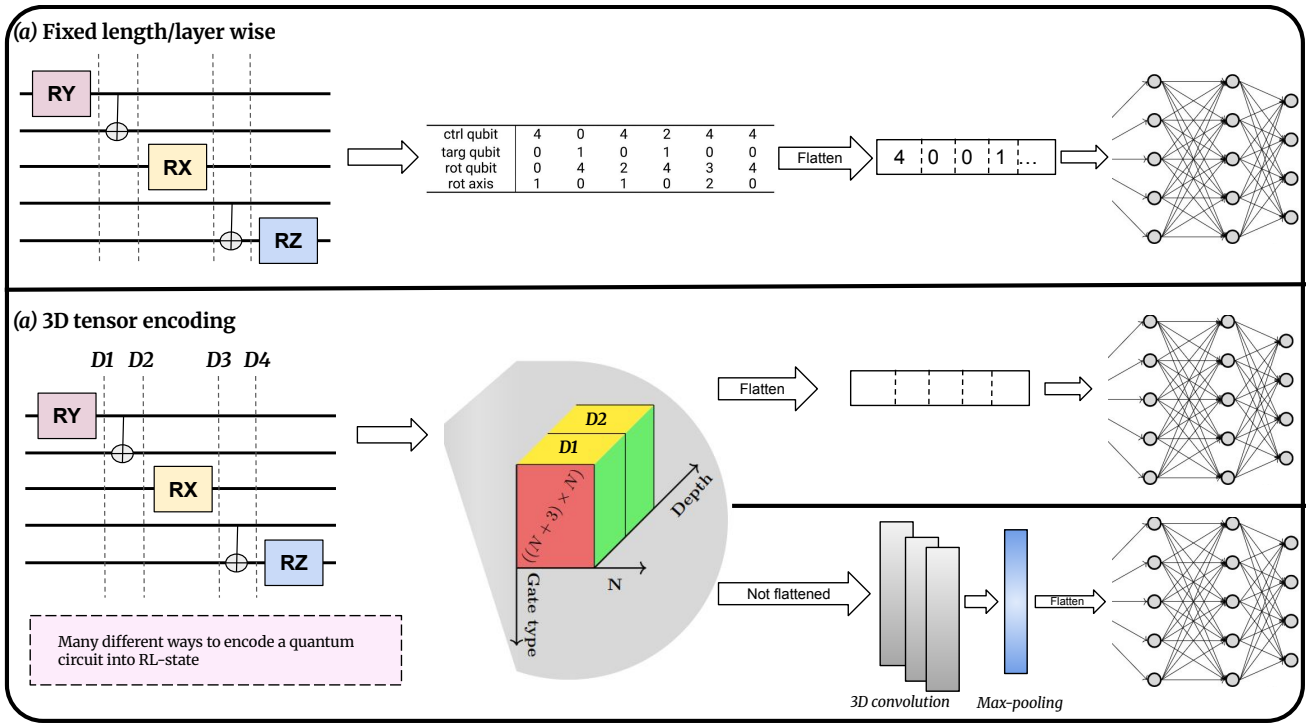

Figure 2 Circuit-centric state encodings for RL-based quantum circuit optimization. **a**, Fixed-length, layer-wise encoding of a quantum circuit, where each layer is mapped to discrete descriptors (e.g. control, target and rotation-qubit indices and rotation axis), padded as needed and flattened before being processed by a feed-forward network. **b**, Three-dimensional tensor encoding on a qubit–depth–gate-type grid, which can be flattened or processed by convolution and pooling before being passed to a neural network. These examples illustrate how different circuit encodings expose different structural information to the RL agent and how naturally they interface with standard neural architectures.

representation determines which aspects of the circuit, problem instance, and hardware are exposed to the agent, and thus directly influences sample efficiency, generalization behavior, and scalability.

In this work, existing RL-QCO methods are grouped along three main axes of state design: (i) how the *circuit* itself is encoded (for example, as a sequence of gates, a graph-based structure, or a tensor/network representation); (ii) how the state is *augmented* with additional signals, such as current cost values, optimized angles, or counts of function evaluations, which are central in variational quantum algorithms; and (iii) how *problem and hardware information* such as the Hamiltonian structure, qubit coupling graphs, and noise characteristics is incorporated. The remainder of this section summarizes these design choices and analyzes their consequences for RL-QCO.

4.1 Circuit-level representation

At the circuit level, most RL-based quantum circuit optimization methods represent the environment state as a structured, finite-dimensional object that encodes which gates act on which qubits at which time steps. Concretely, a circuit is mapped either to (i) a fixed-length sequence of gate descriptors, where each position stores discrete attributes such as gate type, control and target qubits, and rotation axis, or to (ii) a grid- or tensor-based representation, where qubits and circuit moments form spatial axes and gate classes appear as channels. In both cases, padding or special "no-gate" symbols are used to handle variable depth, and continuous gate parameters are frequently omitted from the state and handled by a separate optimization routine. Within this general framework, different works instantiate distinct encoding, which we summarize next.

One of the early works [35], represented in Figure 2(a) the circuit-level state is represented as a fixed-length, layer-wise encoding tailored to a value-based deep RL agent. Each environment state corresponds to an ordered list of at most L layers, where each layer contains a single quantum gate acting on a system of $|Q|$ qubits. CNOT gates are encoded by two integer indices specifying the control and target qubits, while single-qubit rotation gates are encoded by the qubit index together with an integer label for the rotation axis (e.g., $X \to 1, Y \to 2, Z \to 3$). If the actual circuit has fewer than L layers, the remaining positions are padded with identities, implemented by using a special "no-gate" code (chosen as the maximum qubit index) so that the entire circuit can be represented as a rectangular integer array suitable as input to a neural network. A key design choice is that gate parameters are deliberately excluded from the state; instead, an external classical optimizer (e.g., COBYLA or Rotosolve) tunes these angles, and the resulting energy estimate is appended to the state as a scalar feature [35].

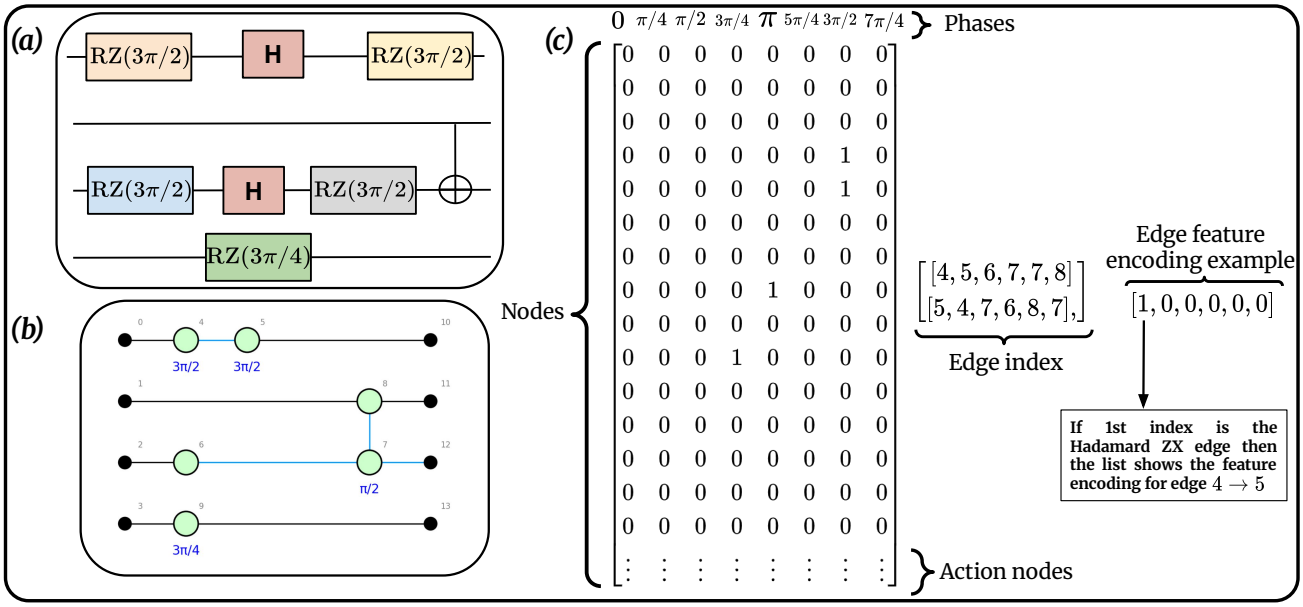

Figure 3 Graph-based circuit representation via ZX diagrams for RL-QCO. (a) Example quantum circuit built from single-qubit rotations and Hadamard gates. (b) Corresponding ZX diagram, where spiders (green nodes) carry phase labels and edges encode connectivity, including Hadamard edges. (c) Graph-based state used as input to a graph neural network: node features include phase information, while edge indices and edge-feature vectors encode the ZX connectivity (with, for example, a dedicated Hadamard indicator). Action nodes specify candidate locations where the RL agent may apply ZX rewrite rules.

In ref. [36], illustrated in Figure 2(b), the environment state is a complete description of the current quantum circuit, encoded on a regular three-dimensional grid suitable for a convolutional neural network. The grid axes correspond to the qubit index, the circuit moment (time step), and a discrete gate-class channel, so that each lattice site stores which gate class (if any) acts on a given qubit at a given moment [36]. This yields a tensor of shape (qubits) × (moments) × (gate classes), where the absence of a gate is represented by an "empty" channel, and the entire tensor is passed as the observation to the agent's deep CNN. The same indexing scheme is reused at the output: each possible local circuit transformation is mapped injectively to one policy neuron arranged on a grid over qubit index, moment, and transformation rule, with invalid actions masked out. This fully convolutional, grid-based encoding treats the circuit as an image-like object, enabling weight sharing across qubits and time steps and allowing the trained agent to extrapolate to circuits larger than those seen during training [36].

In CRLQAS [109], illustrated in Figure 2(c), the authors adopt a circuit-centric state representation based on a regular three-dimensional tensor whose axes correspond to qubit index, circuit depth (moment), and gate type. Each element of this tensor specifies which gate class, if any, acts on a particular qubit at a given moment, so that a complete circuit up to a fixed maximum depth is encoded as an array of shape (qubits) × (moments) × (gate types). This design is closely related to the fixed-length, layer-wise encoding in ref. [35], where each layer is described by explicit integer fields for control and target qubits, rotation qubits, and rotation axes, with shorter circuits padded by special "no-gate" entries. In contrast, CRLQAS subsumes gate identity into the gate-type channel dimension, yielding a more compact tensor representation that can be ingested directly by a feedforward policy/value network.

Another emerging direction for constructing the RL-state is by representing the quantum circuit as graph, notably via ZX-diagrams [110] and related graph encoding such as directed acyclic graphs [111]. Notably, in this setting, the circuit is first translated into a reduced ZX-diagram, which is then viewed as a graph whose vertices (termed as spiders) carry local attributes such as type, phase, and boundary status, and whose edges encode the pattern of (Hadamard) connections between spiders and inputs/outputs [101]. As illustrated in Figure 3, this graph serves as the observation to a graph neural network whose node and edge features capture spider type, phase, and connectivity, while the policy outputs which ZX rewrite rule to apply and where in the diagram [101]. Although this representation is diagrammatic rather than a time-ordered gate list or qubit–moment grid, it is still circuit-level as the graph fully specifies the current computation at the level of a circuit-equivalent intermediate representation i.e. a lower-level form used internally by compilers. Closely related graph-based states also arise in RL that operate on circuit or hardware graphs; such as ZX-graph simplifiers [112], and noise-adaptive mapping on coupling graphs [113] highlighting graph encoding as an increasingly important alternative to sequence or tensor-based circuit encoding for RL-based QCO.

In ref. [114], the observation at time step t is the current encoding circuit U_t, represented as a layered quantum circuit where each layer applies a chosen unitary from a fixed gate set to all accessible qubits in a nearest-neighbor layout. This structured

circuit description $o_t = U_t$ is then processed by MuZero's [115] representation network to produce a latent state, so the agent effectively reasons over gate-layer configurations rather than explicit quantum states or Pauli measurement statistics.

Another solid example of a circuit-level representation appears in the RL-based transpiling framework of ref. [116]. In circuit synthesis, the agent observes the current residual operator O_t, i.e. the operator that remains to be reduced to the identity through successive gate applications, so that the RL state can be written abstractly as $s_t = \mathrm{Enc}(O_t)$. For Clifford synthesis, this encoding is implemented through a tableau-style Boolean representation of the operator, using a $2N \times 2N$ matrix (ignoring the phase vector), which is reshaped into a tensor before being fed to the neural network. In the routing setting, the state is likewise circuit-level: the current partially routed circuit is encoded layer by layer as a binary tensor, for example $s_t = \mathrm{Enc}(C_t^i) \in \{0,1\}^{N\times N\times H}$, where C_t^i denotes the remaining input circuit at step t and H is a fixed routing horizon. Thus, throughout both synthesis and routing, the agent acts directly on representations of the evolving circuit object rather than on state-vector, observable, or purely problem-level features.

Table 2 RL-state representations in quantum circuit optimization.

Work / type	RL state representation	Hardware info
Circuit-level		
Ostaszewski et al. [35]	Fixed-length gate sequence with "no-gate" padding	None
Fösel et al. [36]	Circuit tensor (qubits × moments × gate classes)	None
CRLQAS [109]	Circuit tensor (qubits × moments × gate types)	Noise in env., not in state
Rapp et al. [114]	Layered encoding circuit U_t (gate layers)	Topology fixed (implicit)
Riu et al. [101]	ZX graph with spider/node and edge features	None
Quantum state-level		
Kuo et al. [117]	Length-$3n$ vector of single-qubit Pauli expectations	None
Moro et al. [47]	Probe-state images (e.g. $\|0\rangle$, $\|+\rangle$) under unitary	None
qcd-gym [107]	Full state vector or unitary plus target state	Simulator-only
Liao et al. [118]	Vector of local fidelities/overlaps on qubit neighborhoods	Measurement-based
Hardware-level		
Wang et al. [119]	Hardware-realizable unitaries with invert-cost label	Native gates, coupling
qgym [108]	Logical–physical mapping, device graph, gate/time-slot info	Topology, fidelities
DeepQMap [120]	Circuit graph, hardware graph, per-qubit/link noise	Topology, noise
Problem-level		
Yao et al. [121]	Final energy/cost of prepared ground state	Hamiltonian only
Wauters et al. [122]	Final QAOA cost value	Problem instance
Patel et al. [123]	Final MaxCut / Ising cost in recursive QAOA	Problem instance
AlphaTensor-Quantum [124]	Residual signature tensor and factor history	None

4.2 State-level representation

In this section we exemplify measurement- and state-based encodings that largely abstract away hardware details, trading hardware-agnostic observations for simulator-heavy access to quantum states or observables with varying degrees of hardware compatibility.

The RL-state in ref. [117] is a tomography-inspired summary of the current quantum state. At each time step, for an n-qubit system, the environment returns a real-valued observation vector of length $3n$ containing the single-qubit Pauli expectation values $\langle\sigma_j^x\rangle, \langle\sigma_j^y\rangle, \langle\sigma_j^z\rangle_{j=1}^n$, which is recomputed after each applied gate and fed into the neural network. This state-level encoding deliberately discards the gate sequence and any explicit hardware information, forcing the agent to infer progress toward the target state purely from local measurement statistics, while keeping the observation dimension scalable and hardware-agnostic so that the same interface can be employed for both noiseless and noisy simulators.

In ref. [47], the environment state is defined in terms of how the current circuit transforms a small set of probe states rather than via an explicit gate list or full unitary matrix. At each step, the agent observes the images of representative inputs such as $|0\rangle$ and $|+\rangle$ under the partial circuit, together with their corresponding target images under the desired unitary, effectively

as trajectories on the Bloch sphere. This yields a compact state-level encoding that operationally captures the discrepancy between the current compiled unitary and the target, providing geometrically rich feedback while avoiding the overhead of representing the full unitary or exposing circuit or hardware details directly.

