Preference-Conditioned Reinforcement Learning for Noise-Aware Qubit Routing on Constrained Quantum Processors

A single edge-equivariant policy that trades two-qubit gate count, depth and estimated error at compile time

[1]Sonia Mohapatra, [2]Saswat Swain, [3]Chandrasekhar Panda
[1]Researcher, [2]Ph.D Scholar, [3]Researcher
[1,3]Research and Development Division,
Tekstein Solutions Private Limited, Bhubaneswar, Odisha, India
[2]KSOM — KIIT Deemed to be University,
Bhubaneswar, Odisha, India
[1]smctc98@gmail.com, [2]saswat.mck@gmail.com, [3]contact@chandrasekharpanda.com

Abstract—Compiling a logical quantum circuit onto a processor with restricted qubit connectivity requires inserted SWAP operations, and on present-day hardware the cost of each insertion depends on which physical coupling carries it, on how much it lengthens the critical path and on how many two-qubit gates it adds. Routing heuristics minimise a single proxy, usually the SWAP count, and treat calibration data as an afterthought. This paper formulates routing as a Markov decision process whose reward is the negative increment of a scalarised cost over two-qubit gate count, two-qubit depth and an error-equivalent gate count computed from per-edge and per-qubit calibration data, and trains one proximal-policy-optimisation agent conditioned on the preference vector, so that the trade-off surface is reachable at compile time without retraining. The policy scores every coupling edge with a shared network and is therefore independent of processor size: a single agent trained on random circuits transfers to a 27-qubit heavy-hexagon lattice, a 16-qubit grid and a 16-qubit chain. The evaluation is deliberately strict: 40 quantum Fourier transform, Grover and random circuits, three topologies, 5 calibration snapshots, a noise-aware initial layout shared with every baseline, identical post-routing optimisation, paired Wilcoxon tests with Holm correction, and verification of every small output against the logical unitary. Against SABRE configured with its decay heuristic and 20 trials, the greedy-decoded agent routes with a median two-qubit gate count of +1.5% and a median estimated success probability of −0.5% ($p = 0.61$), i.e. parity rather than superiority, while outperforming the one-step heuristic built on the same features by +4.6% gates; sampling four decodes gives −1.2% depth and +0.3% ESP at equal gate count ($p = 0.002$). The preference vector changes the routing of 66% of configurations in the intended direction over a narrow range, and a density-matrix simulation of 1869 compiled circuits confirms that the error model used in the reward ranks circuits consistently with simulated fidelity (within-circuit Spearman rank correlation of 0.91). Two negative findings are reported: calibration features yield no measurable gain at a balanced preference, and the full Qiskit optimisation-level-3 pipeline remains 15.6% ahead on gate count through gate-level passes that routing cannot supply.

Index Terms—quantum circuit compilation, qubit routing, reinforcement learning, proximal policy optimisation, multi-objective optimisation, noise-aware compilation, quantum software engineering, SWAP insertion.

I. Introduction

Quantum processors of the current generation execute two-qubit gates only between physically coupled qubits, and the coupling graphs of superconducting devices are sparse: a heavy-hexagon lattice has average degree close to two, a square lattice at most four. A logical circuit written against all-to-all connectivity must therefore be transformed before execution. The transformation has two parts, an initial placement of logical qubits on physical qubits (layout) and a sequence of SWAP insertions that moves qubits into adjacency when a gate demands it (routing). Each SWAP decomposes into three CNOT gates, so routing directly inflates the two-qubit gate count, which is the dominant source of infidelity on hardware whose two-qubit error rates sit between 0.3% and 5% while single-qubit errors are an order of magnitude smaller [1], [5].

Three features of the problem make the standard heuristic treatment unsatisfactory. First, the objective is not scalar. A SWAP inserted on a low-error coupling costs less than one inserted on a poor coupling, a SWAP that extends the critical path costs decoherence time that a parallel SWAP does not, and the eventual quantity of interest, the probability that the circuit returns a correct answer, depends on all three quantities jointly. Second, the quantities trade against one another: the shortest SWAP path through the graph may run through the worst coupling, and the lowest-depth schedule may require more SWAPs. Third, calibration data drift from hour to hour, so a compiler that assumes a fixed error landscape will be optimising for a machine that no longer exists by the time the job runs. The widely used SABRE router [3] minimises a distance heuristic with a lookahead term and is agnostic to calibration data; noise-adaptive mappers [4]–[6] incorporate error rates but do so through hand-designed cost functions that fix the trade-off once and for all.

Reinforcement learning (RL) offers a different construction. Routing is naturally sequential, the effect of a SWAP on the remaining circuit is only visible several decisions later, and the quantity being optimised can be made explicit in the reward rather than approximated by a distance heuristic. Prior RL routers [7], [8] have shown that learned policies can match or improve on heuristic SWAP counts, and RL has also been applied to compiler-pass ordering [9] and to gate-level circuit rewriting [10]–[12]. What these works share is a single-objective reward (SWAP count, gate count or depth) and a policy trained for one target device; none of them exposes the trade-off between gate count, depth and calibration-dependent error to the user at compile time.

This paper addresses that gap. The routing problem is cast as a Markov decision process (MDP) in which the agent selects a coupling edge to swap, all gates that become executable are committed automatically, and the reward is the negative increment of a scalarised cost over two-qubit gate count, two-qubit depth and an error-equivalent gate count computed from per-edge and per-qubit calibration data. The preference vector that weights the three objectives is drawn at random during training and supplied to the policy as an input, which turns one trained network into a family of routers indexed by the preference. The policy itself scores each coupling edge with a shared network and pools over edges to form context, so its parameter count does not depend on the processor and a single agent routes onto graphs of different size and shape.

The main contributions are the following.

1) A noise-aware routing MDP with a potential-based reward whose undiscounted return equals the negative change of the scalarised cost, so that return maximisation is exactly cost minimisation (Proposition 1), and whose per-edge state features expose calibration data directly to the policy.
2) A preference-conditioned, edge-equivariant actor–critic trained with proximal policy optimisation (PPO) on randomised circuits, graphs, calibrations and preferences, which routes onto unseen circuits and three hardware topologies without retraining and traces the gate-count/depth/error trade-off surface by varying the preference at inference time.
3) A controlled evaluation against Qiskit's SABRE and BasicSwap routers and against the full optimisation-level-3 preset, with identical initial layouts and identical post-routing optimisation, on quantum Fourier transform (QFT), Grover and random circuits, three topologies and five calibration snapshots, analysed with paired non-parametric statistics and Holm-corrected significance.
4) Independent validation of the reward's error model by density-matrix simulation with depolarising, thermal-relaxation and readout noise, together with ablations that remove calibration features from the state and collapse the reward to a single objective.

The remainder of the paper is organised as follows. Section II reviews related work, Section III states the problem, Sections IV and V develop the MDP, the policy and the mathematical model, Section VI gives the algorithm and its complexity, Section VII describes the simulation setup, Sections VIII–X present results, comparative analysis and key outcomes, and Section XI concludes with limitations and future work.

II. Related Work

Table I groups the works most relevant to the present study. Heuristic routers [3], [13] remain the reference for scalability (exact methods [24] are confined to small instances); noise-aware mappers [4]–[6], [14] established that calibration data improve hardware success rates; learned routers [7], [8], [12] showed that policies trained by RL or tree search can compete with heuristics on SWAP count; and compiler-level RL [9]–[11] optimises pass ordering or gate sequences rather than routing decisions. Timing- and resource-aware mapping [17], the formal statement of the routing problem [18] and exact layout synthesis by SMT solving [24] complete the picture. The common limitations that motivate this work are visible in the last column: each method optimises one proxy, fixes any weighting of error against gate count at design or training time, and, for learned approaches, trains one agent per device.