In the qcd-gym framework [107], the RL state representation operates at the state-vector level rather than encoding the circuit structure explicitly. At each time step t, for an η-qubit system, the agent observes the full complex state vector $|\Psi\rangle$ (or the unitary $V(\Sigma_t)$ for unitary compilation tasks) produced by the current gate sequence Σ_t, concatenated with the target state $|\Phi\rangle$ (or target unitary U), yielding a high-dimensional observation in $\mathbb{C}^{2^\eta+\text{number of target features}}$ or $\mathbb{C}^{2^{2\eta}+\text{number of target features}}$. This maximally informative encoding discards gate-level details and hardware topology, providing direct access to the circuit's output for feedback while remaining Markovian and reversible, though it relies on efficient simulator state-vector readout and becomes impractical on real hardware due to the exponential measurement overhead for tomography. By including the fixed target alongside the current state, the representation supports continuous control tasks analogous to classical robotics, with potential extensions to density-matrix forms for mid-circuit measurements at quadratic state-space cost.

More recently in [118] the environment state is defined at the level of experimentally accessible observables. At each step t, the agent receives a real-valued observation vector whose components are local figures of merit e.g., fidelity or overlaps between the current and target states evaluated on small neighborhoods of qubits, so that each entry quantifies how well the present circuit reproduces the target reduced state in a given region. This local-fidelity encoding provides a compact, hardware-compatible state-level representation that highlights where the approximation is deficient, enabling the agent to target corrections without requiring full tomography or a circuit-level description.

Finally in [125] the authors introduce an interesting structured state-level representation for fault-tolerant logical state preparation. In a nutshell the representation is defined as follows. Let U_t denote the partial circuit constructed up to time step t. The corresponding prepared state is

$$|\psi_t\rangle = U_t|0\rangle^{\otimes n}. \tag{2}$$

Instead of feeding the raw circuit description as in Sec. 4.1 or the full state vector to the agent, the environment observation is taken to be the canonical stabilizer tableau of the current state, $s_t = T_{\text{can}}(|\psi_t\rangle) = T_{\text{can}}\big(U_t|0\rangle^{\otimes n}\big)$. Here, $T_{\text{can}}(\cdot)$ denotes the canonical tableau map, which assigns the same representation to different Clifford circuits whenever they prepare the same stabilizer state. This removes redundancy from the observation space and yields a representation that scales polynomially with the system size. In this sense, the encoding is state-level rather than problem-level: the agent does not observe only a scalar objective or problem descriptor, but a structured algebraic summary of the currently prepared stabilizer state.

4.3 Problem-level representation

Beyond the task of RL for circuit construction, we get a family of RL applications that operate purely at the problem-level RL-state representation. Here the agent interacts only through scalar performance feedback tied to an objective such as ground-state energy or combinatorial cost. In RL-assisted many-body ground-state preparation inspired by counterdiabatic driving [121], the agent selects sequences and durations of control generators from a fixed set and learns solely from the final energy of the prepared state, without any explicit encoding of the intermediate quantum state or circuit layout.

Similarly, in RL-assisted QAOA [122] and its recursive variant [123], the agent adjusts QAOA angles, schedules, or variable-elimination strategies based on the achieved cost-function value e.g., Ising energy or MaxCut value, so that the RL loop optimizes a problem-solving protocol rather than performing quantum circuit architecture search or low-level gate-sequence optimization.

More recently, in AlphaTensor-quantum [124], the RL-state is defined entirely in terms of algebraic data structures rather than quantum circuits or hardware, making it a problem-level representation. At step s of TensorGame, the state consists of the current residual signature tensor $\mathcal{T}(s) \in 0, 1^{N\times N\times N}$ together with the history of previously chosen rank-1 factors $u(1), \ldots, u(s-1)$, where each $u(r) \in 0, 1^N$ corresponds one-to-one to a non-Clifford T gate in the compiled circuit. This encoding captures the T-count optimization problem as low-rank Waring decomposition of $\mathcal{T}$ and is thus purely combinatorial: it exposes neither a gate-level circuit layout nor a quantum state or QPU topology, and hardware costs enter only indirectly via the reward design for example, gadget-based bonuses rather than through explicit hardware features in the state.

4.4 Hardware-level representation

In contrast to state-level encodings that abstract away the information related to quantum hardware, in the following we study an RL-framework that integrate QPU topology and noise directly into the observation space.

In ref. [119], the environment state is expressed as hardware-realizable unitaries constructed from the processor's native one- and two-qubit gates along the fixed chip connectivity. Concretely, the environment iteratively generates "percepts" W_ℓ by composing native gates on the actual coupling graph and assigns to each W_ℓ a scalar value V_ℓ equal to the minimal

number of native gates required to invert it, so that the pairs (W_ℓ, V_ℓ) jointly encode both the native gate alphabet and the topology-specific cost structure of the superconducting device.

In qgym [108], the environment state explicitly encodes hardware constraints alongside the circuit, so that agents learn compilation policies conditioned on the target device rather than in a hardware-agnostic setting. In the mapping and routing environments, the state comprises the current logical-to-physical qubit assignment together with the device's coupling graph, exposing which physical qubits are connected and how logical qubits are embedded on that topology at each step. For scheduling, the state is further augmented with gate and time-slot information, and the framework is designed to incorporate additional QPU-specific attributes such as connection fidelities, making hardware topology and resource limitations first-class components of the observation space rather than hidden in the reward.

In [120] the environment state is explicitly hardware-centric as it combines a representation of the logical circuit with a rich description of a multi-chip NISQ device, including chip-wise connectivity, inter-chip links, and time-dependent noise characteristics learned from hardware telemetry. The hardware part of the state encodes per-qubit and per-link error profiles as a temporally structured sequence processed by the dynamic noise adaptation bidirectional long short-term memory [126], so that at each mapping decision the system has access to predicted short-term noise trajectories and can distinguish high-noise from low-noise regions across chips. This is further fused with a graph-based description of logical qubit interactions via multi-head self-attention, yielding a joint state representation in which qubit placement and routing decisions are conditioned simultaneously on logical circuit structure, chip topology, and noise, enabling genuinely noise-adaptive, hardware-aware circuit mapping on multi-chip architectures.

Remarks

- **Perspective on circuit-level encoding.** From a broader perspective, the encoding in CRLQAS [109] occupies a middle ground between explicit tabular encoding and fully convolutional, image-like representations, arranging circuits on a (qubits) × (moments) × (gate classes) grid but employ convolutional layers to exploit spatial structure and to generalize across circuit sizes. Compared to the tabular design in ref. [35], the tensor-based scheme avoids hand-crafted integer fields and is naturally amenable to batched tensor operations; compared to convolutional approaches [36], it retains the same grid abstraction while using a simpler MLP-based architecture, trading some inductive bias toward locality and translation invariance for reduced architectural complexity and implementation overhead.
- **How neural network choice state representation and vice versa?** We note that the choice of state representation and the choice of neural architecture are tightly coupled. Sequence-based and tabular encoding naturally operate on multilayer perceptrons (MLPs) [127] or sequence models [128]; grid-based circuit tensors are well matched to convolutional neural networks (CNNs) [129], graph-based encoding align well with graph neural networks (GNNs) [130]. More recent architectures such as transformers [131] and KANs [132] suggest designing states that represent long-range structure or smooth parameterizations. We explore this briefly in the Sec. 7.
- **(Future work direction) Graph based QAS for RL.** Graph-based quantum architecture search was applied in self-supervised not RL [133]. In [134] the authors introduce a DAG based encoding for unsupervised representation learning. Could be used for RL.

5 Reward engineering

The design of the reward signal [135, 136, 137] is arguably the most consequential architectural decision in applying reinforcement learning to quantum optimization. Unlike supervised learning, where a fixed loss function is externally prescribed, RL agents learn solely from scalar feedback . The reward encodes everything the agent is permitted to know about the quality of its actions. In quantum settings this challenge is compounded by the fact that the natural figure of merit, state fidelity or cost function is expensive to evaluate, non-smooth with respect to gate sequences, and increasingly noisy on real hardware. Consequently, the literature has converged on two broad paradigms. The first uses *energy- or fidelity-based* terminal rewards that directly reflect the physical objective, accepting the cost of sparse feedback. The second constructs *multi-objective* dense rewards that combine the primary quantum figure of merit with structural circuit penalties such as depth, gate count, and/or error probability, trading exact optimality for richer gradient information during learning. The subsections below overview representative designs from each paradigm, tracing the evolution from single-scalar terminal rewards to carefully shaped, multi-component signals.

5.1 Energy/fidelity-based rewards

The final reward in ref. [122] in the task of implementing a QAOA ground state is $r_P = R(E_P)$. The reward is associated with minimizing the final expectation value $E_P = \langle\psi_P|H|\psi_P\rangle$. Here, $R(E_P)$ is monotonically increasing when E_P decreases.

Table 3 Summary of reward functions in RL-based quantum circuit optimization.

Work	Task	Reward formulation	Objectives
Energy/fidelity-based			
Wauters et al. [122]	QAOA	$r_P = -E_P$	Energy only
Patel et al. [109]	Recursive QAOA	$R \in [0, \max_x \langle x\|H_n\|x\rangle]$	Cost function only
Kuo et al. [117]	Circuit search	r=−0.01/step; r=F−0.01 at goal	Fidelity, path length
Bukov et al. [138]	Quantum control	$r(T) = \|\langle\psi^*\|\psi(T)\rangle\|^2$, else 0	Terminal fidelity
Niu et al. [139]	Quantum control	Control cost function	Control fidelity
Multi-objective			
Fösel et al. [36]	QAOA-MaxCut	r_t=$q(s_{t+1})$−$q(s_t)$, q=d+ηn, η=0.2	Depth, gate count
Zen et al. [125]	State prep.	$r_t = d_{t-1} - d_t$ (tableau dist.)	Tableau distance
Ostaszewski et al. [35]	VQE	Eq. (4): threshold ξ, $E_{\min}$ norm.	Energy, episode length
Kundu [140]	Thermal prep.	Eq. (5): E_{term} and Fid_{term}	Free energy, fidelity
Moro et al. [47]	Compiling	Eq. (6)/(7): $(L-t+1)$ at success; $-d(U_t, U)/L$ otherwise	Accuracy, circuit length
Moflic et al. [141]	QAS	Eq. (8): $r_t = \left(\frac{M_{D_t}}{D_{t+1}} + \frac{M_{C_t}}{C_{t+1}}\right)\frac{C_t}{C_{\min}}$	Depth, gate cost, C

For the paper, the authors choose $R(E_P) = -E_P$. For the similar task ref. [123] consider a reward function of the form $R = [0, \max_{x\in 0,1^n} \langle x|H_n|x\rangle]$.

In ref. [117], for each step before successfully reaching the goal, the agent receives a negative reward −0.01 to encourage the shortest path. When reaching the goal, the agent receives a positive reward of value $(F - 0.01)$.

Apart from quantum circuit optimization and construction in ref. [138] the author shows that an RL-agent can learn about the underlying physical system solely through a single scalar reward

$$r(t) = \begin{cases} 0, & \text{if } t < T \\ F_h(T) = |\langle\psi_*|\psi(T)\rangle|^2, & \text{if } t = T, \end{cases} \tag{3}$$

where to manipulate a quantum system, the computer agent constructs piecewise-constant protocols of duration T by choosing a drive protocol strength at each time, calculated from numerical simulations of the physical system. ψ_* is the target state.

Ref. [139] deals with overcoming the fundamental challenges in quantum control by connecting deeper physical knowledge of the underlying quantum dynamics with state-of-the-art RL techniques. Where the authors utilize the universal control cost function itself as the reward.

5.2 Multi-objective rewards

Ref. [36] tackles the optimizaton of QAOA MaxCUT circuits. To drive the RL-agent to its goal, the reward signal is constructed as a dense, stepwise reduction of a scalar circuit cost $q(s)$, which serves as a proxy for the circuit's error probability. Starting from a simplistic noise model where each qubit accumulates decoherence during runtime, and each gate contributes an additional error factor, the success probability is modeled as $P_{\text{success}} \propto \exp(-\mu T) \prod_{k=1}^{n} u_k$, with T the total runtime, n the number of gates, and u_k is the per-gate performance degradation factor. Defining the cost as $q(s) = -\ln P_{\text{success}}$ yields a quantity that increases monotonically with error probability, which under the assumption of uniform idle and gate costs reduces to a linear combination of circuit depth d and gate count n, $q(s) = d + \eta n$, with $\eta = 0.2$ is the choice the authors made in the article. The immediate reward at RL step t is then chosen as the change in this cost,

$$r_t = q(s_{t+1}) - q(s_t), \tag{4}$$

so that maximizing the cumulative return is equivalent to driving the circuit towards lower depth and gate count, while providing the agent with temporally fine-grained feedback rather than a sparse terminal reward. Following the structure of the reward in Equation 4, in ref. [125] utilizes the reward $r_t = d_{t-1} - d_t$ in the task of fault-tolerant logical state preparation, where d is the tablau distance i.e. the distance between the tableau describing the output state of the currently proposed quantum circuit and the tableau of the target state (see Sec. 4.2 for details).

In the scope of finding the ground state of chemical Hamiltonian in Ostaszewski et al. [35], quantum circuit optimization is cast as an RL task where the agent receives a scalar reward based on the current variational energy E_t of the circuit at step t.

The reward is defined as

$$R_t = \begin{cases} 5, & \text{if } E_t < \xi, \\ -5, & \text{if } t \geq L \text{ and } E_t \geq \xi, \\ \max\left(\dfrac{E_{t-1} - E_t}{E_{t-1} - E_{\min}}, -1\right), & \text{otherwise,} \end{cases} \tag{5}$$

where ξ is a predefined energy threshold, L is the maximum episode length, and $E_{\min}$ is a reference minimum energy used for normalization. The instantaneous energy is evaluated as

$$E_t = \langle \psi(\vec{\theta}) | H | \psi(\vec{\theta}) \rangle,$$

with $\vec{\theta}$ denoting the rotation angles of the parametrized ansatz. Later ref. [109, 142, 143] shows the further applicability of the reward function towards large-scale quantum problems.

In [140], the reward utilized for thermal state preparation is refined from a free-energy-only objective to a multi-objective signal that also accounts for thermal-state fidelity. The basic reward uses the Helmholtz free energy F_t which is fundamentally based on the reward function in Equation 5 but is extended to

$$r_t = \begin{cases} 5, & F_t \leq \zeta_F \text{ and } \mathrm{Fid}(\rho(\vec{\phi})) \geq \zeta_{\mathrm{Fid}}, \\ -5, & \mathrm{Fid}(\rho(\vec{\phi})) < \zeta_{\mathrm{Fid}} \text{ and } t = D_{\max}, \\ 0.6\, E_{\mathrm{term}} + 0.4\, \mathrm{Fid}_{\mathrm{term}}, & \text{otherwise,} \end{cases} \tag{6}$$

where $E_{\mathrm{term}} = \max\left(\frac{F_{\mathrm{prev}} - F_{\mathrm{current}}}{|F_{\mathrm{prev}} - F_{\mathrm{true}}|}, -1, 1\right)$, and $\mathrm{Fid}_{\mathrm{term}} = 2\,\mathrm{Fid}(\rho(\vec{\phi})) - 1$. Thus, the reward explicitly balances free-energy minimization and state fidelity, rather than optimizing only a single thermodynamic objective.