TABLE I. Representative prior work on qubit mapping, routing and learned compilation

Authors, year	Methodology	Advantages	Limitations / gap
Li et al. [3], 2019	SABRE: bidirectional layout + front-layer SWAP heuristic with lookahead	Scales to large circuits; Qiskit default	Noise-agnostic; single distance proxy; no depth control
Zulehner et al. [13], 2019	A*-based mapping with lookahead for IBM QX	Near-optimal SWAP counts on small circuits	Exponential search; noise-agnostic
Murali et al. [4], 2019	Noise-adaptive mapping by SMT / heuristic solvers	Up to 18× success-rate gain on IBM hardware	Hand-fixed cost; solver time grows with size
Tannu and Qureshi [5], 2019	Variability-aware qubit allocation and movement policies	Hardware gains from variability awareness	Heuristic weighting; no multi-objective control
Nishio et al. [6], 2020	Error-aware beam search routing; introduced the ESP metric	ESP correlates with measured success	Beam width limits scalability; single objective
Niu et al. [14], 2020	Fidelity-aware SWAP / bridge-gate heuristic	Lower error than SABRE on IBM devices	Fixed heuristic; depth not modelled
Pozzi et al. [7], 2022	Deep Q-learning routing with annealed action selection	Beats Qiskit/tket SWAP counts on some circuits	SWAP count only; one agent per device
Sinha et al. [8], 2022	GNN-guided Monte Carlo tree search for routing	Strong SWAP counts; graph generalisation	Search cost per circuit; noise-agnostic
Quetschlich et al. [9], 2023	RL selection of compiler-pass sequences	Device-aware pass ordering; benchmarks [15]	Acts on pass order, not on SWAP decisions

Authors, year	Methodology	Advantages	Limitations / gap
Fösel et al. [10], 2021; Li et al. [11], 2024	RL / learned rewriting of gate sequences	Discovers non-obvious gate-count reductions	Pre-routing optimisation; connectivity not addressed
Kremer et al. [12], 2024	RL synthesis and routing passes in a production transpiler	Practical runtime; Clifford synthesis gains	Single reward per pass; preference fixed at training
Nation and Treinish [16], 2023	Post-compilation re-layout by calibration scoring (VF2)	Cheap, effective, default in Qiskit O3	Does not change routing decisions

Within this landscape the proposed method is distinguished by (i) a reward that is an explicit, calibration-dependent, multi-objective cost rather than a SWAP count, (ii) a preference vector supplied to the policy rather than frozen into the reward, and (iii) a parameterisation that is independent of the processor graph. Multi-objective RL with a preference-conditioned policy has been studied in general settings [19]; its use for quantum routing, where the objectives are physically distinct and the preference is a legitimate user choice, is new to the authors' knowledge.

III. Problem Statement

Let C be a logical circuit on n qubits whose gates have been decomposed into the basis {RZ, SX, X, CX}, and let $G = (V, E)$ be the coupling graph of a processor with $|V| = N \geq n$ physical qubits. A calibration snapshot assigns to every edge $e \in E$ a two-qubit gate error ε_e, to every physical qubit q a single-qubit error ε_q, relaxation and dephasing times $T_{1,q}$, $T_{2,q}$ and a readout error $\varepsilon_{\mathrm{ro},q}$. A compiled circuit C' is a sequence of basis gates on V that implements C up to a final permutation of qubit labels; it is produced by an injective initial layout π_0: $\{0, \ldots, n-1\} \to V$ followed by SWAP insertions that keep every two-qubit gate on an edge of G.

The following issues define the problem addressed here.

- *Conflicting objectives.* The two-qubit gate count N_{2q}, the two-qubit depth D and the expected error Λ of C' are each legitimate targets. They are correlated but not aligned: on a calibrated device the SWAP path of minimum length is not in general the path of minimum error, and parallelising SWAPs to reduce D can increase N_{2q}.
- *Calibration dependence.* Λ depends on which physical edges and qubits carry each gate, so the optimal routing changes whenever the calibration changes. A router must therefore read calibration data at compile time rather than encode it in fixed rules.
- *Delayed consequences.* A SWAP that appears neutral for the gates currently executable can be decisive for gates several layers ahead. Greedy heuristics approximate this with a lookahead window; the quality of the approximation is not controlled.
- *Practical constraints.* Routing runs inside a compiler, so a per-circuit search budget of minutes is unacceptable; the method must also work on graphs of different size and must give users a way to express whether they care more about depth, gate count or error for a given job.

The objective is to find, for a user-specified preference $\mathbf{w} = (w_q, w_d, w_e)$ on the probability simplex, a routed circuit that minimises the scalarised cost $J(\mathbf{w})$ defined in Section V, while remaining within a compile-time budget comparable with heuristic routers and without retraining when the processor, the circuit or the calibration changes.

IV. Proposed Methodology

Fig. 1 gives the architecture of the proposed framework, referred to below as MORC (multi-objective routing compiler). It consists of four functional modules: a preprocessing stage that builds the two-qubit dependency graph and an initial layout; the routing MDP, which maintains the layout, the front layer of executable gates and a per-qubit depth counter; the preference-conditioned actor–critic; and a common finalisation and metric stage that is shared with every baseline so that comparisons isolate the routing decisions.

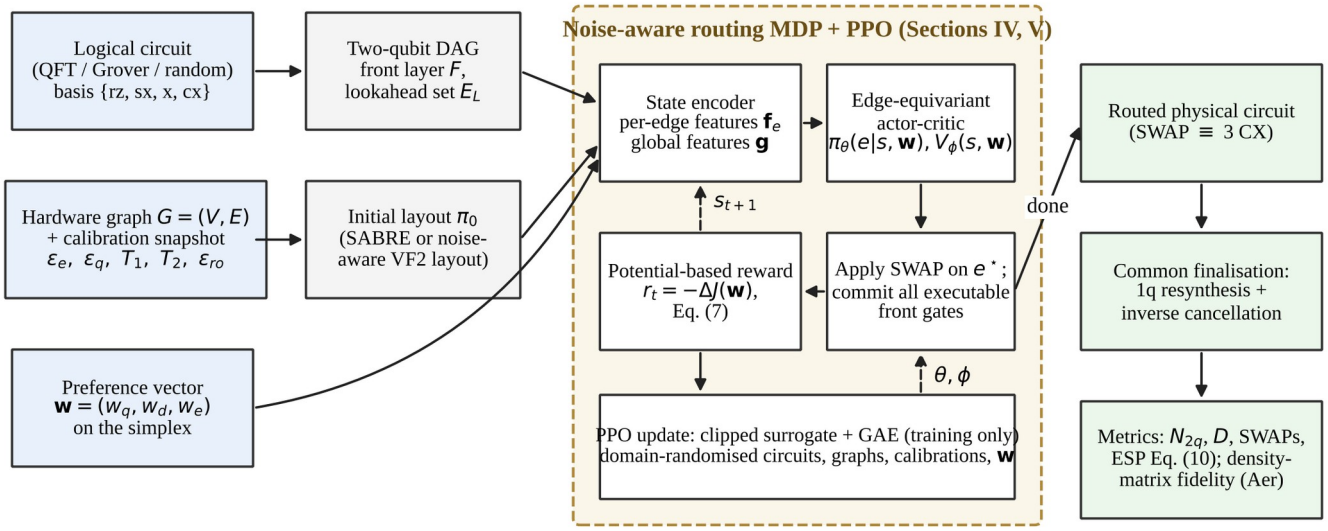


Fig. 1. Block diagram of the MORC framework. Inputs (left) are the logical circuit, the hardware graph with its calibration snapshot and the preference vector; the routing MDP (centre) alternates state encoding, edge selection by the policy, SWAP application with automatic commitment of executable gates, and a potential-based reward; outputs (right) pass through the same finalisation and metric computation as every baseline.

A. Preprocessing

The logical circuit is converted into a directed acyclic graph over its two-qubit gates. Single-qubit gates are attached to the wire segment that precedes the next two-qubit gate on the same qubit and are emitted on the physical qubit currently hosting that wire when the gate is committed, so they do not influence routing but are counted in depth and error. The front layer F is the set of two-qubit gates whose predecessors have all been committed; the lookahead set E_L holds the next $L = 20$ gates reachable from F in topological order. An initial layout is obtained either from SABRE's bidirectional layout pass (noise-agnostic) or from the noise-aware VF2 post-layout of Qiskit's optimisation-level-3 preset [16]; both regimes are evaluated because the layout is the dominant lever for error and must not be confounded with routing.

B. Routing MDP

A state s_t comprises the current layout π_t, the committed-gate set, the per-physical-qubit layer counters τ_q used to track two-qubit depth, and the calibration snapshot. An action is a coupling edge $e = (u, v) \in E$ on which a SWAP is applied; the admissible set $A(s_t)$ is restricted to edges with at least one endpoint hosting a logical qubit that participates in $F \cup E_L$, which removes provably useless moves and shrinks the action space on large graphs. After each SWAP every gate in F whose two qubits have become adjacent is committed, the front layer is advanced, and the process repeats until all gates are committed. Episodes are truncated, with a penalty, if the number of SWAPs exceeds four times the number of two-qubit gates plus ten; the penalty is never triggered by the trained agents on the evaluation suite.