To tackle the task of quantum compiling in Moro et al. [47], the reward is formulated with multi-objective that jointly encodes approximation accuracy and circuit length. For quasi-continuous bases of small rotations, they employ a dense reward

$$r(S_n, a_n) = \begin{cases} (L - t) + 1, & \text{if } d(U_t, U) < \epsilon, \\ -d(U_t, \mathcal{U})/L, & \text{otherwise,} \end{cases} \tag{7}$$

where $\mathcal{U}$ is the target unitary, U_t the current approximation, $d(\cdot, \cdot)$ a distance measure based on average gate fidelity, L the maximum episode length, and the term $(L - t + 1)$ penalizes long sequences. For the discrete gate base, they instead use a sparse, binary reward

$$r(S_t, a_t) = \begin{cases} 0, & \text{if } d(U_t, U) \leq \epsilon, \\ -\dfrac{1}{L}, & \text{otherwise,} \end{cases} \tag{8}$$

with tolerance ϵ and the same per-step length penalty. In both formulations, the reward simultaneously enforces high fidelity to the target and short gate sequences, making it explicitly multi-objective.

More recently Moflic et. al. [141] introduces QASER, an exponential reward function that is denser than that of commonly used reward structures. The reward at each step simultaneously optimizes the cost function C_t, the depth of circuit D_t and the cost of circuit C_t where C_t is defined by $C_t = \dfrac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$ where w_i is the assigned weight of each of the n gate class and x_i is the number of gate in the circuit at time step t. The reward function looks like:

$$r_t = \left(\frac{\mathbb{M}_{D_t}}{D_t + 1} + \frac{\mathbb{M}_{C_t}}{C_t + 1}\right)^{\dfrac{C_t}{C_{\min}}}, \tag{9}$$

where $\mathbb{M}_{D_t}$ and $\mathbb{M}_{C_t}$ are initially set as hyperparameters but during training these are replaced by the maximum depth and gate cost encountered along a learning trajectory up to time step t. $C_{\min}$ is the minimum value of the cost function.

Remarks

- **Reward sparsity and credit assignment.** The most informative reward, i.e. terminal fidelity or energy signal, is also the hardest for an agent to learn from, because a single scalar at the end of a long episode provides no intermediate

guidance. Dense reformulations (e.g., stepwise Δq as in [36], or incremental energy improvement as in [35]) alleviate this but introduce a design choice: the shaping term must not alter the optimal policy [102] which is a condition that is non-trivial to verify in quantum settings where the circuit space is exponentially large.

- **Proxy metrics and their validity.** Several works collapse the multi-objective reward into a single linear combination of depth and gate count, motivated by simple decoherence models (see [36]). While tractable, this conflates circuit length with circuit noise, which need not be proportional on real hardware where two-qubit gate error rates, qubit connectivity, and idle-time decoherence vary spatially. More principled proxies, such as the weighted gate cost C_t in QASER [141], begin to address this but still rely on hardware-agnostic weights set before training.
- **(Future work direction) Noise-aware reward functions.** An important gap in the existing literature is the absence of reward functions that directly incorporate a *hardware-calibrated noise model* into the reward computation. All existing multi-objective rewards treat noise only implicitly, as a monotone function of depth or gate count. A principled noise-aware reward would take the form

$$r_t \;=\; r_{\text{fidelity}}(s_t) \;-\; \lambda\, \mathcal{N}(c_t), \tag{10}$$

  where $\mathcal{N}(c_t)$ is a noise penalty evaluated on the current partial circuit c_t using a device-specific noise map such as per-gate depolarizing rates or T_1/T_2 profiles, and $\lambda > 0$ controls the noise-fidelity trade-off. Such a reward would allow the agent to distinguish between two circuits of equal depth that differ in their noise susceptibility. For instance preferring circuits that avoid high-error two-qubit gates or heavily decoherent qubits. Designing $\mathcal{N}$ to remain differentiable with respect to gate selection, and ensuring that the combined reward still satisfies potential-based shaping conditions, constitute open research questions. Moreover, the reliability-adjusted prioritization introduced in the ReaPER+ replay buffer [144] offers a complementary mechanism: rather than encoding noise only in the reward, the buffer itself can up-weight transitions collected under low-noise conditions, effectively performing noise-aware credit assignment at the data level. Unifying these two perspectives, reward-level and buffer-level noise awareness, represents a promising direction for hardware-efficient quantum circuit optimization with RL.
- **(Future work direction) Amortized and training-free proxies as reward.** Two strategies reduce the cost of evaluating the reward signal itself. In ref. [144] introduced ammortized curriculum RL that accumulates circuit edits across several steps before querying the simulator. *Training-free proxies* replace simulation entirely: He et al. [145] show that an adaptive mixture of structural scores, covering trainability, expressivity, and hardware robustness, outperforms single-proxy and fully trained QAS baselines, effectively using the proxy *as* the reward. A natural future direction is to periodically recalibrate proxy weights using amortized simulation, keeping the training-free signal aligned with the true quantum objective as the circuit distribution evolves during training.

6 Action space design

The action space is the combinatorial vocabulary from which an RL agent assembles or rewrites a quantum circuit. Its design sits at the intersection of two competing objectives that are well-recognized in the RL literature:*expressivity* how large a family of circuits the agent can in principle construct and *sample efficiency* how quickly a policy can be learned within that space [146, 147]. In discrete control problems, an overly large action space exponentially inflates the number of state i.e. the action pairs the agent must visit before forming reliable value estimates, a phenomenon studied formally in the context of combinatorial action spaces by Dulac-Arnold et al. [146] and Van de Wiele et al. [148]. Conversely, an action space that is too coarse forfeits the ability to represent the optimal solution, a failure mode closely related to *action aliasing* in factored MDPs [149].

These tensions are especially acute in RL-QCO. The circuit space is discrete and combinatorially vast: even for $n = 10$ qubits and a gate vocabulary of modest size, the number of distinct circuits of depth d scales as $|\mathcal{A}|^d$ (cf. Eq. (11)), far exceeding the sample budgets available when each reward evaluation requires a quantum simulation or hardware query. This motivates *action space shaping* [150, 151], the systematic pruning or structuring of $\mathcal{A}$ to focus the agent's search on physically meaningful regions of circuit space, analogously to how reward shaping [102] refocuses the agent's gradient signal without altering the optimal policy. Eliminating provably suboptimal or hardware-invalid actions has been shown in classical settings to obtain order of magnitude reductions in sample complexity with no loss in asymptotic performance [151, 147], and the same principle drives the most effective action space designs in RL-QCO. In RL-QCO, three principal design axes appear repeatedly in the literature:

1. the *granularity* of the primitive actions (individual gates versus composite macros, Sec. 6.1-6.2);
2. the *type* of operation (construction, rewriting, or layout, Sec. 6.2 -6.4); and

3. whether hardware constraints are *baked directly* into the action vocabulary, restricting $\mathcal{A}$ to naively realizable operations (Sec. 6.4).

6.1 Low-level gate actions

The simplest choice is to define one action per elementary gate: the agent picks a gate type, a qubit (or qubit pair), and, for parametrized gates, either a fixed rotation angle or a flag that triggers a classical sub-optimizer. Concretely, for a system of n qubits with a gate vocabulary $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$, where $\mathcal{G}_1$ and $\mathcal{G}_2$ denote the sets of 1- and 2-qubit gates respectively, the flat action space is

$$\mathcal{A} \;=\; \big(\mathcal{G}_1 \times [n]\big) \;\cup\; \big(\mathcal{G}_2 \times [n] \times [n]\big), \quad |\mathcal{A}| = |\mathcal{G}_1|\, n \;+\; |\mathcal{G}_2|\, n(n-1), \tag{11}$$

where $[n] = \{0, 1, \ldots, n-1\}$ indexes physical qubits and the second factor of $\mathcal{G}_2$ excludes self-loops (control $\neq$ target). Ostaszewski et al. [35] use exactly this scheme with $\mathcal{G}_1 = \{R_x, R_y, R_z\}$ and $\mathcal{G}_2 = \{\text{CNOT}\}$: at each step the agent appends a single gate drawn from $\mathcal{A}$ to the growing circuit, with gate parameters handled by COBYLA or Rotosolve after each insertion. Fösel et al. [36] and CRLQAS [109] similarly treat each possible (gate class, qubit index, moment) triple as one discrete action, masking out transitions that would violate circuit-depth or connectivity constraints; their effective action space is a topology-filtered subset

$$\mathcal{A}_{\text{valid}} \;\subseteq\; \mathcal{G}_2 \times \mathcal{E}, \tag{12}$$

where $\mathcal{E} \subseteq [n] \times [n]$ is the edge set of the device coupling graph, so that only natively connected qubit pairs are ever targeted by two-qubit gates.

Kölle et al. [152] extend this paradigm to the Clifford+T gate set $\mathcal{G} = \{H, S, T, \text{CNOT}, \ldots\}$ in a state-preparation benchmark where the agent constructs circuits for Bell and GHZ states using PPO. Their environment exposes every valid (gate, qubit-target) combination as a flat action, providing a controlled testbed for studying how the combinatorial explosion of $|\mathcal{A}|$ with qubit count translates directly into sample complexity growth and convergence difficulty in practice.

Low-level actions keep the action space minimal, but $|\mathcal{A}|$ grows as $O(|\mathcal{G}|\, n^2)$, and the number of steps required to build even a moderate-depth circuit grows linearly with circuit depth, jointly putting pressure on exploration and sample efficiency.

6.2 Higher-level macros and structured action spaces

A widely adopted remedy is to replace single-gate primitives with *gadgets* [153, 154] which are the composite operations that place multiple gates in a single step. The benefit is a dramatic reduction in episode length and a natural injection of domain knowledge about which gate patterns are physically meaningful. A common macro type in VQE ansatz search is the *entangling layer*: a fixed pattern of CNOT or CZ gates acting on all pairs of neighboring qubits, followed by a layer of single-qubit rotations. By treating such a block as one action, the agent works at the layer level rather than the gate level, reducing episode length from $O(d \cdot n)$ to $O(d)$ for an n-qubit, d-layer circuit.

An orthogonal family of macros arises in circuit *simplification* rather than construction. Riu et al. [101] define the action space as the set of valid ZX-calculus rewrite rules (spider fusion, π-copy, Hadamard cancellation, etc.) together with the node in the ZX-diagram on which to apply each rule. Each action is therefore a (rule, node) pair, and applying it transforms the current ZX-graph into a semantically equivalent but structurally simpler one. Because ZX rewrites are provably correctness-preserving by construction, the agent never needs to verify circuit equivalence explicitly, and the combinatorial overhead of invalid actions is eliminated. Koziell-Pipe et al. [112] and Mattick et al. [113] follow the same paradigm, pairing ZX-graph states with GNN policies that output distributions over rewrite locations.

A distinct and practically important category positions *compiler optimization passes* as actions rather than individual gates or rewrite rules. Mills et al. [155] train a PPO agent whose action space consists of seven PyTKET [156] optimization passes (e.g., `KAKDecomposition`, `CliffordResynthesis`, `ZXGraphlikeoptimization`) together with a `DoNothing` termination action. The agent, conditioned on a GNN representation of the current circuit, selects which pass to apply next, solving the well-known *phase-ordering problem*: no single fixed pass sequence is optimal across diverse circuit families. On an in-distribution test set of over 40,000 circuits, the RL agent achieves a mean two-qubit gate removal rate of 57.7%, versus 41.8% for the best fixed PyTKET default sequence, and generalizes to out-of-distribution circuits up to 100 qubits. This paradigm represents a form of *action abstraction at the compiler level*: instead of deciding gate by gate how to transform a circuit, the agent decides optimization-strategy-by-strategy, delegating the low-level mechanics to well-engineered classical subroutines.

For variational quantum eigensolver (VQE) tasks targeting molecular Hamiltonians, the most expressive macros are *fermionic excitation operators*. A single and double excitation operator,

$$\hat{U}_{pq}(\theta) = e^{\theta(\hat{a}_p^\dagger \hat{a}_q - \text{h.c.})}, \qquad \hat{U}_{pqrs}(\theta) = e^{\theta(\hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_r \hat{a}_s - \text{h.c.})}, \tag{13}$$

act on the fermionic Fock space and are directly motivated by the unitary coupled-cluster (UCC) ansatz. When such operators are pre-compiled into fixed gate sequences (e.g. via ADAPT-VQE templates [50]) and offered as actions, the agent searches over chemically meaningful circuit structures rather than arbitrary gate combinations. Trotterized time-evolution operators, $e^{-i\Delta t \hat{H}_k}$ for each term $\hat{H}_k$ in the Hamiltonian, serve an analogous role in quantum simulation tasks: a single action deposits a Trotter step, compressing what would be dozens of gates into one decision. Yao et al. [121] use a related idea in their counterdiabatic-inspired RL agent, where each action selects a control generator from a fixed set of physically motivated operators. Wang and Mazziotti [157] extend this operator-selection paradigm to many-body simulation via the contracted Schrödinger equation (CSE) [158] which at each step allows the agent to selects a two-body exponential operator guided by the CSE residual. This yields compact, physically informed ansatz circuits for strongly correlated systems.

The most flexible macro paradigm allows the action vocabulary itself to evolve during training. Kundu and Sarra [143] introduce *gadget reinforcement learning* (GRL), in which a program synthesis subroutine automatically composes composite gates from building blocks encountered on easier problem instances, then incorporates these gadgets as new atomic actions for harder tasks. Unlike a pre-fixed macro library, the GRL vocabulary is data-driven and grows to capture problem-specific structure: on VQE and Ising benchmarks, learned gadgets reduce circuit depth and improve success rates beyond what either single-gate or hand-crafted macro vocabularies achieve, while transferring efficiently to real hardware via device-compatible decompositions. AlphaTensor-Quantum [124] implicitly follows an analogous strategy at the tensor-decomposition level: gadget bonuses in the reward incentivize the agent to discover factorizations that correspond to known efficient circuit identities, effectively growing an implicit library of useful multi-gate patterns through the optimization process itself.

Comparison of abstraction levels. Table 4 summarizes representative action space designs across the literature.

Table 4 Representative action space designs in RL-based quantum circuit optimization. "D" = discrete, "H" = hybrid discrete-continuous.

Work	Task	Action granularity	Action type	D/H
Ostaszewski et al. [35]	VQE ansatz	Single gate	Gate insertion	D
Fösel et al. [36]	QAOA opt.	Single gate	Gate insertion / deletion	D
CRLQAS [109]	QAS under noise	Single gate / block	Gate insertion	D
Kölle et al. [152]	State synthesis	Single gate (C+T)	Gate insertion	D
Riu et al. [101]	Circuit simp.	Rewrite rule	ZX rewrite	D
Mattick et al. [113]	ZX simplification	Rewrite rule	ZX rewrite + GNN policy	D
Yao et al. [121]	Ground-state prep.	Control generator	Operator selection	D
Wang & Mazziotti [157]	Many-body sim.	Two-body operator	CSE-residual selection	D
Mills et al. [155]	Circuit opt.	Compiler pass	Pass sequence (meta-action)	D
GRL [143]	VQE / Ising	Learned gadget	Synthesized macro	D
HyRLQAS [159]	VQE ansatz	Gate + angles	Joint placement & param.	H
Preti et al. [160]	Trapped-ion comp.	Gate ordering	Order + gradient param.	H
Kremer et al. [116]	Compilation	Synthesis / routing	Gate synthesis + SWAP	D
qgym [108]	Compilation	Mapping / routing	Qubit assignment + SWAP	D
Wang et al. [119]	Compilation	Native gate	Hardware-native gate	D
AlphaTensor-Q [124]	T-count opt.	Rank-1 factor	Tensor factorization	D
Van Veen et al. [161]	DQC routing	Qubit-pair op.	Direct pair + masking	D

6.3 Expressivity vs. sample efficiency

The central tension in action space design is between *expressivity* which is the set of circuits reachable within a fixed episode budget and *sample efficiency* which is described by the number of environment interactions needed to learn a good policy. As summarized in Sec. 6.1 and in Sec. 6.2, flat gate-level spaces are maximally expressive but impose an $O(|\mathcal{G}| \cdot n^2)$ branching factor at each step, while macro-based spaces trade reachability for drastically shorter episodes. Both families typically rely on *action masking* to reduce the effective branching factor without restricting the reachable circuit family: at each step, actions inconsistent with the current circuit state or hardware constraints have their policy logits zeroed before the softmax [36, 109].

When rotation angles (which are continuous in nature) are included in the action (usually discrete), the space becomes continuous or mixed discrete-continuous. Moro et al. [47] use a quasi-continuous base of small rotation increments, effectively discretizing the rotation group at fine resolution. The most common strategy in QAS works, however, is to keep the action space purely discrete (gate type and qubit indices) and delegate angle optimization to a classical sub-routine (with COBYLA, Rotosolve, or gradient descent) invoked between RL steps. This decoupling is computationally efficient but

creates a non-stationarity: the variational energy observed after each action depends on the sub-optimizer's convergence, introducing noise into the reward signal.

HyRLQAS [159] directly addresses this decoupling limitation by proposing a *hybrid action space* that simultaneously chooses discrete gate types and placements and continuous rotation angles. In each step, the agent selects a gate and its target qubit(s) via a discrete head, and simultaneously outputs and refines continuous parameter values for both the new gate and previously placed gates via a continuous head. Tested on molecular VQE benchmarks, HyRLQAS consistently achieves lower energy errors and shorter circuits than discrete-only and continuous-only baselines, demonstrating that joint discrete-continuous learning can outperform the conventional two-stage (structure search, then angle optimization) pipeline. Preti et al. [160] apply a related hybrid strategy to trapped-ion compilation: a projective-simulation RL agent learns gate orderings (discrete) while a gradient-based optimizer handles rotation angles (continuous), with the two stages alternating within each episode.

6.4 Encoding hardware knowledge in the action space

A key advantage of RL over black-box optimizers is that hardware constraints can be enforced *structurally* rather than via penalty terms in the reward. This is achieved by restricting the available actions to those that are physically realizable on the target device.

Real quantum processors expose only a sparse subset of all qubit pairs as natively entangling. Figure 4 illustrates the heavy-hex coupling graph of the IBM `ibm_nazca` 133-qubit device: each physical qubit q_i is connected to at most three neighbors, and two-qubit gates are only valid on edges (q_i, q_j) present in the graph.

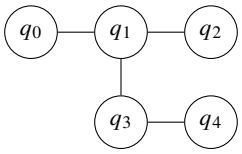

Figure 4 Fragment of an IBM heavy-hex coupling graph (5 qubits). Edges indicate natively supported CZ/ECR interactions. The corresponding two-qubit action set is $\mathcal{A}_{2Q} = \{(0,1),(1,2),(1,3),(3,4)\}$, excluding all non-adjacent pairs.

When the RL action space is defined over physical qubit pairs, restricting $\mathcal{A}_{2Q}$ to edges in the coupling graph immediately rules out any circuit that requires non-local 2-qubit gates, eliminating the need for SWAP insertion in the inner loop. Wang et al. [119] implement this by generating "percepts" exclusively from the native gate set of a superconducting processor, so the agent never encounters an action that is physically invalid. qgym [108] takes a complementary approach: the routing environment exposes both the logical circuit and the device coupling graph as part of the state, and SWAP gates are included explicitly in the action set, allowing the agent to learn a topology-aware routing policy from scratch. DeepQMap [120] further augments the action space with noise-weighted priorities, so that actions placing gates on high-fidelity links are implicitly preferred via the learned value function.

Van Veen et al. [161] extend hardware-aware action design to *distributed quantum computing* (DQC), where qubits are spread across multiple interconnected processor modules. Their key contribution is a novel action-space formulation that allows direct qubit-pair operations across module boundaries, rather than restricting the agent to neighboring intra-chip pairs, combined with action masking to prevent physically invalid inter-module requests. Benchmarked on random circuits across hardware graphs with differing connectivity (linear chains, 2D grids, and complex topologies), the agent achieves up to a 35% relative reduction in modeled circuit execution time compared to the previous best RL baseline [162], with 30% less wall-clock training time, demonstrating that action-space engineering is a first-class lever for scalable DQC compilation.

Each QPU exposes a *basis gate set* to which all circuits must be compiled before execution. IBM devices, for example, currently use the basis $\{\mathrm{ECR}, R_z(\theta), \sqrt{X}, X\}$, where ECR (echoed cross-resonance) is the native entangling gate. If the RL action vocabulary is defined directly over this basis, as in Wang et al. [119] and Kremer et al. [116], the output circuit requires no further compilation: actions are immediately executable, and the reward (circuit fidelity or energy) can be evaluated without a transpilation pass. Conversely, action spaces that include logical gates such as CNOT or CZ require an intermediate decomposition step, which may add additional depth and degrade the correspondence between the RL objective and hardware cost. Table 5 summarizes how representative works align their action vocabularies with physical device constraints.

Remarks

- **General remarks.** The action space design choices discussed above lie on a spectrum from maximally flexible (arbitrary single gates, hardware-agnostic) to maximally constrained (native-basis macros, topology-filtered). Neither

Table 5 Hardware awareness in RL action space design. ✓ = explicitly modeled, ✗ = hardware-agnostic, ● = constraints in environment but not in the action set.

Work	Topology-aware	Native gate set	Mechanism
Wang et al. [119]	✓	IBM superconducting	Percept generation on coupling graph
qgym [108]	✓	Configurable	Coupling graph in state + SWAP actions
DeepQMap [120]	✓	Multi-chip NISQ	Noise-weighted topology encoding
Kremer et al. [116]	✓	IBM (ECR, RZ, $\sqrt{X}$)	Direct native-gate synthesis
Van Veen et al. [161]	✓	Multi-module DQC	Novel qubit-pair formulation + masking
Mills et al. [155]	●	Quantinuum native	PyTKET passes rebased after each action
CRLQAS [109]	●	CNOT, RY, RX, RZ	Noise injected in environment, not action
Fösel et al. [36]	✗	Logical	Topology implicit in reward
Ostaszewski et al. [35]	✗	CNOT, RY, RX, RZ	Hardware-agnostic

extreme is universally preferable: hardware-agnostic spaces permit richer policy transfer across devices [109], while hardware-native spaces reduce post-processing overhead and ensure that reward signals faithfully reflect execution cost. The most effective strategies in recent work thread this needle by using a *two-level* action hierarchy; coarse macros for architecture search, fine-grained gates for local optimization, and enforcing topology constraints via action masking rather than reward penalties.

- **(Future work direction) RL with action chunking.** Reinforcement learning with action chunking [163] proposes that agents commit to *sequences* of actions at once, executing a chunk of k gates before re-querying the policy. In quantum settings this maps naturally onto the idea of Trotter steps or variational layers as atomic decisions, and preliminary evidence from imitation-learning-based quantum control suggests that chunked policies generalize better across problem sizes than step-wise policies [143]. A promising open question is how to choose the chunk size adaptively, balancing exploration width (small k) against episode-length reduction (large k).
- **(Future work direction) RL with action shaping.** Action space shaping [150] removes irrelevant or harmful actions from the vocabulary during training, analogously to how reward shaping modifies the feedback signal. In RL-QCO, a natural form of shaping is to prune gates whose insertion is guaranteed to increase circuit cost (e.g. adding a CNOT that immediately cancels with an adjacent CNOT), or to bias sampling toward actions that respect symmetries of the target Hamiltonian. Symmetry-aware action masking [164] has been explored in classical combinatorial optimization and represents a largely untapped direction for quantum circuit RL.
- **(Future work direction) Adaptive and learned action vocabularies.** Current methods fix the action vocabulary before training begins. GRL [143] and the gadget-bonus mechanism in AlphaTensor-Quantum [124] demonstrate that synthesizing or discovering new composite actions mid-training substantially improves both circuit quality and sample efficiency. A natural next step is *automated vocabulary expansion* triggered by training dynamics. For instance, detecting when exploration stagnates and synthesizing new macros from frequently co-occurring gate sub-sequences. Combined with the hybrid discrete-continuous framework of HyRLQAS [159], such an adaptive vocabulary could simultaneously optimize circuit structure, gate parameters, and the action language itself in a unified RL loop.
- **(Future work direction) Multi-level action spaces for modular architectures.** As quantum systems scale toward distributed multi-chip designs, the action space must encode different *levels* of operation: intra-chip gate operations, qubit routing within a chip, and inter-chip entanglement generation. Van Veen et al. [161] introduce direct qubit-pair operations for DQC routing, but a truly hierarchical framework that plans at all three levels simultaneously, architecture search, intra-chip routing, and inter-chip communication, within a single RL agent does not yet exist and represents a pressing open challenge for fault-tolerant and distributed quantum compilation.

7 Agents and learning algorithms

The choice of RL agent such as algorithm class, neural architecture, and hyperparameter regime is a primary and crucial design decision in RL for quantum circuit design. In classical settings, identical algorithms can converge or fail entirely based on learning rate, buffer capacity, and network depth [104, 165]. These sensitivities are amplified in RL-QCO: each interaction requires a quantum simulation or hardware execution, rewards are noisier (shot statistics, gate fidelity, sub-optimizer convergence), and the state space encodes combinatorial circuit structures rather than continuous observations.

The most systematic study is BenchRL-QAS [166], which evaluates nine RL agents across VQE, VQSD, VQC, and GHZ state preparation on 2 to 8-qubit systems under noiseless and noisy conditions. Its central finding is the confirmation that

RL-QCO also follows the no free lunch principle [167] i.e that no single agent dominates across tasks, qubit counts, and noise regimes. Table 6 summarizes algorithm usage across the literature.

Table 6 RL algorithms used across representative RL-QAS works. Compiled from the literature; see also [166].

Work	RL algorithm(s)
Ostaszewski et al. [35]; Patel et al. [109]; Kundu et al. [62]; Kundu and Mangini [142]; Kundu and Sarra [143]	DDQN
Ye and Chen [37]	PPR-DQN
Olle et al. [168]; Foderà et al. [169]; Mills et al. [155]	PPO
Kuo, Fang, and Chen [117]; Fösel et al. [36]	A2C, PPO
Lockwood [170]	TD3, SAC, PPO
Patel et al. [123]	REINFORCE
Altmann et al. [107]	A2C, PPO, SAC, TD3
Zhu and Hou [171]	A2C, PPO, TRPO
Dutta et al. [172]	A2C, PPO, DDQN, TD3
Stenzel et al. [173]	PPO, A2C
BenchRL-QAS [166]	A2C, A3C; DQN-family (DQN, DQN_{PER}, DQN_{rank}, Dueling DQN, DDQN); PPO, TPPO

7.1 Value-based methods

Value-based RL agents learn a state-action value function $Q(s, a; \theta)$, parameterized by a neural network with weights θ, and derive a policy by greedily selecting $a^* = \arg\max_a Q(s, a; \theta)$. The original deep Q-network (DQN) of Mnih et al. [174] stabilizes training through a target network (updated periodically rather than at every step) and an experience replay buffer that breaks temporal correlations in the training stream. Double DQN (DDQN) [88] corrects the overestimation bias of DQN by decoupling action selection from action evaluation:

$$y_t \;=\; r_t + \gamma\, Q\Big(s_{t+1},\, \arg\max_{a'} Q(s_{t+1}, a'; \theta);\, \theta^-\Big), \tag{14}$$

where θ^- denotes the target network parameters. Dueling DQN [175] decomposes the Q-value into a state-value stream $V(s; \theta)$ and an advantage stream $A(s, a; \theta)$, improving the stability of value estimates in states where the action choice has little impact on the immediate outcome–a common situation mid-episode in circuit construction, where many gate placements are equally sub-optimal.

Among the value-based methods DDQN in particular, are by far the most widely deployed agent class in RL-QAS. Ostaszewski et al. [35] introduce the first DDQN-based VQE ansatz search algorithm, coupling the agent with COBYLA angle optimization invoked after each gate insertion. CRLQAS [109] retains DDQN as its backbone but adds curriculum-driven episode halting and an illegal-action masking mechanism that assigns $Q(s, a) = -\infty$ to gate placements violating topology or redundancy constraints. The same DDQN backbone underpins the RL-VQSD framework [62] for diagonalizing arbitrary quantum states.

Ye and Chen [37] introduce the Probabilistic Policy Reuse DQN (PPR-DQN), which augments standard deep Q-learning with a continual learning mechanism: when the device noise model changes, rather than retraining from scratch, the agent probabilistically reuses its prior policy, dramatically reducing the sample cost of adapting to new hardware configurations. This is an early demonstration that off-policy value function reuse transfers effectively across quantum environments.

At the opposite end, Giordano et al. [176] show that *tabular* Q-learning with a hybrid circuit-aware reward i.e. a static fidelity term addition to dynamic penalties for gate congestion and redundant revisits consistently finds minimal-depth circuits for graph-state preparation on up to 7-qubit. This provides a useful lower bound: tabular methods match deep Q-networks for small Hilbert spaces, suggesting the primary advantage of neural Q-functions in RL-QCO is generalization across circuit sizes.

In Patel et al. [123] the authors apply the REINFORCE policy gradient to RL-assisted recursive QAOA, where the agent learns a sequence of problem-reduction steps. Altmann et al. [107] systematically study A2C, PPO, SAC, and TD3 across quantum circuit design settings, identifying barren-plateau-like gradient suppression in the policy landscape as a dominant failure mode for continuous-action-space agents on deep circuits. Dutta et al. [172] explore A2C, PPO, DDQN, and TD3 for curriculum RL-QAS in tensor-network circuits.

Table 7 Neural architectures for the RL policy/value network in RL-QCO.

Arch.	Works	Encoding	Key property
MLP	[109, 166]	Flat tensor	Fast, widely used
CNN (2D)	[36]	Gate–qubit grid	Local patterns, size generalization
CNN (3D)	[140]	Depth × qubit × gate	Thermal states, hardware transfer
KAN	[181, 182, 183, 184, 185]	Flat tensor	Interpretable, compact, quantum-inspired activations
GNN	[112, 113, 155]	Circuit DAG	Variable-size circuits, local structure
GNN+Transf.	[186]	DAG + token sequence	Joint local and global context
Transformer	[187, 186]	Gate sequence with 2D positions	Efficient global attention

7.2 Model-based and planning approaches

Model-free RL is the dominant paradigm in RL-QCO. In this framework the agent directly learns from the environment interactions. Several works explore this paradigm under RL-QCO which are elaborated below.

Rapp et al. [114] apply MuZero [115], a model-based RL algorithm that learns a latent dynamics model and uses Monte Carlo Tree Search (MCTS) for planning, to the problem of designing encoding circuits for quantum machine learning. By predicting the reward and next latent state for hypothetical gate sequences *without executing them* in the quantum simulator, the agent looks ahead several steps before committing to a gate placement–achieving competitive circuit quality with significantly fewer simulator calls than DDQN-based baselines on small-scale QML benchmarks.

Meanwhile in Tang et al. [177] combine a neural network policy with Monte Carlo Tree Search for quantum circuit routing (*AlphaRouter*): the neural prior focuses tree expansion on promising gate sub-sequences, analogously to AlphaGo in game playing [178], making planning tractable where pure MCTS would be intractable due to exponential branching. Wang et al. [179] propose a nested MCTS algorithm for automated circuit design that requires no neural prior, providing a search-only baseline that is competitive for small circuits but does not scale beyond 6-7 qubits without neural guidance.

Google Quantum AI and DeepMind [180] demonstrate a qualitatively different application of RL to quantum hardware control: rather than designing circuits offline, the RL agent continuously calibrates physical control parameters of a superconducting processor *during* active computation, using syndrome detection events from the surface code as the reward signal. Tested on the Willow processor, the agent improves logical error rate stability 3.5-fold against injected parameter drift, with simulations of distance-15 surface codes showing that the optimization speed does not degrade with system size. Although this is not directly related to circuit design, it establishes that RL agents operating on the same gate-level decision vocabulary as RL-QCO can extend their utility to the full hardware stack i.e. from ansatz construction down to real time recalibration. Which points toward a unified RL control loop for fault-tolerant quantum computing.