C. State Encoding

The policy sees the state through a feature vector $\mathbf{f}_e$ for every edge and a global vector $\mathbf{g}$ (Table II). The two distance-reduction features are the change in the summed shortest-path distance over F and over E_L that the SWAP on e would produce; they carry the same information as the SABRE heuristic but are presented as inputs rather than combined by a fixed rule. The noise features expose the edge's error weight relative to the device mean and the smaller of the two endpoint T_1 values. The timing features give the earliest layer at which a SWAP on e could start relative to the current minimum and maximum qubit times, which is what the agent needs to reason about depth. The global vector includes the preference $\mathbf{w}$, so that the same network can act differently under different preferences.

TABLE II. STATE FEATURES SUPPLIED TO THE POLICY FOR EACH COUPLING EDGE E = (U, V) AND GLOBALLY

Symbol	Definition	Role
f_1	front-layer distance reduction $\Delta F_e / \lvert F\rvert$, Eq. (4)	routing progress
f_2	lookahead distance reduction $\Delta E_e / \lvert E_L\rvert$	delayed progress
f_3	$\eta_e / \bar{\eta}$, edge error weight relative to device mean	noise
f_4, f_{10}	endpoint hosts a qubit of F / of E_L (indicator)	relevance
f_5	edge was the previous SWAP (indicator)	anti-oscillation
f_6, f_7	(max, min of τ_u, $\tau_v - \tau_{\min}$) / $\tau_{\max}$	depth / timing
f_8	$\min(T_{1,u}, T_{1,v})$ / 100 µs	decoherence
f_9	both endpoints occupied (indicator)	layout density
g	fraction committed, $\lvert F\rvert/n$, mean F and E_L distances, SWAPs per gate, **w**	global context + preference

D. Edge-Equivariant Actor–Critic

Every edge is embedded by the same two-layer network applied to $[\mathbf{f}_e; \mathbf{g}]$; mean- and max-pooling over admissible edges gives a context vector that is concatenated with $\mathbf{g}$ and transformed once more; the actor's logit for edge e is a function of its own embedding and the context, masked to $A(s_t)$ and normalised by a softmax, Eq. (5). The critic reads the context only. Because every edge is processed by shared weights and the pooling is symmetric, the architecture is equivariant to relabelling of edges and defined for any number of edges, which is what permits a single agent to serve a 15-

edge chain and a 28-edge heavy-hexagon lattice. The hidden width is 96 and the network has 58,562 parameters.

E. Reward and Training

The reward at step t is the negative increment of the scalarised cost, Eq. (7). Training uses PPO [20] with generalised advantage estimation on episodes drawn from a pool of random circuits (4–14 qubits, 4–18 layers) placed on each of the three topologies, with a freshly sampled calibration snapshot, a noise scale drawn from [0.6, 1.5] and a preference drawn from the uniform distribution on the simplex at every episode. Domain randomisation over all four factors is what makes the policy transferable; no benchmark circuit from the evaluation suite is seen during training.

F. Finalisation and Metrics

Every routed circuit, whether produced by MORC or by a baseline, is passed through the same Qiskit optimisation-level-1 pipeline with routing disabled, which decomposes SWAPs into three CX gates, resynthesises single-qubit runs and cancels adjacent inverse pairs. The reported two-qubit gate count, depth, estimated success probability (ESP) and circuit duration are computed on the resulting physical circuit, and for small circuits the physical circuit is additionally simulated with a density-matrix noise model (Section VII).

V. Mathematical Modelling

A. Layout Dynamics and Executability

Let π_t: $\{0, \dots, n-1\} \to V$ be the injective layout at step t and let σ_e denote the transposition of the two endpoints of edge $e = (u, v)$. A SWAP on e updates the layout as

$$\pi_{t+1} = \sigma_e \circ \pi_t, \quad e \in A(s_t) \tag{1}$$

and a two-qubit gate $g = (a, b)$ is executable at step t if and only if $(\pi_t(a), \pi_t(b)) \in E$, i.e. the shortest-path distance $d(\pi_t(a), \pi_t(b)) = 1$. Gate commitment is deterministic given the layout, so the stochasticity of the process lies entirely in the policy.

B. Error-Equivalent Gate Count

For an edge or qubit with error rate ε, define the error weight $\eta = -\ln(1 - \varepsilon)$. Weights add along a circuit, so the gate contribution to the log of the estimated success probability is additive,

$$\Lambda_{\text{gate}} = \Sigma_{g \in C'} \eta_g = -\Sigma_{g \in C'} \ln(1 - \varepsilon_g) \tag{2}$$

where g ranges over all CX, SX and X gates of the physical circuit, each taking the error of the edge or qubit it occupies. Dividing by the device mean edge weight $\bar{\eta} = |E|^{-1} \Sigma_e \eta_e$ yields an *error-equivalent gate count* $\Lambda_{\text{gate}}/\bar{\eta}$, i.e. the number of average-quality CX gates that would accumulate the same error; this normalisation places the error objective on the same scale as the gate count so that preference weights are interpretable.

C. Scalarised Multi-Objective Cost

With N_{2q} the number of two-qubit gates (committed CX plus three per SWAP), D the two-qubit depth produced by the layer counters, and $\mathbf{w}$ on the simplex, the cost of a partially or fully routed circuit is

$$J(\mathbf{w}) = w_q N_{2q} + w_d D + w_e \Lambda_{\text{gate}} / \bar{\eta} \tag{3}$$

The depth counters follow the recurrence $\tau_u, \tau_v \leftarrow \max(\tau_u, \tau_v) + c_g$ when a gate occupying u and v is committed, with $c_g = 1$ for CX and 3 for SWAP, and $\tau_q \leftarrow \tau_q + 1$ for a single-qubit gate; $D = \max_q \tau_q$. By the standard weighted-sum result of multi-objective optimisation [21], for any $\mathbf{w}$ with strictly positive components every minimiser of $J(\mathbf{w})$ is Pareto-optimal for the vector objective $(N_{2q}, D, \Lambda_{\text{gate}})$; sweeping $\mathbf{w}$ therefore traces a subset of the Pareto front.

D. Distance-Reduction Features

The first two state features are potential differences of the SABRE distance function under the candidate swap,

$$\Delta F_e = \Sigma_{(a,b) \in F} \left[d(\pi_t(a), \pi_t(b)) - d(\sigma_e \pi_t(a), \sigma_e \pi_t(b)) \right] \tag{4}$$

with ΔE_e defined identically over E_L. Both are computed from a precomputed all-pairs distance matrix in $O(|E|\,(|F| + L))$ time per step.

E. Policy and Value Functions

Let $\mathbf{h}_e = \varphi_\theta([\mathbf{f}_e; \mathbf{g}])$ be the edge embedding and $\mathbf{c} = \psi_\theta([\text{mean}_{e \in A} \mathbf{h}_e; \max_{e \in A} \mathbf{h}_e; \mathbf{g}])$ the pooled context. The policy and the value estimate are

$$\pi_\theta(e \mid s, \mathbf{w}) = \mathbb{1}[e \in A(s)] \exp(\ell_\theta(\mathbf{h}_e, \mathbf{c})) / \Sigma_{e'} \mathbb{1}[e' \in A(s)] \exp(\ell_\theta(\mathbf{h}_{e'}, \mathbf{c})), \quad V_\phi(s, \mathbf{w}) = v_\phi(\mathbf{c}) \tag{5}$$

where ℓ_θ and v_ϕ are two-layer perceptrons. Because the preference enters through $\mathbf{g}$, Eq. (5) defines a continuum of policies, one for each $\mathbf{w}$, with shared parameters; the value function is likewise preference-dependent, as required for consistent advantage estimation under a changing reward.