7.3 Neural architectures for circuit optimization

Five neural network families appear in RL-QCO literature, each with a distinct inductive bias over the circuit state (Table 7).

MLPs. Fully connected MLPs remain the most widely deployed backbone. BenchRL-QAS [166] standardizes all nine agents on an MLP with L layers of 1000 neurons fed a flattened tensor of shape $[D_{\max} \times (N{+}3) \times N]$. Despite their simplicity, MLPs are competitive across all tasks, suggesting the bottleneck in RL-QCO lies in the reward signal and action space rather than representational capacity.

CNNs. When the circuit is encoded as a spatial grid, convolutions capture local gate patterns invisible to flat MLPs. Fösel et al. [36] encode a 2D gate-qubit grid and achieve 27% depth and 15% gate count reductions on 12-qubit circuits, with zero-shot generalization to larger circuits via translation equivariance. Kundu [140] extends this to a 3D tensor (depth×qubit×gate type) for SYK thermal state preparation: the 3D-CNN agent reduces CNOT count by two orders of magnitude versus first-order Trotterization for $N \geq 12$ Majorana fermions, and transfers directly to noisy `IBM Eagle r3` hardware without retraining.

KANs. KANQAS [181] replaces the DDQN MLP with a Kolmogorov-Arnold Network [132], where each weight is a learnable univariate spline. On two to four qubit benchmarks it achieves 2-5 times higher success probability than MLP baselines, with superior fidelity under depolarizing noise, at the cost of slower episodes. The broader KAN-for-quantum family reinforces this appeal: QKAN [182] builds a fault-tolerant block-encoding framework demonstrating efficient multivariate state preparation; Jiang et al. [183] replace classical splines with quantum variational activation functions (QVAFs) via data-reuploading circuits, establishing quantum advantages in approximation accuracy; QKAN-LSTM [184] integrates DARUAN modules into LSTM gating, reducing trainable parameters by 79% versus standard LSTMs; and Gated QKAN-FWP [185] extends this to scalable sequence learning. Collectively, these works position KAN-style representations as a general-purpose backbone for quantum-aware RL policies.

GNNs. A quantum circuit is naturally a DAG: nodes are gates/qubits, edges encode temporal order and qubit-wire connectivity. GNNs propagate messages along edges, attending to local patterns invisible to flat encodings. Three settings: *(i)* ZX-calculus rewriting, where Koziell-Pipe et al. [112] and Mattick et al. [113] train GNN policies over ZX-diagram graphs, generalizing to variable circuit sizes; *(ii)* compiler pass sequencing, where Mills et al. [155] use a GINConv actor-critic with 8-dim gate nodes and 4-dim qubit-role edges, achieving 57.7% two-qubit gate removal; and *(iii)* modular compilation, where QARMA [186] report 97-100% inter-core reduction.

Transformers. Self-attention directly relates distant gates on the same qubit wire, avoiding the many message-passing rounds a GNN needs for long-range dependencies. Mathematically, transformers can be viewed as message-passing GNNs on fully connected token graphs, where self-attention implements learned edge weights and positional encodings provide structural hints [187]. QARMA [186] exploits this by pairing a transformer encoder with a GNN: the GNN captures local two-qubit structure, while the transformer attends globally over the gate sequence, outperforming GNN-only baselines for qubit mapping in modular architectures. The main open issue is positional encoding, since quantum circuits live on a two-dimensional grid (time step by qubit index) rather than a simple 1D sequence.

Remarks

- **No free lunch in RL-QCO.** BenchRL-QAS [166] provides the first large-scale confirmation that the NFL theorem [167] applies to RL-based quantum circuit design: no agent class, architecture, or algorithm dominates across tasks, qubit counts, and noise conditions. Unlike classical benchmarks where PPO has become a near-universal default, the additional structure imposed by quantum physics (noise model, Hamiltonian symmetries, hardware topology) fragments the performance landscape, requiring problem-specific algorithm selection.
- **Beyond vanilla DQN and actor-critic.** Classical deep RL shows that combining double Q-learning, dueling heads, prioritized replay, distributional value functions, and noisy nets into a unified Rainbow agent [188] yields substantial gains in data efficiency and final performance. The BRO framework (Bigger, Regularized, Optimistic) further demonstrates that scaling critic capacity with strong regularization and optimistic exploration achieves state-of-the-art continuous control with high sample efficiency [189]. Adapting Rainbow-style stacks and BRO-style large regularized critics to RL-QCO is an open direction, as the quantum setting naturally benefits from both improved sample efficiency (due to expensive environment steps) and more expressive value functions for highly structured reward landscapes.
- **(Future work direction) Pretraining and compute-optimal scaling.** As value-based RL-QCO agents grow larger, it becomes important not only to increase model capacity, but to allocate compute across model size, batch size, and update-to-data (UTD) ratio in a compute-optimal way. Fu et al. [190] study this trade-off for online value-based deep RL and identify a TD-overfitting phenomenon: for small models, increasing batch size quickly harms Q-function accuracy, whereas large models can exploit much larger batches without degradation. Their scaling laws provide guidelines for choosing batch size and UTD under a fixed compute budget. Translating these ideas to RL-QCO suggests two directions: (i) scaling critic capacity together with batch size to avoid TD-overfitting when using large-batch quantum simulations, and (ii) combining such compute-optimal training with offline pretraining on large corpora of synthetic or compiled circuits, then fine-tuning online on hardware-constrained objectives.

8 Benchmark and evaluation

Benchmarking reinforcement learning-based quantum circuit optimization (RL-QCO) is challenging because the final reward often mixes several distinct quantities: physical task performance, circuit compactness, hardware compatibility, and the cost of training the RL agent. A useful benchmark should therefore answer four questions: Did the agent solve the quantum task? How expensive is the resulting circuit? How many quantum or simulator evaluations were required to find it? Does the circuit remain useful after compilation to realistic hardware? Reporting only the final reward is usually insufficient, since two

Table 8 Representative benchmark families for RL-based quantum circuit optimization.

Benchmark family	Typical instances	What should be reported
Molecular VQE/QAS	H_2, LiH, BeH_2, H_4, active-space chemistry Hamiltonians	Energy error, chemical accuracy, depth, two-qubit gates, parameters, quantum evaluations
Spin and model Hamiltonians	Transverse-field Ising, Heisenberg, XXZ, Hubbard, SYK-inspired models	Energy error, scaling with qubit number, transfer across sizes, robustness to disorder
QAOA and combinatorial optimization	MaxCut, Ising spin glasses, Max-k-SAT, graph coloring, TSP-derived Hamiltonians	Approximation ratio, expected cost, best sampled cost, probability of near-optimal samples
State preparation and synthesis	Bell, GHZ, W, Dicke, stabilizer, random states, target unitaries	State fidelity, process fidelity, sequence length, success probability
Compilation, routing, and rewriting	Clifford+T circuits, QASM circuits, MQT Bench, QASMBench, SupermarQ, coupling maps	Depth reduction, gate-count reduction, SWAP overhead, native-gate count, correctness
Fault-tolerant and logical circuits	Arithmetic primitives, T-count benchmarks, logical state preparation, verification circuits	T-count, T-depth, ancilla count, logical depth, verification overhead

agents may obtain similar rewards while differing substantially in circuit depth, two-qubit-gate count, training cost, or noise robustness.

8.1 Benchmark families

Existing RL-QCO studies mainly use six benchmark families, summarized in Table 8. Molecular VQE and QAS benchmarks are common because small molecules provide well-defined Hamiltonians and exact reference energies, making them useful for evaluating ansatz discovery [35, 109]. Recent works [142] consider molecules from the "*Top 20 molecules for quantum computing*" available by Pennylane [191]. For similar applications one can also use QMProt which is a dataset developed to support quantum computing applications in protein research [192]. QAOA and combinatorial optimization benchmarks, especially MaxCut and Ising-type problems, are useful for testing whether RL policies can generalize across graph instances and problem sizes [122, 123]. State-preparation and unitary-synthesis tasks test whether an agent can construct short circuits that reach a target state or unitary with high fidelity [117, 47]. Compilation, rewriting, routing, and scheduling benchmarks evaluate RL as a circuit-improvement or hardware-mapping tool, including recent graph-based and ZX-calculus approaches [108, 101]. Finally, fault-tolerant and logical-level benchmarks are becoming increasingly important because they shift attention from NISQ depth reduction to resources such as T-count, T-depth, ancilla cost, and logical error constraints [124, 125].

As Table 8 indicates, the appropriate metric depends strongly on the task. This is a major source of inconsistency in the literature. In molecular benchmarks, the external classical parameter optimizer can dominate the final energy and obscure the contribution of the RL policy. In QAOA benchmarks, graph ensembles, classical baselines, and depth budgets often differ across papers. In state-preparation and synthesis tasks, full access to the target state or unitary is sometimes assumed, which is not scalable to large systems. In compilation benchmarks, logical circuit metrics are frequently mixed with post transpilation native-gate metrics. In fault-tolerant settings, resource estimates depend strongly on the assumed code, architecture, and magic-state factory.

8.2 Evaluation metrics

A compact but complete evaluation protocol should include four metric groups: task performance, circuit resources, learning efficiency, and noise or hardware robustness. This separation is important because an RL-QCO method may improve the physical objective while producing circuits that are too deep, too expensive to discover, or too fragile after hardware transpilation [1, 21, 193].

Task performance. For energy-minimization tasks, the standard metric is the energy error

$$\Delta E = |E(C, \theta^\star) - E_0|, \tag{15}$$

where C is the circuit found by the agent, $\theta^\star$ are the optimized parameters, and E_0 is the reference ground-state energy. For molecular VQE, chemical accuracy should also be reported, following the standard practice in variational quantum chemistry [60, 21]. For QAOA and combinatorial optimization, the natural metrics are the expected objective value, approximation ratio, best sampled bitstring, and the probability of sampling an optimal or near-optimal solution [61, 122]. For state preparation and compiling, one should report state fidelity, process fidelity, or an equivalent task-specific distance [194, 47].

Circuit resources. Task performance alone is not enough: an RL agent should also produce circuits that are compact and executable. At minimum, studies should report qubit count, circuit depth, total gate count, 2-qubit gate count, and number of trainable parameters [1, 21]. For hardware-aware studies, these metrics should be reported after transpilation to the native gate set, because logical depth and gate count can differ substantially from post-routing hardware cost [75, 79, 195, 6]. For fault-tolerant benchmarks, T-count, T-depth, logical depth, ancilla count, and verification overhead are more informative than raw gate count, since non-Clifford resources dominate the cost of many fault-tolerant implementations [76, 78, 124, 125].

Learning efficiency. RL-QCO should also be evaluated by the cost of discovering the circuit. This is particularly important when each environment step requires a quantum simulation, noisy circuit evaluation, or hardware execution. Studies should report the number of episodes, environment steps, quantum circuit evaluations, shots, wall-clock training time, and memory usage. In variational settings, the cost of the classical parameter optimizer must be included. A useful quantity is

$$N_{\text{qeval}} = \sum_{e,t} N_{\text{opt}}(e,t) N_{\text{meas}}(e,t), \tag{16}$$

where $N_{\text{opt}}(e,t)$ is the number of parameter-optimizer calls triggered at episode e and step t, and $N_{\text{meas}}(e,t)$ is the number of measurement evaluations required per optimizer call. Without this accounting, methods that frequently re-optimize variational parameters may appear more sample-efficient than they actually are. This issue is closely related to broader concerns in deep-RL reproducibility, where reported performance can depend strongly on implementation details, random seeds, and tuning budgets [104, 165, 166].

Noise and hardware robustness. Noiseless simulation is useful for debugging, but it is not sufficient for NISQ-oriented circuit optimization [1]. Evaluation should ideally include ideal simulation, shot-noise simulation, calibrated noise-model simulation, and, where possible, real-hardware execution. Hardware-aware reports should specify the backend, native gate set, coupling map, transpiler settings, shot budget, calibration date, and any error-mitigation procedure [196, 79, 193, 197]. This is essential because two circuits with the same depth may behave differently if one uses high-error 2-qubit links, unfavorable routing paths, or qubits with poor coherence.

8.3 Hyperparameter tuning and reproducibility

Hyperparameter optimization is a hidden subroutine in RL-QCO evaluation. Performance can depend strongly on learning rate, discount factor, entropy coefficient, replay-buffer size, target-network update frequency, PPO clipping range, batch size, exploration schedule, reward scaling, and network architecture [104, 165]. Several of these choices are not incidental implementation details but are tied to the underlying RL algorithm itself: replay buffers and target networks are central to DQN-style methods [87, 89], clipping parameters shape PPO updates [92], and entropy regularization controls exploration in maximum-entropy actor–critic methods such as SAC [93]. In RL-QCO, this sensitivity is amplified by noisy quantum rewards, expensive simulator or hardware calls, and task-dependent circuit representations; recent RL-QAS benchmarking further shows that no single agent or hyperparameter regime dominates across tasks, qubit counts, and noise settings [166].

Several emerging resources can support more reproducible evaluation. RL-specific environments such as qgym and qcd-gym provide controlled settings for compilation and circuit-design tasks [108, 107]. BenchRL-QAS is particularly relevant because it compares multiple RL agents across VQE, VQSD, VQC, and state-preparation tasks under noiseless and noisy conditions [166]. Beyond RL-specific tools, MQT Bench, QASMBench, SupermarQ, QED-C benchmarks, HamLib and YAQQ provide standardized circuits or Hamiltonians that can be wrapped as RL environments [195, 6, 197, 193, 198, 199].

Overall, RL-QCO benchmarks should move toward multi-objective, budget-aware, and hardware-aware reporting. A fair comparison should specify: (i) the task distribution and train–validation–test split; (ii) the circuit language, gate set, connectivity constraints, and maximum depth; (iii) the state representation, action space, reward function, and episode termination rule; (iv) the simulator or hardware backend, noise model, shot budget, and transpilation settings; (v) the quantum-evaluation budget and classical-optimizer budget; (vi) the hyperparameter tuning budget and selected hyperparameters; and (vii) statistical variation over multiple random seeds. Such reporting would make it easier to distinguish genuine algorithmic progress from gains due to favorable benchmarks, larger tuning budgets, hidden classical optimization costs, or task-specific engineering.

8.4 Baselines, ablations, and comparison protocol

A reliable benchmark should not evaluate an RL-QCO method in isolation. Since the RL pipeline combines several design choices corresponds to the RL-state representation, reward shaping, action space construction, neural architecture, and classical parameter optimization. Instead the performance gains should be attributed through baselines and ablations [104, 165, 166]. For ansatz-search tasks, relevant baselines include hardware-efficient ansatzes, problem-inspired ansatzes,

random architecture search, greedy gate insertion, evolutionary search, Bayesian optimization, and adaptive methods such as ADAPT-VQE [22, 50, 52, 63, 53]. For QAOA tasks, RL policies should be compared with standard angle-optimization strategies, fixed-depth QAOA, recursive QAOA variants, and classical heuristics for the underlying combinatorial problem [61, 122, 123]. For compilation, routing, and rewriting, RL agents should be compared with standard compiler pipelines, greedy routing, heuristic SWAP insertion, peephole optimization, ZX-calculus simplification, and pass-sequence baselines [75, 79, 156, 77, 101, 155].

Ablation studies are equally important. At minimum, an RL-QCO study should test whether the reported improvement remains when individual components are removed or simplified: the state encoding, reward-shaping terms, action masking, macro-actions or learned gadgets, hardware-noise information, replay-buffer strategy, and neural-network architecture. Reward shaping and action masking are particularly important to ablate because they can substantially alter exploration and credit assignment [102, 103]. For variational tasks, the effect of the classical optimizer should be isolated by using the same optimizer and optimization budget across all compared methods. This prevents improvements due to extra parameter tuning from being misattributed to the RL policy.