F. Potential-Based Reward

Let J_t denote the cost of the partially routed circuit after the commitments that follow action a_t. The reward is

$$r_t = -(J_t - J_{t-1}) - \rho_{\text{trunc}} \, \mathbb{1}[\text{truncated}] \tag{6}$$

where ρ_{trunc} is the truncation penalty. Since the episode cost is a telescoping sum, the undiscounted return satisfies

$$\Sigma_{t=0}^{T-1} r_t = J_0 - J_T \tag{7}$$

Proposition 1. *For any fixed* $\mathbf{w}$ *and any complete episode, the undiscounted return equals* $J_0 - J_T$*; hence a policy that maximises expected undiscounted return minimises the expected final cost* $E[J_T(\mathbf{w})]$*, and the per-step shaping introduces no change of optimal policy. Proof.* Eq. (7) follows by cancellation of intermediate terms in Eq. (6); J_0 is fixed by the initial layout and independent of the policy. The reward is of the potential-based form $r_t = \Phi(s_t) - \Phi(s_{t+1})$ with $\Phi = -J$, which by [22] preserves the set of optimal policies of the sparse terminal reward $-J_T$. ∎ In training a discount of 0.995 is used for variance reduction, which perturbs the exact identity by a negligible amount over episodes of fewer than 300 steps.

G. Learning Objective

PPO maximises the clipped surrogate

$$L^{\text{CLIP}}(\theta) = \mathbb{E}[\, \min(\, \rho_t \hat{A}_t, \, \text{clip}(\rho_t, 1 - \epsilon, 1 + \epsilon)\, \hat{A}_t \,) \,] \tag{8}$$

with $\rho_t = \pi_\theta(a_t|s_t, \mathbf{w})/\pi_{\theta\text{old}}(a_t|s_t, \mathbf{w})$, generalised advantage $\hat{A}_t$ computed with $(\gamma, \lambda) = (0.995, 0.95)$, clipping $\epsilon = 0.2$, an entropy bonus of 0.01 and a Huber value loss weighted 0.5; rewards are scaled by 0.1 before advantage estimation. Four epochs of minibatch size 256 are run over each rollout of 2048 steps with Adam at learning rate 3×10^{-4}.

H. Estimated Success Probability

Following [6], the ESP of a finalised physical circuit of duration T_{circ} (as-soon-as-possible schedule with 300 ns CX, 35 ns SX/X, 700 ns readout) is

$$\text{ESP} = \exp[\, -(\Lambda_{\text{gate}} + T_{\text{circ}} \Sigma_{q \in Q} \, 1/T_{1,q} + \Sigma_{q \in Q} \, \eta_{\text{ro},q}) \,] \qquad (9)$$

where Q is the set of physical qubits occupied by the circuit at any time. Only Λ_{gate} enters the reward during routing; the decoherence and readout terms are reported for completeness and are identical in form for all methods. Section VIII-F tests the ESP against density-matrix simulation.

VI. Algorithm

Algorithm 1 gives the inference procedure; Algorithm 2 the training loop. Both are implemented without any device-specific code path.

Algorithm 1: MORC routing (inference)

Input: logical circuit C, hardware G, calibration (ε_e, ε_q, T_1), layout π_0, preference $\mathbf{w}$, trained parameters θ
Build two-qubit DAG; initialise F, counters $\tau_q \leftarrow 0$, $N_{2q} \leftarrow 0$, $\Lambda \leftarrow 0$
Commit every executable gate in F; update F, τ, N_{2q}, Λ
while uncommitted gates remain **do**
  Compute E_L, admissible set $A(s)$, features $\{\mathbf{f}_e\}$ and $\mathbf{g}$ (Table II)
  $e^* \leftarrow \arg\max_{e \in A} \pi_\theta(e \mid s, \mathbf{w})$ ▷ greedy decoding
  Apply SWAP on e^*: $\pi \leftarrow \sigma_{e^*} \circ \pi$; N_{2q} += 3; Λ += $3\eta_{e^*}$; update τ
  Commit every executable gate in F; update F, τ, N_{2q}, Λ
end while
Emit trailing single-qubit gates; run common finalisation
Output: physical circuit C', final layout π_T

Algorithm 2: PPO training with domain randomisation

Initialise θ, ϕ; build pool of random circuits with SABRE layouts on each topology
for iteration = 1 … K **do**
  for step = 1 … 2048 **do**
    If episode ended: sample circuit, topology, calibration seed, noise scale $\in$ [0.6, 1.5], $\mathbf{w} \sim$ Dir(1,1,1); reset
    Sample $a_t \sim \pi_\theta(\cdot \mid s_t, \mathbf{w})$; store ($s_t$, a_t, log π, V_ϕ); step environment; receive r_t by Eq. (6)
  end for
  Compute $\hat{A}_t$ by GAE; optimise Eq. (8) with entropy and Huber value terms for 4 epochs
end for
Output: θ, ϕ

A. Complexity

Per routing step the feature computation costs $O(|E|\,(|F| + L))$ distance look-ups and the network forward pass $O(|E|\,H^2)$ with hidden width $H = 96$; the number of steps equals the number of inserted SWAPs S. The inference time of a routing episode is therefore $O(S\,|E|\,(|F| + L + H^2))$, linear in the number of SWAPs and in the number of coupling edges, and independent of the number of physical qubits except through $|E|$. Memory is $O(|E|\,H)$ for activations plus the $O(N^2)$ distance matrix, which is precomputed once per device. For comparison, SABRE's per-step cost is $O(|E|\,(|F| + L))$ with a much smaller constant, and LookaheadSwap grows exponentially in its search depth. Training costs 6×10^5 environment steps, 20–49 min of wall-clock time on one CPU core, and is a one-off expense not repeated per circuit, device or calibration.

VII. Simulation Setup

A. Software and Hardware

All experiments were implemented in Python 3.13 with Qiskit 2.6.1 for circuit construction, decomposition, the SABRE, BasicSwap and LookaheadSwap routers, the optimisation-level-3 preset and the common finalisation pass; Qiskit Aer 0.18.0 [25] for density-matrix simulation; PyTorch 2.14.1 (CPU) for the actor–critic and a self-contained PPO implementation whose update follows the clipped-surrogate form of Stable-Baselines3; NumPy, SciPy and pandas for analysis. Training and evaluation ran on a two-core x86-64 Linux container with 7 GB of memory, one thread per run; no GPU was used. Statistical tests were computed with SciPy; MATLAB scripts that re-implement the paired Wilcoxon analysis (signrank, bootci) and the Pareto plot are provided in the reproducibility bundle for cross-checking but were not executed in the authors' build environment, and the numbers reported here are those of the Python pipeline. All random seeds are fixed and listed in the bundle.

B. Hardware Models and Calibration Snapshots

Three coupling graphs are used (Fig. 2): the 27-qubit heavy-hexagon lattice of IBM Falcon-class processors (28 edges), a 4 × 4 square lattice (24 edges) and a 16-qubit linear chain (15 edges). Calibration snapshots are synthetic and seeded. Two-qubit errors follow a log-normal body with median 0.9% and geometric standard deviation ≈ 2.0, and 12% of edges are replaced by outliers drawn uniformly from [4%, 10%], reproducing the heavy right tail of public calibration tables [4], [5]; single-qubit errors are log-normal with median 3×10^{-4}; T_1 is log-normal with median 110 µs and $T_2 \le 2T_1$ with median 90 µs; readout errors have median 1.5%. Across the five evaluation seeds the edge error has median 1.0%, mean 1.9%, 90th percentile 5.6% and maximum 9.9%. Gate durations are 300 ns (CX), 35 ns (SX, X), 0 ns (virtual RZ) and 700 ns (readout). Table III lists the remaining parameters.