Finally, benchmarks should distinguish in-distribution performance from generalization. An agent trained on a fixed molecule, graph family, circuit size, or hardware topology may not transfer to larger systems, different Hamiltonians, new graph distributions, or updated device calibrations. Therefore, a strong evaluation protocol should include both standard test instances and out-of-distribution tests, such as larger qubit numbers, unseen molecules, unseen MaxCut graph families, different coupling maps, or noisier hardware settings. This is especially important for RL-QCO because the ultimate goal is not to overfit a small benchmark instance, but to learn reusable circuit-design or circuit-optimization strategies [109, 114, 166].

9 Open challenges and future directions

The next stage of RL-based quantum circuit optimization should not be viewed simply as training larger agents on larger circuits. The deeper challenge is to formulate better decision processes: state spaces that expose useful quantum structure, action spaces that avoid unnecessary combinatorial growth, and rewards that reflect both physical performance and resource constraints. This view is consistent with the MDP-based organization of RL-QCO around state, reward, action, and agent design choices [83, 166]

Abstraction rather than brute-force scaling. A major limitation of current RL-QCO methods is that many still operate close to the gate level, where each episode becomes a long sequence of local decisions. This makes exploration difficult and credit assignment weak. A promising direction is hierarchical RL, where high-level policies choose ansatz blocks, ZX-rewrite patterns, compiler passes, fermionic excitations, or learned gadgets, while low-level policies perform local refinement. Such abstraction can reduce the effective horizon of the search problem and allow policies to transfer across related Hamiltonians, graph instances, and hardware layouts [109, 101].

Learning reusable circuit priors. Most RL-QCO agents are trained nearly from scratch for each task family, leading to significant inefficiency. In practice, many useful circuit motifs recur across problems, including entangling layers, symmetry-preserving blocks, cancellation identities, routing patterns, and compiler-pass sequences. This suggests a natural direction toward amortized circuit optimization, where agents are pretrained on large and diverse collections of circuits, Hamiltonians, compiler traces, or successful quantum circuits, and subsequently fine-tuned for new tasks. Such an approach would shift RL-QCO from instance-specific search toward learning reusable circuit-design priors. Initial steps in this direction have already been explored through reusable circuit gadgets [143] and cross-task replay buffer reuse [144].

Reward evaluation as the bottleneck. In many settings, the expensive part of RL-QCO is not the policy update but the environment call: estimating an energy, fidelity, noisy expectation value, or hardware performance. Future agents should therefore learn when to query expensive evaluators. Surrogate reward models, uncertainty-aware predictors, multi-fidelity simulation, and active-learning-style querying could allow an agent to explore cheaply most of the time and reserve accurate simulations or hardware calls for the most informative circuits. Recent work on training-free QAS proxies and amortized reward evaluation already points in this direction [145, 144].

Joint optimization of structure, parameters, and measurements. Many RL-QAS pipelines separate discrete circuit construction from continuous parameter optimization. This separation is convenient but incomplete: the quality of a variational circuit depends jointly on the gate structure, parameter initialization, optimizer trajectory, measurement allocation, and shot budget. A more integrated direction is to let the agent decide not only which gate or block to add, but also when to re-optimize parameters, how many shots to allocate, and which observables to measure together. Hybrid discrete–continuous

RL and trapped-ion compilation studies provide early evidence that jointly treating structure and parameters can improve circuit quality over purely discrete design loops [159, 160].

Hardware-adaptive and continual RL. The simulation-hardware gap cannot be solved only by adding noise to the simulator. Real devices have time-dependent calibration data, multi-core scalability, non-uniform qubit quality, crosstalk, queue-time variation, and backend-specific native gates. Future RL-QCO agents should be conditioned on hardware descriptors and should adapt continually as calibration data change. Instead of learning one fixed policy for one static device, the goal should be policies that generalize across a distribution of devices and update efficiently when the hardware drifts. This direction is supported by noise-adaptive compilation, continual QAS under changing noise, and recent RL-based hardware-control studies [200, 79, 37, 119, 180].

Fault-tolerant co-design. As quantum computing moves beyond NISQ demonstrations, the objective of circuit optimization will change. Depth and CNOT count will no longer be sufficient; T-count, T-depth, magic-state consumption, ancilla scheduling, lattice-surgery layout, syndrome-extraction constraints, and logical error rates will become central. Recent work suggests that RL and related search methods can already contribute to logical state preparation and non-Clifford resource reduction. A key future direction is to integrate RL-QCO directly with fault-tolerant compilation and error-correction workflows rather than treating error correction as a later compilation stage. [76, 78, 124, 125].

Theory of circuit-space exploration. RL-QCO is still largely empirical. We lack a theory explaining when an action space is expressive enough, when reward shaping preserves the desired optimum, how many circuit evaluations are needed to find a useful architecture, and why some learned policies transfer while others overfit. Progress on sample complexity, expressivity under action abstraction, equivalence classes of circuits, and exploration in symmetry-constrained circuit spaces would make the field more principled and less dependent on trial-and-error environment design. Useful starting points include classical theory on reward shaping and action-space design, together with quantum results on expressibility and barren plateaus [102, 150, 80, 42].

From black-box agents to interpretable design rules. The most valuable RL-QCO systems may not be those that output a single optimized circuit, but those that reveal reusable design principles. Agents that discover interpretable motifs, rewrite identities, symmetry-preserving blocks, or hardware-specific routing strategies could provide scientific insight beyond benchmark improvement. This suggests an important shift: RL should not only automate circuit optimization, but also help build a library of human-understandable circuit-design patterns [201, 49]. ZX-calculus-based simplification, compiler-pass selection, and automated gadget discovery are promising steps toward this more interpretable form of RL-QCO [77, 101, 155, 153].

Overall, the central opportunity is to move from instance-specific circuit search to reusable, hardware-aware, and theory-guided circuit-design intelligence. The field will advance most rapidly if future RL-QCO systems combine structured quantum representations, adaptive action abstractions, efficient reward evaluation, and direct integration with both NISQ hardware and fault-tolerant compilation stacks [1, 21, 166].

10 Conclusion

Reinforcement learning has emerged as a promising design engine for quantum circuit optimization, casting varied tasks like variational ansatz construction, unitary and state compilation, qubit mapping and routing, circuit simplification, and fault-tolerant resource reduction as instances of a common constrained search. Rather than organizing the literature by application domain or by hardware platform, here we analyzed the field along four core design axes: state representation, reward engineering, action space, and agent architecture. The recurring observation is that what most strongly differentiates one RL-QCO system from another is not the underlying RL algorithm, but the way these four design choices interact with quantum structure, hardware constraints, and the cost of reward evaluation.

Across the works, several patterns emerge. State representations span circuit-, state-, problem-, and hardware-level encodings, and the most scalable methods exploit structured representations such as tensor grids, ZX-diagrams, stabilizers, or coupling graphs rather than raw state vectors. Reward design has converged on multi-objective signals that combine an energy or fidelity term with explicit resource and noise penalties, with reward shaping and action masking serving as practical levers for tractable exploration in combinatorial circuit spaces. Action vocabularies range from low-level gate placements to higher-level macros, ZX rewrites, Fermionic excitations, and learned gadgets, and recent works show that hardware-native and topology-filtered action spaces consistently improve the alignment between the RL objective and the eventual hardware cost. On the agent design, the field is dominated by DDQN, PPO, and AlphaZero/MuZero-style search-augmented methods,

and benchmarking now confirms that no single agent or neural backbone dominates across tasks, qubit counts, and noise regimes.

This study surfaces persistent limitations. Most demonstrations are still confined to small qubit counts and shallow circuits, the simulation-to-hardware gap remains poorly characterized, and benchmarks differ in tasks, baselines, hyperparameter budgets, and accounting of classical optimizer cost, which makes cross-paper comparison difficult. Few studies separate the contribution of the RL policy from that of reward shaping, action masking, or the inner variational optimizer, and there is little theoretical guidance on when an action space is expressive enough or how many circuit evaluations are needed to find a useful architecture.

Looking ahead, the prominent opportunity is to move from instance-specific circuit search to reusable, hardware-aware, and theory-guided circuit-design intelligence. Hierarchical and abstraction-driven action spaces, surrogate and multi-fidelity reward models, joint optimization of structure, parameters, and measurement budgets, hardware-adaptive and continual policies, and tighter integration with fault-tolerant compilation stacks all point towards RL-QCO methods that scale beyond NISQ demonstrations. Realizing this potential will require not only better agents, but also community-wide conventions on benchmarks, baselines, and reporting that allow genuine algorithmic progress in RL for quantum circuit optimization to be identified and built upon.

References

[1] John Preskill. Quantum computing in the nisq era and beyond. Quantum, 2:79, 2018.

[2] Jens Eisert and John Preskill. Mind the gaps: The fraught road to quantum advantage. arXiv preprint arXiv:2510.19928, 2025.

[3] Zoltán Zimborás, Bálint Koczor, Zoë Holmes, Elsi-Mari Borrelli, András Gilyén, Hsin-Yuan Huang, Zhenyu Cai, Antonio Acín, Leandro Aolita, Leonardo Banchi, et al. Myths around quantum computation before full fault tolerance: What no-go theorems rule out and what they don't. arXiv preprint arXiv:2501.05694, 2025.

[4] Krzysztof Kurowski, Piotr Rydlichowski, Konrad Wojciechowski, Tomasz Pecyna, and Mateusz Slysz. Application performance benchmarks for quantum computers. arXiv preprint arXiv:2310.13637, 2023.

[5] Andrew W Cross, Lev S Bishop, Sarah Sheldon, Paul D Nation, and Jay M Gambetta. Validating quantum computers using randomized model circuits. Physical Review A, 100(3):032328, 2019.

[6] Ang Li, Samuel Stein, Sriram Krishnamoorthy, and James Ang. Qasmbench: A low-level quantum benchmark suite for nisq evaluation and simulation. ACM Transactions on Quantum Computing, 4(2):1–26, 2023.

[7] Guanru Feng, Joel J Wallman, Brandon Buonacorsi, Franklin H Cho, Daniel K Park, Tao Xin, Dawei Lu, Jonathan Baugh, and Raymond Laflamme. Estimating the coherence of noise in quantum control of a solid-state qubit. Physical review letters, 117(26):260501, 2016.

[8] Daniel Koch, Brett Martin, Saahil Patel, Laura Wessing, and Paul M Alsing. Demonstrating nisq era challenges in algorithm design on ibm's 20 qubit quantum computer. AIP Advances, 10(9), 2020.

[9] Jonathan J Burnett, Andreas Bengtsson, Marco Scigliuzzo, David Niepce, Marina Kudra, Per Delsing, and Jonas Bylander. Decoherence benchmarking of superconducting qubits. npj Quantum Information, 5(1):54, 2019.

[10] Tomislav Begušić, Johnnie Gray, and Garnet Kin-Lic Chan. Fast and converged classical simulations of evidence for the utility of quantum computing before fault tolerance. Science Advances, 10(3):eadk4321, 2024.

[11] Youngseok Kim, Andrew Eddins, Sajant Anand, Ken Xuan Wei, Ewout Van Den Berg, Sami Rosenblatt, Hasan Nayfeh, Yantao Wu, Michael Zaletel, Kristan Temme, et al. Evidence for the utility of quantum computing before fault tolerance. Nature, 618(7965):500–505, 2023.

[12] Christopher G Yale, Rich Rines, Victory Omole, Bharath Thotakura, Ashlyn D Burch, Matthew NH Chow, Megan Ivory, Daniel Lobser, Brian K McFarland, Melissa C Revelle, et al. Noise-aware circuit compilations for a continuously parameterized two-qubit gateset. Physical Review Applied, 24(2):024057, 2025.

[13] Hanrui Wang, Yongshan Ding, Jiaqi Gu, Yujun Lin, David Z Pan, Frederic T Chong, and Song Han. Quantumnas: Noise-adaptive search for robust quantum circuits. In 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pages 692–708. IEEE, 2022.

[14] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. Parameterized quantum circuits as machine learning models. Quantum science and technology, 4(4):043001, 2019.

[15] Mateusz Ostaszewski, Edward Grant, and Marcello Benedetti. Structure optimization for parameterized quantum circuits. Quantum, 5:391, 2021.

[16] Tyler Volkoff and Patrick J Coles. Large gradients via correlation in random parameterized quantum circuits. Quantum Science & Technology, 6(2):025008, 2021.

[17] Guangxi Li, Ruilin Ye, Xuanqiang Zhao, and Xin Wang. Concentration of data encoding in parameterized quantum circuits. Advances in Neural Information Processing Systems, 35:19456–19469, 2022.

[18] Sukin Sim, Jonathan Romero, Jérôme F Gonthier, and Alexander A Kunitsa. Adaptive pruning-based optimization of parameterized quantum circuits. Quantum Science & Technology, 6(2):025019, 2021.

[19] Tobias Haug and MS Kim. Natural parametrized quantum circuit. Physical Review A, 106(5):052611, 2022.

[20] Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermanni Heimonen, Jakob S Kottmann, Tim Menke, et al. Noisy intermediate-scale quantum algorithms. Reviews of Modern Physics, 94(1):015004, 2022.
[21] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. Nature Reviews Physics, 3(9):625–644, 2021.
[22] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. nature, 549(7671):242–246, 2017.
[23] Lorenzo Leone, Salvatore FE Oliviero, Lukasz Cincio, and Marco Cerezo. On the practical usefulness of the hardware efficient ansatz. Quantum, 8:1395, 2024.
[24] Alexandre Choquette, Agustin Di Paolo, Panagiotis Kl Barkoutsos, David Sénéchal, Ivano Tavernelli, and Alexandre Blais. Quantum-optimal-control-inspired ansatz for variational quantum algorithms. Physical Review Research, 3(2):023092, 2021.
[25] Dekel Meirom and Steven H Frankel. Pansatz: Pulse-based ansatz for variational quantum algorithms. Frontiers in Quantum Science and Technology, 2:1273581, 2023.
[26] Kaoru Nakamura, M Lakshmanan, Pierre Gaspard, and SA Rice. Field-theoretical model inspired by adiabatic-ansatz eigenvalue problems. Physical Review A, 46(10):6311, 1992.
[27] Harper R Grimsley, Daniel Claudino, Sophia E Economou, Edwin Barnes, and Nicholas J Mayhall. Is the trotterized uccsd ansatz chemically well-defined? Journal of chemical theory and computation, 16(1):1–6, 2019.
[28] Rongxin Xia and Sabre Kais. Qubit coupled cluster singles and doubles variational quantum eigensolver ansatz for electronic structure calculations. Quantum Science & Technology, 6(1):015001, 2021.
[29] Sukin Sim, Peter D. Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. Advanced Quantum Technologies, 2(12):1900070, 2019.
[30] Yuhan Huang, Qingyu Li, Xiaokai Hou, Rebing Wu, Man-Hong Yung, Abolfazl Bayat, and Xiaoting Wang. Robust resource-efficient quantum variational ansatz through an evolutionary algorithm. Phys. Rev. A, 105:052414, May 2022.
[31] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. Nature Communications, 9(1), November 2018.
[32] Samson Wang, Enrico Fontana, M. Cerezo, Kunal Sharma, Akira Sone, Lukasz Cincio, and Patrick J. Coles. Noise-induced barren plateaus in variational quantum algorithms. Nature Communications, 12(1), November 2021.
[33] Michael Ragone, Bojko N. Bakalov, Frédéric Sauvage, Alexander F. Kemper, Carlos Ortiz Marrero, Martín Larocca, and M. Cerezo. A lie algebraic theory of barren plateaus for deep parameterized quantum circuits. Nature Communications, 15(1), August 2024.
[34] Shi-Xin Zhang, Chang-Yu Hsieh, Shengyu Zhang, and Hong Yao. Differentiable quantum architecture search. Quantum Science and Technology, 7(4):045023, August 2022.
[35] Mateusz Ostaszewski, Lea M Trenkwalder, Wojciech Masarczyk, Eleanor Scerri, and Vedran Dunjko. Reinforcement learning for optimization of variational quantum circuit architectures. Advances in neural information processing systems, 34:18182–18194, 2021.
[36] Thomas Fösel, Murphy Yuezhen Niu, Florian Marquardt, and Li Li. Quantum circuit optimization with deep reinforcement learning. arXiv preprint arXiv:2103.07585, 2021.
[37] Esther Ye and Samuel Yen-Chi Chen. Quantum architecture search via continual reinforcement learning. arXiv preprint arXiv:2112.05779, 2021.
[38] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578, 2016.
[39] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. ACM Computing Surveys, 54(4):1–34, 2021.
[40] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. Journal of Machine Learning Research, 20(55):1–21, 2019.
[41] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 4095–4104. PMLR, 2018.
[42] Sukin Sim, Peter D Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. Advanced Quantum Technologies, 2(12):1900070, 2019.
[43] Aleks Kissinger and John van de Wetering. Reducing the number of non-clifford gates in quantum circuits. Physical Review A, 102(2):022406, 2020.
[44] Emanuel Knill. Quantum computing with realistically noisy devices. Nature, 434(7029):39–44, 2005.
[45] Pei Yuan, Jonathan Allcock, and Shengyu Zhang. Does qubit connectivity impact quantum circuit complexity? IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 43(2):520–533, 2023.
[46] Alessio Cicero et al. Simulation of quantum computers: Review and acceleration opportunities. ACM Transactions on Quantum Computing, 7(1):1–35, 2025.
[47] Lorenzo Moro, Matteo GA Paris, Marcello Restelli, and Enrico Prati. Quantum compiling by deep reinforcement learning. Communications Physics, 4(1):178, 2021.
[48] Yuxuan Du, Tao Huang, Shan You, Min-Hsiu Hsieh, and Dacheng Tao. Quantum circuit architecture search for variational quantum algorithms. npj Quantum Information, 8(1):62, 2022.
[49] Shubing Xie, Aritra Sarkar, and Sebastian Feld. Deqompile: quantum circuit decompilation using genetic programming for explainable quantum architecture search. arXiv preprint arXiv:2504.08310, 2025.
[50] Harper R Grimsley, Sophia E Economou, Edwin Barnes, and Nicholas J Mayhall. An adaptive variational algorithm for exact molecular simulations on a quantum computer. Nature communications, 10(1):3007, 2019.