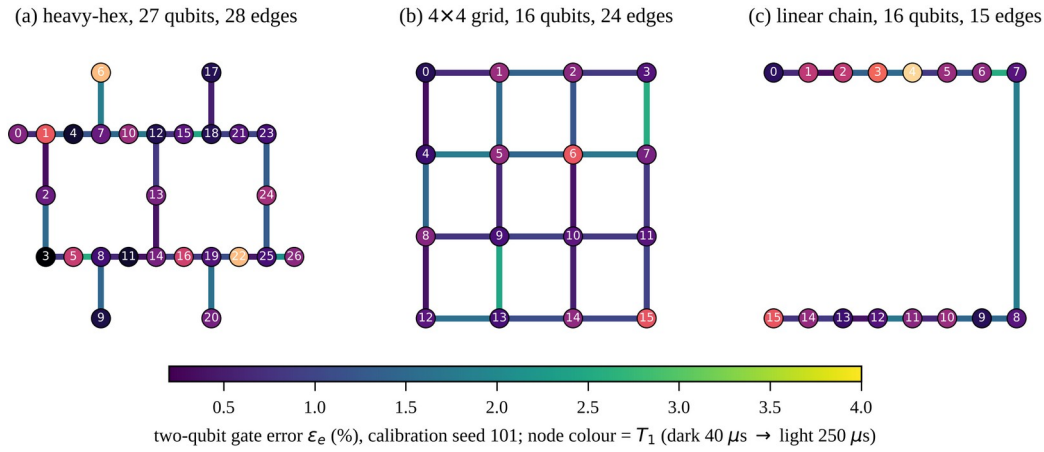


Fig. 2. Coupling graphs used in the evaluation with one synthetic calibration snapshot (seed 101): (a) 27-qubit heavy-hexagon lattice, (b) 4 × 4 square lattice, (c) 16-qubit linear chain. Edge colour gives the two-qubit gate error ε_e; node colour gives T_1.

TABLE III. SIMULATION AND TRAINING PARAMETERS

Parameter	Value
Training circuits	900 random circuits, 4–14 qubits, 4–18 layers (300 per topology)
Calibration during training	new seed per episode; noise scale ∈ U[0.6, 1.5]
Preference during training	**w** ~ Dirichlet(1, 1, 1)
Lookahead L; truncation	20 gates; 4 × |gates| + 10 SWAPs
Network	edge-shared MLP, width 96, 58,562 parameters
PPO	rollout 2048, 4 epochs, batch 256, γ = 0.995, λ = 0.95, ϵ = 0.2, lr 3×10^{-4}
Training budget	6×10^5 steps per run; 3 seeds + 2 ablation runs
Evaluation suite	40 circuits × 3 topologies × 5 calibration seeds
Evaluation preference	**w** = (1/3, 1/3, 1/3) unless stated; greedy decoding
Baselines	SABRE (decay heuristic, 20 trials), BasicSwap, Greedy-MO, Qiskit O3; MORC best-of-4
Noisy simulation	Aer density matrix, ≤ 9 active qubits, 2×10^4 shots

C. Benchmark Circuits

The evaluation suite (Table IV) contains QFT circuits on 4–16 qubits applied to a Fourier product state so that the ideal output is a single basis state, Grover search circuits [23] on 3–6 qubits with the optimal number of iterations and an ancilla-free multi-controlled-Z oracle, and 30 random circuits generated by Qiskit's random_circuit with up to two-qubit operations on 6, 10 and 14 qubits at depths 8 and 16. All circuits are decomposed into the {RZ, SX, X, CX} basis before routing; the random family spans 10–191 CX gates and Grover-6 is the largest circuit with 984 CX gates. None of these circuits is in the training pool.

TABLE IV. EVALUATION SUITE (LOGICAL, AFTER BASIS DECOMPOSITION)

Family	Instances	Qubits	CX gates	Depth
QFT	6	4–16	12–240	24–120
Grover	4	3–6	24–984	55–1295
random	30	6–14	10–191	25–120

D. Protocol

For each circuit, topology and calibration seed, an initial layout is computed once per regime: the SABRE layout (noise-agnostic) and the VF2 post-layout returned by the O3 preset against a calibrated Target (noise-aware). Every router is then run from the same layout, and every routed circuit passes through the same finalisation. MORC is evaluated with the three independently trained seeds and reported as their mean per configuration, with the per-seed spread given in Section VIII-C. Paired comparisons use the Wilcoxon signed-rank test over the 600 (circuit, topology, calibration) configurations, two-sided, with Holm correction across the 72 tests of each table; effect sizes are reported as the median relative change with a 2000-sample bootstrap 95% confidence interval. The ESP of Eq. (9) is validated against density-matrix simulation for every compiled circuit of the four families' smallest members (3–6 logical qubits) whose physical footprint does not exceed nine qubits; the noise model applies a two-qubit depolarising channel with parameter $4\varepsilon_e/3$ (so that its average gate infidelity equals ε_e), thermal relaxation on every gate with the qubit's T_1, T_2 and the gate duration, a single-qubit depolarising channel of infidelity ε_q, and symmetric readout error $\varepsilon_{\mathrm{ro},q}$; fidelity is the Hellinger fidelity between the ideal and the simulated output distributions over the logical qubits.

VIII. RESULTS AND DISCUSSION

A. Training Behaviour

Fig. 3 shows the learning curves of the three seeds and the two ablation runs. The SWAP-per-gate ratio of the random training circuits falls from 3.67 at the start of training to 0.38, 0.37 and 0.37 for seeds 0–2 after 6×10^5 steps, while every episode terminates by completion rather than truncation from roughly 1.5×10^5 steps on. The calibration-blind run reaches 0.38 and the gate-count-only run 0.36; their curves lie within the seed spread of the full agent, which is expected since the SWAP count is the dominant term of every reward variant. Continuing seed 0 for a further 1.2×10^6 steps at a reduced learning rate (0.36 SWAPs per gate at the end) is reported separately below as a training-budget variant. Training wall-clock time was 20–49 min per 6×10^5-step run on one core, the spread reflecting machine load from concurrent runs.

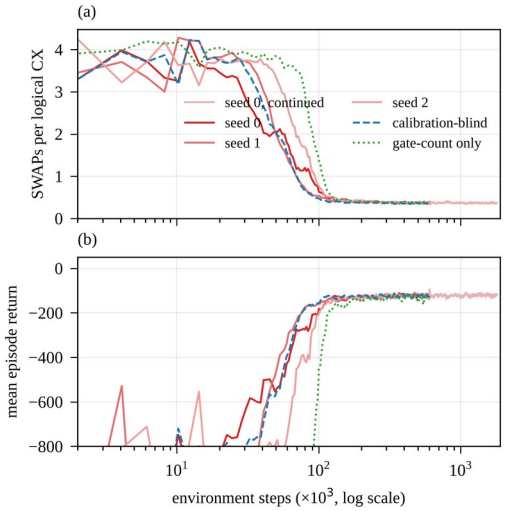


Fig. 3. Learning curves (8-iteration moving average). (a) SWAPs inserted per logical two-qubit gate on the training distribution; (b) mean undiscounted episode return, equal to the negative scalarised cost change by Eq. (7). The three seeds of the full agent are shown in red, the calibration-blind ablation dashed and the gate-count-only ablation dotted.

B. Routing Quality Under a Shared Noise-Aware Layout

Table V reports mean metrics per benchmark family for the noise-aware layout regime, and Fig. 4 the per-configuration ratios to SABRE with bootstrap confidence intervals. Under the noise-aware layout the greedy-decoded agent is statistically indistinguishable from SABRE on the error objective and marginally behind it on the gate objective: its median two-qubit gate count is +1.5% relative to SABRE (bootstrap 95% CI [+0.6%, +2.2%]), the median depth +1.2% and the median ESP −0.5% ($p = 0.61$). The picture differs by family and by size. On the QFT circuits the agent uses fewer two-qubit gates than SABRE on average (196.1 against 197.5; best-of-4: 188.1), with the advantage concentrated at 6–8 qubits (98.4 against 109.8 gates at $n = 8$, all topologies); Fig. 4(d) shows that on the heavy-hexagon lattice the greedy decoder falls behind SABRE from 10 qubits upwards and needs best-of-8 sampling to match it at 16 qubits (617, 542 and 550 gates), i.e. the policy generalises less well to circuits larger than its 14-qubit training range; on the random family, whose two-qubit interaction graphs are unstructured, SABRE retains a small advantage (162.0 against 169.4 gates), and on Grover the two are within one percent (685.3 against 699.4). BasicSwap, included as a sanity bound, inserts +54.5% more gates than SABRE and loses 90.5% of ESP. Greedy-MO, which applies the same distance and noise features as the agent but with a fixed one-step rule, is clearly worse than both (+4.6% gates, −8.6% ESP), so the learned value of future consequences, not the feature set, is what brings the agent to parity with SABRE's lookahead heuristic. All 2880 agent outputs on circuits with at most six logical qubits passed the statevector equivalence test of Section VII-D (0 failed); 2 agent routings out of 3600 in the main evaluation exhausted the SWAP budget (rand_n14_d8_4 on line16, sabre layout, morc_s0, calibration seed 101; rand_n14_d8_4 on line16, sabre layout, morc_s0, calibration seed 102) and are excluded from the paired rows.

TABLE V. MEAN METRICS PER BENCHMARK FAMILY, NOISE-AWARE INITIAL LAYOUT, 3 TOPOLOGIES × 5 CALIBRATION SEEDS (SWAPS ARE NOT DEFINED FOR THE FULL O3 PIPELINE)

Method	2q gates	Depth	SWAPs	ESP	2q gates	Depth	SWAPs	ESP	2q gates	Depth	SWAPs	ESP
	QFT (6 circuits)				Grover (4 circuits)				Random (30 circuits)			
BasicSwap	554.5	256.3	153.7	0.125	957.6	1055.7	204.8	0.104	256.7	235.5	62.5	0.077
SABRE	197.5	137.1	34.7	0.173	685.3	878.2	113.4	0.139	162.0	132.3	30.7	0.128
Greedy-MO	223.8	164.2	50.4	0.185	746.0	923.2	144.0	0.145	160.1	140.9	32.3	0.136
MORC (mean of 3 seeds)	196.1	147.1	39.9	0.179	699.4	860.7	132.0	0.139	169.4	137.0	34.5	0.128
MORC best-of-4 (seed 0)	188.1	138.7	38.3	0.183	693.0	851.8	130.5	0.145	165.4	131.7	33.3	0.130
Qiskit O3 (full pipeline)	149.9	140.1	–	0.191	581.8	910.9	–	0.151	145.3	121.5	–	0.147

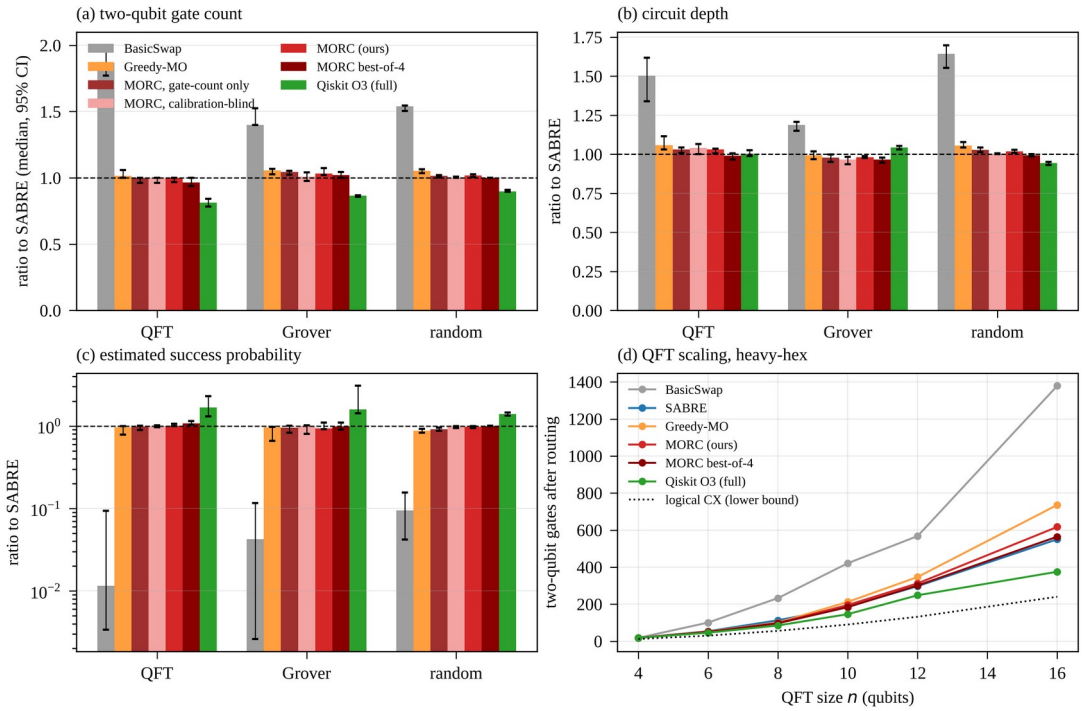


Fig. 4. Per-configuration ratios to SABRE under the noise-aware layout (median with bootstrap 95% CI; values below 1 are better in (a), (b), above 1 better in (c)): (a) two-qubit gate count, (b) depth, (c) ESP (log scale); (d) QFT scaling on the heavy-hexagon lattice, mean over calibration seeds, with the logical CX count as a lower bound.

C. Paired Statistical Comparison

Table VI gives the paired comparison against SABRE under both layout regimes. The sign of every small difference is confirmed by the paired test, which has considerable power at n = 600: the agent's gate count is higher than SABRE's in 332 of 600 configurations and lower in 154 (median +1.5%, $p < 0.001$), whereas the ESP difference is not significant after Holm correction ($p = 0.61$). Decoding four candidates and keeping the lowest-cost one removes the gate-count deficit (median +0.0%, wins/ties/losses 222/122/256) and turns depth and ESP into small but significant gains (−1.2% depth, $p = 0.002$; +0.3% ESP, $p = 0.002$, 313/20/267). With eight candidates the gains grow to +0.0% gates, −2.6% depth and +3.0% ESP ($p < 0.001$). Training seed 0 for a further 1.2×10^6 steps changes the greedy-decoded medians to +0.9% gates and −0.4% ESP, which indicates that the residual gap to SABRE is not closed by budget alone at this network size. Under the noise-agnostic SABRE layout the same ordering holds (+1.7% gates, −0.4% ESP for the greedy decoder; +1.3% ESP, $p < 0.001$, for best-of-4), so the conclusions do not depend on which layout is used. The seed-to-seed spread of the greedy decoder is small on gate count (median change +1.3%, +0.5% and +0.9% for the three seeds) but not on ESP (median change −5.4%, −0.0% and −0.3%): seed 0, the seed used for the best-of-k and ablation rows, is the weakest of the three on the error objective, so the best-of-4 gains above are measured against the least favourable starting point.

TABLE VI. PAIRED COMPARISON WITH SABRE (SAME CIRCUIT, TOPOLOGY, CALIBRATION AND INITIAL LAYOUT): MEDIAN RELATIVE CHANGE WITH BOOTSTRAP 95% CI, HOLM-CORRECTED WILCOXON P; W/T/L = WINS/TIES/LOSSES IN ESP

Method	2q gates	p	Depth	p	ESP	p	W/T/L
Noise-aware layout (n = 600)							
BasicSwap	+54.5% [+52.5%, +56.3%]	p < 0.001	+55.2% [+51.0%, +58.3%]	p < 0.001	−90.5% [−94.9%, −85.6%]	p < 0.001	11/18/571
Greedy-MO	+4.6% [+3.4%, +5.6%]	p < 0.001	+4.5% [+3.6%, +5.8%]	p < 0.001	−8.6% [−12.7%, −4.5%]	p < 0.001	164/20/372
MORC (mean of 3 seeds)	+1.5% [+0.6%, +2.2%]	p < 0.001	+1.2% [+0.3%, +2.1%]	p < 0.001	−0.5% [−3.5%, −0.0%]	p = 0.61	248/19/333
MORC best-of-4 (seed 0)	+0.0% [+0.0%, +0.0%]	p = 0.004	−1.2% [−2.1%, +0.0%]	p = 0.002	+0.3% [+0.0%, +0.6%]	p = 0.002	313/20/267
MORC 1.8 M steps (seed 0)	+0.9% [+0.0%, +1.9%]	p < 0.001	+1.2% [+0.0%, +2.3%]	p < 0.001	−0.4% [−2.8%, −0.0%]	p = 0.81	251/20/329
MORC calibration-blind	+0.0% [+0.0%, +0.8%]	p < 0.001	+0.0% [+0.0%, +0.7%]	p = 0.77	−0.5% [−3.4%, −0.0%]	p = 1.00	252/20/328
MORC gate-count-only	+1.3% [+0.0%, +2.1%]	p < 0.001	+2.1% [+0.9%, +3.0%]	p < 0.001	−5.4% [−8.5%, −2.6%]	p < 0.001	203/20/377
Qiskit O3 (full pipeline)	−11.2% [−11.8%, −10.4%]	p < 0.001	−4.4% [−4.7%, −3.5%]	p < 0.001	+40.2% [+33.0%, +46.0%]	p < 0.001	562/0/38
SABRE layout (n = 600)							
BasicSwap	+54.1% [+51.5%, +54.8%]	p < 0.001	+47.2% [+43.0%, +55.2%]	p < 0.001	−94.4% [−97.4%, −91.0%]	p < 0.001	5/20/575
Greedy-MO	+7.1% [+6.4%, +9.2%]	p < 0.001	+5.9% [+4.3%, +8.0%]	p < 0.001	−12.7% [−20.3%, −6.1%]	p < 0.001	146/25/340
MORC (mean of 3 seeds)	+1.7% [+0.5%, +2.5%]	p < 0.001	+2.1% [+1.3%, +2.9%]	p < 0.001	−0.4% [−1.7%, −0.0%]	p = 0.22	254/22/324
MORC best-of-4 (seed 0)	+0.0% [+0.0%, +0.0%]	p = 0.003	+0.0% [−1.0%, +0.0%]	p = 0.51	+1.3% [+0.1%, +3.7%]	p < 0.001	344/25/231
MORC 1.8 M steps (seed 0)	+0.6% [+0.0%, +1.8%]	p < 0.001	+0.6% [+0.0%, +1.6%]	p = 0.002	−0.4% [−1.5%, −0.0%]	p = 0.22	247/25/328
MORC calibration-blind	+0.0% [+0.0%, +0.0%]	p < 0.001	+0.0% [+0.0%, +0.7%]	p = 0.22	−0.4% [−2.3%, +0.0%]	p = 0.19	253/25/322
MORC gate-count-only	+0.2% [+0.0%, +1.0%]	p < 0.001	+0.9% [+0.0%, +2.2%]	p < 0.001	−6.0% [−10.5%, −2.1%]	p < 0.001	202/25/373