[51] Shi-Xin Zhang, Chang-Yu Hsieh, Shengyu Zhang, and Hong Yao. Differentiable quantum architecture search. Quantum Science & Technology, 7(4):045023, 2022.
[52] Anqi Zhang and Shengmei Zhao. Evolutionary-based quantum architecture search. arXiv preprint arXiv:2212.00421, 2022.
[53] Xudong Lu, Kaisen Pan, Ge Yan, Jiaming Shan, Wenjie Wu, and Junchi Yan. Qas-bench: rethinking quantum architecture search and a benchmark. In International conference on machine learning, pages 22880–22898. PMLR, 2023.
[54] Vicente P. Soloviev et al. Trainability maximization using estimation of distribution algorithms assisted by surrogate modelling for quantum architecture search. EPJ Quantum Technology, 11(1):69, 2024.
[55] Xuan Zhang, Limei Wang, Jacob Helwig, Youzhi Luo, Cong Fu, Yaochen Xie, Meng Liu, Yuchao Lin, Zhao Xu, Keqiang Yan, et al. Artificial intelligence for science in quantum, atomistic, and continuum systems. Foundations and Trends® in Machine Learning, 18(4):385–849, 2025.
[56] Yuri Alexeev, Marwa H Farag, Taylor L Patti, Mark E Wolf, Natalia Ares, Alán Aspuru-Guzik, Simon C Benjamin, Zhenyu Cai, Shuxiang Cao, Christopher Chamberland, et al. Artificial intelligence for quantum computing. Nature Communications, 16(1):10829, 2025.
[57] Callum W Duncan, Pablo M Poggi, Marin Bukov, Nikolaj Thomas Zinner, and Steve Campbell. Taming quantum systems: a tutorial for using shortcuts-to-adiabaticity, quantum optimal control, and reinforcement learning. PRX Quantum, 6(4):040201, 2025.
[58] Marin Bukov and Florian Marquardt. Reinforcement learning for quantum technology. arXiv preprint arXiv:2601.18953, 2026.
[59] Darya Martyniuk, Johannes Jung, and Adrian Paschke. Quantum architecture search: a survey. In 2024 IEEE International Conference on Quantum Computing and Engineering (QCE), volume 1, pages 1695–1706. IEEE, 2024.
[60] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. Nature communications, 5(1):4213, 2014.
[61] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. arXiv preprint arXiv:1411.4028, 2014.
[62] Akash Kundu, Przemysław Bedełek, Mateusz Ostaszewski, Onur Danaci, Yash J Patel, Vedran Dunjko, and Jarosław A Miszczak. Enhancing variational quantum state diagonalization using reinforcement learning techniques. New Journal of Physics, 26(1):013034, 2024.
[63] Trong Duong, Sang T Truong, Minh Tam, Bao Bach, Ju-Young Ryu, and June-Koo Kevin Rhee. Quantum neural architecture search with quantum circuits metric and bayesian optimization. arXiv preprint arXiv:2206.14115, 2022.
[64] Yize Sun, Zixin Wu, Volker Tresp, and Yunpu Ma. Quantum architecture search with unsupervised representation learning. Quantum, 10:1994, 2026.
[65] Philip Andreasson, Joel Johansson, Simon Liljestrand, and Mats Granath. Quantum error correction for the toric code using deep reinforcement learning. Quantum, 3:183, 2019.
[66] Arshpreet Singh Maan and Alexandru Paler. Machine learning message-passing for the scalable decoding of qldpc codes. npj Quantum Information, 11(1):78, 2025.
[67] Alessio Fallani, Matteo AC Rossi, Dario Tamascelli, and Marco G Genoni. Learning feedback control strategies for quantum metrology. PRX Quantum, 3(2):020310, 2022.
[68] Valeria Cimini, Mauro Valeri, Emanuele Polino, Simone Piacentini, Francesco Ceccarelli, Giacomo Corrielli, Nicolò Spagnolo, Roberto Osellame, and Fabio Sciarrino. Deep reinforcement learning for quantum multiparameter estimation. Advanced Photonics, 5(1):016005, 2023.
[69] Samuel Yen-Chi Chen, Chao-Han Huck Yang, Jun Qi, Pin-Yu Chen, Xiaoli Ma, and Hsi-Sheng Goan. Variational quantum circuits for deep reinforcement learning. IEEE access, 8:141007–141024, 2020.
[70] Owen Lockwood and Mei Si. Reinforcement learning with quantum variational circuit. In Proceedings of the AAAI conference on artificial intelligence and interactive digital entertainment, volume 16, pages 245–251, 2020.
[71] Nico Meyer, Christian Ufrecht, George Yammine, Georgios Kontes, Christopher Mutschler, and Daniel D Scherer. Benchmarking quantum reinforcement learning. arXiv preprint arXiv:2501.15893, 2025.
[72] Huanyu Liu, Ge Li, Jia Li, Hao Zhu, Kechi Zhang, and Yihong Dong. Saturn: Sat-based reinforcement learning to unleash llms reasoning. Advances in Neural Information Processing Systems, 38:27680–27712, 2026.
[73] Christopher M Dawson and Michael A Nielsen. The solovay-kitaev algorithm. arXiv preprint quant-ph/0505030, 2005.
[74] Anna M Krol, Aritra Sarkar, Imran Ashraf, Zaid Al-Ars, and Koen Bertels. Efficient decomposition of unitary matrices in quantum circuit compilers. Applied Sciences, 12(2):759, 2022.
[75] 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, pages 1001–1014, 2019.
[76] Matthew Amy, Dmitri Maslov, and Michele Mosca. Polynomial-time t-depth optimization of clifford+ t circuits via matroid partitioning. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 33(10):1476–1489, 2014.
[77] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John Van De Wetering. Graph-theoretic simplification of quantum circuits with the zx-calculus. Quantum, 4:279, 2020.
[78] Luke E Heyfron and Earl T Campbell. An efficient quantum compiler that reduces t count. Quantum Science and Technology, 4(1):015004, 2019.
[79] Prakash Murali, Jonathan M Baker, Ali Javadi-Abhari, Frederic T Chong, and Margaret Martonosi. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems, pages 1015–1029, 2019.
[80] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. Nature communications, 9(1):4812, 2018.

[81] Harsh Wadhwa, Rahul Bhowmick, Naipunnya Raj, Rajiv Sangle, Ruchira V Bhat, and Krishnakumar Sabapathy. Model selection in hybrid quantum neural networks with applications to quantum transformer architectures. arXiv preprint arXiv:2603.21749, 2026.
[82] Adi Botea, Akihiro Kishimoto, and Radu Marinescu. On the complexity of quantum circuit compilation. In Proceedings of the International Symposium on Combinatorial Search, volume 9, pages 138–142, 2018.
[83] Richard S Sutton, Andrew G Barto, et al. Reinforcement learning: An introduction, volume 1. MIT press Cambridge, 1998.
[84] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. Artificial intelligence, 101(1-2):99–134, 1998.
[85] Richard Bellman. Dynamic programming. science, 153(3731):34–37, 1966.
[86] Christopher JCH Watkins and Peter Dayan. Q-learning. Machine learning, 8(3):279–292, 1992.
[87] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. nature, 518(7540):529–533, 2015.
[88] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI conference on artificial intelligence, volume 30, 2016.
[89] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. arXiv preprint arXiv:1511.05952, 2015.
[90] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8(3):229–256, 1992.
[91] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In International conference on machine learning, pages 1928–1937. PmLR, 2016.
[92] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.
[93] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International conference on machine learning, pages 1861–1870. Pmlr, 2018.
[94] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In Machine learning proceedings 1990, pages 216–224. Elsevier, 1990.
[95] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science, 362(6419):1140–1144, 2018.
[96] Nuttapong Chentanez, Andrew G. Barto, and Satinder Singh. Intrinsically motivated reinforcement learning. In Advances in Neural Information Processing Systems 17 (NeurIPS 2004), 2004.
[97] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. Frontiers in Neurorobotics, 1:108, 2007.
[98] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. Advances in neural information processing systems, 29, 2016.
[99] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In International conference on machine learning, pages 2778–2787. PMLR, 2017.
[100] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In Proceedings of the 26th annual international conference on machine learning, pages 41–48, 2009.
[101] Jordi Riu, Jan Nogué, Gerard Vilaplana, Artur Garcia-Saez, and Marta P Estarellas. Reinforcement learning based quantum circuit optimization via zx-calculus. Quantum, 9:1758, 2025.
[102] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In Icml, volume 99, pages 278–287. Citeseer, 1999.
[103] Shengyi Huang and Santiago Ontañón. A closer look at invalid action masking in policy gradient algorithms. arXiv preprint arXiv:2006.14171, 2020.
[104] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In Proceedings of the AAAI conference on artificial intelligence, volume 32, 2018.
[105] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. arXiv preprint arXiv:1606.01540, 2016.
[106] Mark Towers, Ariel Kwiatkowski, John Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Kallinteris Andreas, Markus Krimmel, Arjun Kg, Rodrigo Perez-Vicente, et al. Gymnasium: A standard interface for reinforcement learning environments. Advances in Neural Information Processing Systems, 38, 2026.
[107] Philipp Altmann, Jonas Stein, Michael Kölle, Adelina Bärligea, Maximilian Zorn, Thomas Gabor, Thomy Phan, Sebastian Feld, and Claudia Linnhoff-Popien. Challenges for reinforcement learning in quantum circuit design. In 2024 IEEE International Conference on Quantum Computing and Engineering (QCE), volume 1, pages 1600–1610. IEEE, 2024.
[108] Stan Van Der Linde, Willem De Kok, Tariq Bontekoe, and Sebastian Feld. qgym: A gym for training and benchmarking rl-based quantum compilation. In 2023 IEEE international conference on quantum computing and engineering (qce), volume 2, pages 26–30. IEEE, 2023.
[109] Yash J. Patel, Akash Kundu, Mateusz Ostaszewski, Xavier Bonet-Monroig, Vedran Dunjko, and Onur Danaci. Curriculum reinforcement learning for quantum architecture search under hardware errors. In The Twelfth International Conference on Learning Representations, 2024.

[110] Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics. New Journal of Physics, 13(4):043016, 2011.
[111] Andrew M Childs, Eddie Schoute, and Cem M Unsal. Circuit transformations for quantum architectures. arXiv preprint arXiv:1902.09102, 2019.
[112] Alexander Koziell-Pipe, Richie Yeung, and Matthew Sutcliffe. Towards faster quantum circuit simulation using graph decompositions, gnns and reinforcement learning. In The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24, 2024.
[113] Alexander Mattick, Maniraman Periyasamy, Christian Ufrecht, Abhishek Y Dubey, Christopher Mutschler, Axel Plinge, and Daniel D Scherer. Optimizing quantum circuits via zx diagrams using reinforcement learning and graph neural networks. arXiv preprint arXiv:2504.03429, 2025.
[114] Frederic Rapp, David A Kreplin, Marco F Huber, and Marco Roth. Reinforcement learning-based architecture search for quantum machine learning. Machine Learning: Science and Technology, 6(1):015041, 2025.
[115] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. Nature, 588(7839):604–609, 2020.
[116] David Kremer, Victor Villar, Hanhee Paik, Ivan Duran, Ismael Faro, and Juan Cruz-Benito. Practical and efficient quantum circuit synthesis and transpiling with reinforcement learning. arXiv preprint arXiv:2405.13196, 2024.
[117] En-Jui Kuo, Yao-Lung L Fang, and Samuel Yen-Chi Chen. Quantum architecture search via deep reinforcement learning. arXiv preprint arXiv:2104.07715, 2021.
[118] Manwen Liao, Yan Zhu, Weitian Zhang, and Yuxiang Yang. Reinforced learning explicit circuit representations for quantum state characterization from local measurements. In Forty-second International Conference on Machine Learning.
[119] ZT Wang, Qiuhao Chen, Yuxuan Du, ZH Yang, Xiaoxia Cai, Kaixuan Huang, Jingning Zhang, Kai Xu, Jun Du, Yinan Li, et al. Quantum compiling with reinforcement learning on a superconducting processor. arXiv preprint arXiv:2406.12195, 2024.
[120] Atiye Zeynali and Zahra Bakhshi. Noise-adaptive quantum circuit mapping for multi-chip nisq systems via deep reinforcement learning. arXiv preprint arXiv:2511.18079, 2025.
[121] Jiahao Yao, Lin Lin, and Marin Bukov. Reinforcement learning for many-body ground-state preparation inspired by counterdiabatic driving. Physical Review X, 11(3):031070, 2021.
[122] Matteo M Wauters, Emanuele Panizon, Glen B Mbeng, and Giuseppe E Santoro. Reinforcement-learning-assisted quantum optimization. Physical Review Research, 2(3):033446, 2020.
[123] Yash J Patel, Sofiene Jerbi, Thomas Bäck, and Vedran Dunjko. Reinforcement learning assisted recursive qaoa. EPJ Quantum Technology, 11(1):6, 2024.
[124] Francisco JR Ruiz, Tuomas Laakkonen, Johannes Bausch, Matej Balog, Mohammadamin Barekatain, Francisco JH Heras, Alexander Novikov, Nathan Fitzpatrick, Bernardino Romera-Paredes, John van de Wetering, et al. Quantum circuit optimization with alphatensor. Nature Machine Intelligence, pages 1–12, 2025.
[125] Remmy Zen, Jan Olle, Luis Colmenarez, Matteo Puviani, Markus Müller, and Florian Marquardt. Quantum circuit discovery for fault-tolerant logical state preparation with reinforcement learning. Physical Review X, 15(4):041012, 2025.
[126] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.
[127] Marius-Constantin Popescu, Valentina E Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. Multilayer perceptron and neural networks. WSEAS Transactions on Circuits and Systems, 8(7):579–588, 2009.
[128] Yaser Keneshloo, Tian Shi, Naren Ramakrishnan, and Chandan K Reddy. Deep reinforcement learning for sequence-to-sequence models. IEEE transactions on neural networks and learning systems, 31(7):2469–2489, 2019.
[129] Keiron O'shea and Ryan Nash. An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458, 2015.
[130] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems, 32(1):4–24, 2020.
[131] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.
[132] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. arXiv preprint arXiv:2404.19756, 2024.
[133] Zhimin He, Maijie Deng, Shenggen Zheng, Lvzhou Li, and Haozhen Situ. Gsqas: graph self-supervised quantum architecture search. Physica A: Statistical Mechanics and its Applications, 630:129286, 2023.
[134] Yize Sun, Zixin Wu, Yunpu Ma, and Volker Tresp. Quantum architecture search with unsupervised representation learning. arXiv preprint arXiv:2401.11576, 2024.
[135] Abhishek Gupta, Aldo Pacchiano, Yuexiang Zhai, Sham Kakade, and Sergey Levine. Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity. Advances in Neural Information Processing Systems, 35:15281–15295, 2022.
[136] Jonas Eschmann. Reward function design in reinforcement learning. Reinforcement learning algorithms: Analysis and Applications, pages 25–33, 2021.
[137] Rui Yu, Shenghua Wan, Yucen Wang, Chen-Xiao Gao, Le Gan, Zongzhang Zhang, and De-Chuan Zhan. Reward models in deep reinforcement learning: A survey. arXiv preprint arXiv:2506.15421, 2025.
[138] Marin Bukov, Alexandre GR Day, Dries Sels, Phillip Weinberg, Anatoli Polkovnikov, and Pankaj Mehta. Reinforcement learning in different phases of quantum control. Physical Review X, 8(3):031086, 2018.
[139] Murphy Yuezhen Niu, Sergio Boixo, Vadim N Smelyanskiy, and Hartmut Neven. Universal quantum control through deep reinforcement learning. npj Quantum Information, 5(1):33, 2019.

[140] Akash Kundu. Improving thermal state preparation of sachdev–ye–kitaev model with reinforcement learning on quantum hardware. Machine Learning: Science and Technology, 6(2):025066, 2025.
[141] Ioana Moflic, Alexandru Paler, and Akash Kundu. Qaser: Breaking the depth vs. accuracy trade-off for quantum architecture search. arXiv preprint arXiv:2511.16272, 2025.
[142] Akash Kundu and Stefano Mangini. Tensorrl-qas: Reinforcement learning with tensor networks for improved quantum architecture search. Advances in Neural Information Processing Systems, 38:120896–120930, 2026.
[143] Akash Kundu and Lorenzo Sarra. Reinforcement learning with learned gadgets to tackle hard quantum problems on real hardware. Communications Physics, 9:44, 2026.
[144] Akash Kundu and Sebastian Feld. Replay-buffer engineering for noise-robust quantum circuit optimization. arXiv preprint arXiv:2604.21863, 2026.
[145] Zhimin He, Maijie Deng, Shenggen Zheng, Lvzhou Li, and Haozhen Situ. Training-free quantum architecture search. In Proceedings of the AAAI conference on artificial intelligence, volume 38, pages 12430–12438, 2024.
[146] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. arXiv preprint arXiv:1904.12901, 2019.
[147] Yash Chandak, Georgios Theocharous, James Kostas, Scott Jordan, and Philip Thomas. Learning action representations for reinforcement learning. In International conference on machine learning, pages 941–950. PMLR, 2019.
[148] Tom Van de Wiele, David Warde-Farley, Andriy Mnih, and Volodymyr Mnih. Q-learning in enormous action spaces via amortized approximate maximization. arXiv preprint arXiv:2001.08116, 2020.
[149] Brian Sallans and Geoffrey E Hinton. Using free energies to represent q-values in a multiagent reinforcement learning task. Advances in Neural Information Processing Systems, 13, 2000.
[150] Anssi Kanervisto, Christian Scheller, and Ville Hautamäki. Action space shaping in deep reinforcement learning. In 2020 IEEE conference on games (CoG), pages 479–486. IEEE, 2020.
[151] Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J Mankowitz, and Shie Mannor. Learn what not to learn: Action elimination with deep reinforcement learning. Advances in neural information processing systems, 31, 2018.
[152] Michael Kölle, Tom Schubert, Philipp Altmann, Maximilian Zorn, Jonas Stein, and Claudia Linnhoff-Popien. A reinforcement learning environment for directed quantum circuit synthesis. arXiv preprint arXiv:2401.07054, 2024.
[153] Leopoldo Sarra, Kevin Ellis, and Florian Marquardt. Discovering quantum circuit components with program synthesis. Machine Learning: Science and Technology, 5(2):025029, 2024.
[154] Lea M Trenkwalder, Andrea López-Incera, Hendrik Poulsen Nautrup, Fulvio Flamini, and Hans J Briegel. Automated gadget discovery in the quantum domain. Machine Learning: Science and Technology, 4(3):035043, 2023.
[155] Daniel Mills, Ifan Williams, Jacob Swain, Gabriel Matos, Enrico Rinaldi, and Alexander Koziell-Pipe. Reinforcement learning for adaptive composition of quantum circuit optimisation passes. arXiv preprint arXiv:2601.21629, 2026.
[156] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. t| ket>: a retargetable compiler for nisq devices. Quantum Science & Technology, 6(1):014003, 2021.
[157] Yuchen Wang and David A Mazziotti. Quantum many-body simulations from a reinforcement-learned exponential ansatz. Physical Review A, 112(2):022403, 2025.
[158] David A Mazziotti. Contracted schrödinger equation: Determining quantum energies and two-particle density matrices without wave functions. Physical Review A, 57(6):4219, 1998.
[159] Jiayang Niu, Yan Wang, Jie Li, Ke Deng, Azadeh Alavi, Muhammad Usman, and Yongli Ren. Hybrid action reinforcement learning for quantum architecture search. arXiv preprint arXiv:2511.04967, 2025.
[160] Francesco Preti, Michael Schilling, Sofiene Jerbi, Lea M Trenkwalder, Hendrik Poulsen Nautrup, Felix Motzoi, and Hans J Briegel. Hybrid discrete-continuous compilation of trapped-ion quantum circuits with deep reinforcement learning. Quantum, 8:1343, 2024.
[161] 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 preprint arXiv:2605.02389, 2026.
[162] Panagiotis Promponas, Akrit Mudvari, Luca Della Chiesa, Paul Polakos, Louis Samuel, and Leandros Tassiulas. Compiler for distributed quantum computing: a reinforcement learning approach. In ICC 2025-IEEE International Conference on Communications, pages 4615–4621. IEEE, 2025.
[163] Qiyang Li, Zhiyuan Paul Zhou, and Sergey Levine. Reinforcement learning with action chunking. Advances in Neural Information Processing Systems, 38:55518–55553, 2026.
[164] Hai Nguyen, Tadashi Kozuno, Cristian C Beltran-Hernandez, and Masashi Hamaya. Symmetry-aware reinforcement learning for robotic assembly under partial observability with a soft wrist. In 2024 IEEE international conference on robotics and automation (ICRA), pages 9369–9375. IEEE, 2024.
[165] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What matters for on-policy deep actor-critic methods? a large-scale study. In International conference on learning representations, 2021.
[166] Azhar Ikhtiarudin, Aditi Das, Param Thakkar, and Akash Kundu. Benchrl-qas: Benchmarking reinforcement learning algorithms for quantum architecture search. In Proceedings of the AAAI Symposium Series, volume 7, pages 358–367, 2025.
[167] David H Wolpert and William G Macready. No free lunch theorems for optimization. IEEE transactions on evolutionary computation, 1(1):67–82, 2002.
[168] Jan Olle, Oleg M Yevtushenko, and Florian Marquardt. Scaling the automated discovery of quantum circuits via reinforcement learning with gadgets. arXiv preprint arXiv:2503.11638, 2025.
[169] Simone Foderà, Gloria Turati, Riccardo Nembrini, Maurizio Ferrari Dacrema, and Paolo Cremonesi. Reinforcement learning for variational quantum circuits design. arXiv preprint arXiv:2409.05475, 2024.

[170] Owen Lockwood. Optimizing quantum variational circuits with deep reinforcement learning. arXiv preprint arXiv:2109.03188, 2021.
[171] Xianchao Zhu and Xiaokai Hou. Quantum architecture search via truly proximal policy optimization. Scientific Reports, 13(1):5157, 2023.
[172] Siddhant Dutta, Nouhaila Innan, Sadok Ben Yahia, and Muhammad Shafique. Qas-qtns: Curriculum reinforcement learning-driven quantum architecture search for quantum tensor networks. In 2025 IEEE International Conference on Quantum Computing and Engineering (QCE), volume 1, pages 1739–1747. IEEE, 2025.
[173] Gerhard Stenzel, Isabella Debelic, Michael Kölle, Tobias Rohe, Leo Sünkel, Julian Hager, and Claudia Linnhoff-Popien. Reinforcement learning for parameterized quantum state preparation: A comparative study. arXiv preprint arXiv:2602.16523, 2026.
[174] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.
[175] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In International conference on machine learning, pages 1995–2003. PMLR, 2016.
[176] Sara Giordano, Kornikar Sen, and Miguel A Martin-Delgado. Hybrid reward-driven reinforcement learning for efficient quantum circuit synthesis. Quantum Machine Intelligence, 8(1):9, 2026.
[177] Wei Tang, Yiheng Duan, Yaroslav Kharkov, Rasool Fakoor, Eric Kessler, and Yunong Shi. Alpharouter: Quantum circuit routing with reinforcement learning and tree search. In 2024 IEEE International Conference on Quantum Computing and Engineering (QCE), volume 1, pages 930–940. IEEE, 2024.
[178] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. nature, 529(7587):484–489, 2016.
[179] Peiyong Wang, Muhammad Usman, Udaya Parampalli, Lloyd CL Hollenberg, and Casey R Myers. Automated quantum circuit design with nested monte carlo tree search. IEEE Transactions on Quantum Engineering, 4:1–20, 2023.
[180] Volodymyr Sivak, Alexis Morvan, Michael Broughton, Rodrigo G Cortiñas, Johannes Bausch, Andrew W Senior, Matthew Neeley, Alec Eickbusch, Noah Shutty, Laleh Aghababaie Beni, et al. Reinforcement learning control of quantum error correction. arXiv preprint arXiv:2511.08493, 2025.
[181] Akash Kundu, Aritra Sarkar, and Abhishek Sadhu. Kanqas: Kolmogorov-arnold network for quantum architecture search. EPJ Quantum Technology, 11(1):76, 2024.
[182] Petr Ivashkov, Po-Wei Huang, Kelvin Koor, Lirandë Pira, and Patrick Rebentrost. Qkan: quantum kolmogorov-arnold networks with applications in machine learning and multivariate state preparation. npj Quantum Information, 12(1):73, 2026.
[183] Jiun-Cheng Jiang, Morris Yu-Chao Huang, Tianlong Chen, and Hsi-Sheng Goan. Quantum variational activation functions empower kolmogorov-arnold networks. arXiv preprint arXiv:2509.14026, 2025.
[184] Yu-Chao Hsu, Jiun-Cheng Jiang, Chun-Hua Lin, Kuo-Chung Peng, Nan-Yow Chen, Samuel Yen-Chi Chen, En-Jui Kuo, and Hsi-Sheng Goan. Qkan-lstm: Quantum-inspired kolmogorov–arnold long short-term memory. In 2026 International Conference on Quantum Communications, Networking, and Computing (QCNC), pages 650–659. IEEE, 2026.
[185] Kuo-Chung Peng, Samuel Yen-Chi Chen, Jiun-Cheng Jiang, Chen-Yu Liu, En-Jui Kuo, Yun-Yuan Wang, Prayag Tiwari, Andrea Ceschini, Chi-Sheng Chen, Yu-Chao Hsu, et al. Gated qkan-fwp: Scalable quantum-inspired sequence learning. arXiv preprint arXiv:2605.06734, 2026.
[186] Sokea Sang, Leanghok Hour, and Youngsun Han. Learning-optimized qubit mapping and reuse to minimize intercore communication in modular quantum architectures. Physical Review Applied, 25(5):054023, 2026.
[187] Chaitanya K Joshi. Transformers are graph neural networks. arXiv preprint arXiv:2506.22084, 2025.
[188] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the AAAI conference on artificial intelligence, volume 32, 2018.
[189] Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample efficient continuous control. Advances in neural information processing systems, 37:113038–113071, 2024.
[190] Preston Fu, Oleh Rybkin, Zhiyuan Paul Zhou, Michal Nauman, Pieter Abbeel, Sergey Levine, and Aviral Kumar. Compute-optimal scaling for value-based deep rl. Advances in Neural Information Processing Systems, 38:160363–160402, 2026.
[191] Pennylane. Top 20 molecules for quantum computing. https://pennylane.ai/blog/2024/01/top-20-molecules-for-quantum-computing#2-methylene-chsub2sub, 2025. Accessed: 2026-06-16.
[192] Laia Coronas Sala and Parfait Atchade-Adelemou. Qmprot: A comprehensive dataset of quantum properties for proteins. arXiv preprint arXiv:2505.08956, 2025.
[193] Thomas Lubinski, Sonika Johri, Paul Varosy, Jeremiah Coleman, Luning Zhao, Jason Necaise, Charles H Baldwin, Karl Mayer, and Timothy Proctor. Application-oriented performance benchmarks for quantum computing. IEEE Transactions on Quantum Engineering, 4:1–32, 2023.
[194] Michael A Nielsen and Isaac L Chuang. Quantum computation and quantum information. Cambridge university press, 2010.
[195] Nils Quetschlich, Lukas Burgholzer, and Robert Wille. Mqt bench: Benchmarking software and design automation tools for quantum computing. Quantum, 7:1062, 2023.
[196] Nikiforos Paraskevopoulos, David Hamel, Aritra Sarkar, Carmen G Almudever, and Sebastian Feld. Arta: automating design space exploration of spin-qubit architectures: N. paraskevopoulos et al. Quantum Information Processing, 24(6):184, 2025.

[197] Teague Tomesh, Pranav Gokhale, Victory Omole, Gokul Subramanian Ravi, Kaitlin N Smith, Joshua Viszlai, Xin-Chuan Wu, Nikos Hardavellas, Margaret R Martonosi, and Frederic T Chong. Supermarq: A scalable quantum benchmark suite. In 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pages 587–603. IEEE, 2022.

[198] Nicolas PD Sawaya, Daniel Marti-Dafcik, Yang Ho, Daniel P Tabor, David E Bernal Neira, Alicia B Magann, Shavindra Premaratne, Pradeep Dubey, Anne Matsuura, Nathan Bishop, et al. Hamlib: A library of hamiltonians for benchmarking quantum algorithms and hardware. Quantum, 8:1559, 2024.

[199] Aritra Sarkar, Akash Kundu, Matthew Steinberg, Sibasish Mishra, Sebastiaan Fauquenot, Tamal Acharya, Jarosław A Miszczak, and Sebastian Feld. Yaqq: yet another quantum quantizer design space exploration of quantum gate sets using novelty search. New Journal of Physics, 28(4):044504, 2026.

[200] Abhishek Sadhu, Aritra Sarkar, and Akash Kundu. Cutqas: Topology-aware quantum circuit cutting via reinforcement learning. arXiv preprint arXiv:2504.04167, 2025.

[201] Aritra Sarkar. Automated quantum software engineering. Automated Software Engineering, 31(1):36, 2024.