D. Full-Stack Reference and Ablations

The optimisation-level-3 preset is not a routing method but a complete compiler, and it is included to show where routing sits in the overall budget. Against it the agent's circuits carry +15.6% two-qubit gates and −28.4% ESP, and SABRE's, from the identical layout, +12.6% and −28.7%. The difference is produced by the preset's gate-level passes (commutation analysis, inverse cancellation across commuting gates, two-qubit block consolidation and resynthesis), which neither router applies; every routing method in Table VI receives only the level-1 finalisation. Under the noise-aware layout all routers, SABRE included, reach a mean ESP of about 0.128 on the random family compared with the preset's 0.147, which quantifies the headroom that gate-level optimisation leaves after routing. The same table also shows that the choice of layout moves the mean ESP of every router from about 0.092 to 0.138, a larger effect than any routing decision measured here; this is consistent with the findings of [16] and is the reason the comparison was run under both regimes.

The two ablations isolate the contributions of the reward design. The calibration-blind agent, which sees the device-mean error weight on every edge, routes with a median gate count of +0.0% and ESP −0.5% relative to SABRE under the balanced preference, statistically indistinguishable from the full agent. This is a negative result for the simplest reading of noise-aware routing: at $\mathbf{w} = (1/3, 1/3, 1/3)$ the error term is too weak, relative to the gate-count term, to change which edge the agent swaps on, so the calibration features do not pay off unless the preference is moved towards the error objective (Section VIII-E). The gate-count-only agent ($\mathbf{w}$ fixed to (1, 0, 0) during training and inference) reaches +1.3% gates and −5.4% ESP, which confirms that routing cost is dominated by SWAP count and that a scalar reward is sufficient to reach SABRE parity; what the multi-objective agent adds is the ability to move away from that point on request, not a better point itself.

E. Preference Sweep

Fig. 5 traces the sweep over ten preference vectors. Relative to SABRE, the pure gate-count preference gives a mean gate ratio of 1.014 and a mean gate-error-weight ratio of 1.011, whereas the pure error preference gives 1.032 and 0.996; the balanced preference sits at 1.017 and 0.996 and the depth preference at a depth ratio of 1.024 against 1.026 for the gate preference. Moving the preference therefore changes the output (66% of configurations route differently under $\mathbf{w} = (1, 0, 0)$ and (0, 0, 1)), and it changes it in the intended direction on average, but the achievable trade-off range is narrow: between the two extremes the mean Λ_{gate} change is −1.1% (lower in 41% of configurations, higher in 24%) for a mean gate-count change of +1.9%, and the median change of either quantity is zero because most configurations move by a few

gates or not at all. Greedy-MO evaluated with the same preferences lies above and to the right of the agent's points throughout, so the learned policy dominates the one-step rule at every preference. Qiskit O3 lies outside the reachable region of both (0.879 gate ratio, 0.860 error-weight ratio) for the reason given above. The sweep also exposed the agent's only failure mode: under the extreme preference (0, 0, 1), where a SWAP on a low-error edge is almost free, the greedy decoder occasionally circles without progress until the SWAP budget is exhausted; this happened in 11 of 1080 sweep routings under (0, 0, 1) and in 2 routings under all other preferences combined. A production compiler would fall back to SABRE in that case, as the best-of-k decoder already does implicitly.

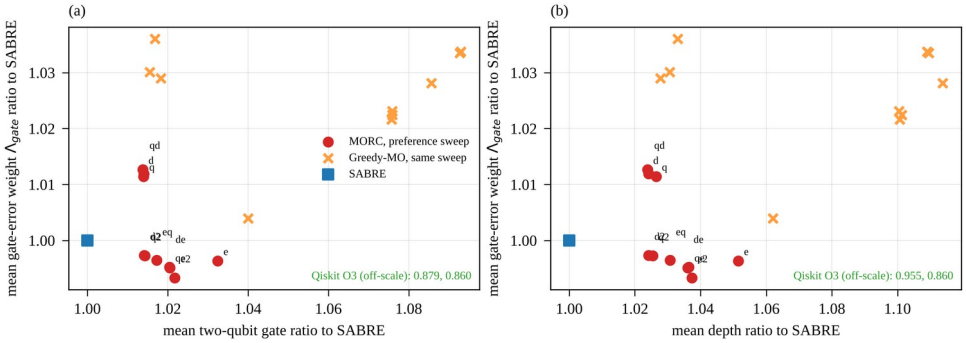


Fig. 5. Preference sweep of the trained agents (ten preference vectors, labelled by their dominant weights) under the noise-aware layout, averaged over all configurations as ratios to SABRE: (a) gate-error weight Λ_{gate} against two-qubit gate count, (b) against depth. Greedy-MO is evaluated with the same preferences; SABRE is the reference point; Qiskit O3 lies off-scale.

F. Validation of the Error Model by Density-Matrix Simulation

Fig. 6(a) compares the ESP of Eq. (9) with the Hellinger fidelity obtained by density-matrix simulation for 1869 compiled circuits (3–6 logical qubits, up to nine active physical qubits, all methods and both layouts, three calibration seeds). The comparison that matters for the reward is between compiled variants of the same circuit, and there the two measures agree closely: over the 14 circuits whose compilations have ESP above 10^{-3}, the within-circuit Spearman correlation between ESP and simulated fidelity has median $\rho = 0.91$ (range 0.83–0.96). Pooled across circuits the correlation is much weaker ($\rho = 0.368$ over 1600 compilations) because the absolute fidelity level depends on the shape of the ideal distribution: a flat random-circuit output retains a high Hellinger fidelity under partial depolarisation whereas a peaked QFT or Grover output does not, an effect that a product-of-survival-probabilities model cannot represent. The remaining 269 circuits, all Grover-5 and Grover-6 compilations with several hundred two-qubit gates, have ESP below 10^{-3} and a simulated fidelity pinned at the random-guess floor, where neither measure carries information (Spearman over all 1869 circuits: $\rho = 0.578$). Within a fixed circuit, topology, calibration and layout the ESP orders any two compiled variants in the same way as the simulated fidelity in 87.3% of 3524 pairs. The absolute level is therefore not predicted, and the decoherence term of Eq. (9), which charges the full circuit duration to every qubit, adds a further circuit-dependent offset; the reward relies on ESP only as a ranking signal within a circuit, which is what the data support. Mean simulated fidelities under the noise-aware layout are 0.662 for the agent and 0.660 for SABRE, against 0.574 and 0.574 under the SABRE layout and 0.683 for the O3 preset, reproducing in simulation the ordering of the ESP columns of Table V.

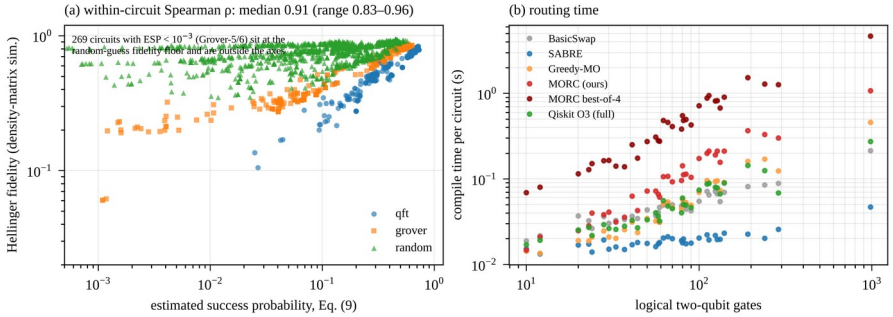


Fig. 6. (a) Hellinger fidelity from noisy density-matrix simulation against the ESP of Eq. (9) for every compiled circuit with at most nine active physical qubits (all methods, both layouts, three calibration seeds); (b) routing time per circuit against logical two-qubit gate count, noise-aware layout, mean over calibration seeds.

G. Compile Time

Fig. 6(b) and the last column of Table VII give compile times. The greedy decoder averages 0.135 s per circuit against 0.019 s for SABRE and 0.056 s for the full O3 preset, i.e. 7× SABRE; best-of-4 costs 0.574 s. The largest circuit, Grover-6 with 984 logical CX gates, takes 1.07 s with the greedy decoder and 0.05 s with SABRE. The overhead is almost entirely the Python-level computation of the distance-reduction features for every admissible edge at every step; the network forward pass is a small fraction of it. The scaling is linear in the number of inserted SWAPs, as predicted in Section VI-A, and no circuit required more than a few seconds.

H. Mid-Build Correction and Negative Findings

Two findings that did not confirm the initial design are reported here rather than omitted. First, the agent as first trained, with a narrower calibration spread (log-normal geometric standard deviation 1.7 and no outlier edges) and without the immediate-cost feature of Table II, learned a policy that was almost independent of the preference vector: on a probe set of 12 circuits the output changed under different preferences in 25% of cases and the error weight did not fall when the error preference was raised. The calibration model was changed to the heavy-tailed form of Section VII-B and the immediate scalarised cost of a SWAP on each edge was added as a feature, after which the preference changed the output in 92% of a 24-circuit probe set; all results above are from the retrained agents, and the superseded checkpoints are kept in the reproducibility bundle. Second, the authors' expectation that noise-aware routing would raise ESP at the balanced preference was not borne out: the calibration-blind ablation matches the full agent, and the ESP gain of the full agent over SABRE is confined to the best-of-k decoder and to the error-weighted end of the preference sweep. The practical implication is that calibration awareness in routing is a second-order lever behind layout selection and gate-level optimisation on devices with the error statistics assumed here.

IX. Comparative Analysis

Table VII places the proposed method alongside the approaches of Table I on the dimensions that matter for deployment inside a compiler. The quantitative comparison is restricted to methods that were re-implemented or are available in Qiskit, because published SWAP counts are not comparable across benchmark definitions, basis sets and device graphs. On gate count the proposed method reaches parity with SABRE, the strongest heuristic router available in Qiskit, whereas the learned routers of [7] and [8] reported SWAP-count improvements over the SABRE implementations of their time on different benchmark sets; the comparison in this paper uses SABRE's current decay heuristic with 20 trials and a shared noise-aware layout, which is a harder reference. On the error objective the method matches SABRE at the balanced preference and improves on it with sampling or an error-weighted preference, which no SWAP-count router can do, but the gain is smaller than the gains reported for noise-adaptive layout selection in [4] and [16], which this method inherits by starting from the same layout rather than competes with.

TABLE VII. Qualitative comparison with existing routing and compilation approaches

Approach	Objective(s)	Calibration-aware	User preference	Graph-agnostic	Time / circuit
SABRE [3]	SWAP count	no	no	yes	ms
Noise-adaptive SMT [4]	success prob.	yes	fixed	yes	s–min
Error-aware beam search [6]	ESP	yes	fixed	yes	s
DQN routing [7]	SWAP count	no	no	per device	s
GNN + MCTS [8]	SWAP count	no	no	yes	s–min
RL pass ordering [9]	composite figure of merit	via Target	fixed	per device	s
VF2 post-layout [16]	layout error score	yes	no	yes	ms
MORC (this work)	2q gates + depth + error, Eq. (3)	yes (state)	yes, at inference	yes (edge-shared)	ms–s

Two further distinctions are worth stating. Cost: the per-circuit compile time of MORC is 7× that of SABRE in this implementation (Fig. 6b), which is dominated by Python-level feature computation rather than by the network; a compiled implementation would close most of the gap, and the training cost is amortised over every subsequent circuit, device and calibration. Reliability: the agents completed every configuration of the main evaluation except 2 of 3600 routings and failed in the sweep almost only under the extreme error-only preference, and every small-circuit output was verified against the logical unitary.

X. Key Outcomes

- *Novel contribution.* Routing is formulated as a preference-conditioned, potential-based MDP whose return equals the negative change of an explicit three-objective cost (Proposition 1), and a single edge-equivariant policy serves graphs of 15–28 edges without retraining.
- *Engineering result.* Across 600 configurations the greedy-decoded agent routes within 1.5% of SABRE's gate count and matches its ESP, improves on SABRE on structured QFT circuits, and with four sampled decodes achieves −1.2% depth and +0.3% ESP at equal gate count, all with one checkpoint on three processor graphs.
- *Validated error model.* The ESP used in the reward orders compiled circuits in the same way as a density-matrix simulation with depolarising, relaxation and readout noise (Spearman $\rho = 0.91$; pairwise ordering agreement 87.3% over 3524 pairs).
- *Practical significance.* The preference vector gives compiler users a single knob for the depth/gate/error trade-off that is otherwise hidden inside heuristic constants; the complete pipeline runs on a two-core CPU and every result is reproducible from fixed seeds.
- *Honest limits.* The agent does not beat SABRE on gate count when decoded greedily, its calibration awareness yields no ESP gain at the balanced preference, the reachable trade-off range is narrow, the full Qiskit pipeline remains far ahead because of gate-level optimisation that routing cannot supply, and all calibrations are synthetic.

XI. Conclusion

This paper asked whether qubit routing can be learned as a controllable multi-objective policy rather than tuned as a heuristic. The answer, from 600 paired configurations on three processor graphs, is a qualified yes. A single PPO agent with 58,562 parameters, trained once on random circuits with randomised graphs, calibrations and preferences, routes unseen QFT, Grover and random circuits with a gate count within 1.5% of SABRE and an ESP that is statistically indistinguishable from it, outperforms the one-step heuristic built on its own features by 4.6% of gates, and, when four decodes are sampled, delivers −1.2% depth and +0.3% ESP at SABRE's gate count. The preference vector changes the output of most configurations in the intended direction, and the error model that drives the reward was shown to rank compiled circuits in agreement with a density-matrix simulation in 87.3% of pairwise comparisons.

The contributions that survive this evaluation are the potential-based multi-objective MDP with its exact return identity, the edge-equivariant preference-conditioned policy that transfers across graphs without retraining, and a controlled protocol, with shared layouts, shared finalisation, paired non-parametric statistics and independent simulation of the error model, that makes the comparison with heuristic routers meaningful. What the evaluation does not support is the stronger claim, implicit in much of the learned-routing literature, that reinforcement learning routes substantially better than a well-configured SABRE on general circuits; in this study it routes about as well, and its advantage is control rather than raw gate count.

A. Limitations

The calibration snapshots are synthetic; their marginal statistics follow public calibration tables, but spatial correlations, crosstalk and drift within a job are absent, and no hardware execution was performed. The benchmark suite has 40 circuits of at most 16 qubits and the processors at most 27, which is the regime in which density-matrix validation and exhaustive paired evaluation remain affordable; behaviour on 100-qubit lattices is untested. The agent sees only single-SWAP actions, so it cannot discover bridge gates or exploit commutation, which is where the O3 preset obtains its advantage. The SABRE baseline was run with its decay heuristic and 20 trials but without the layout-and-routing iteration of the full SabreLayout pass, which was held fixed to isolate routing. The MATLAB scripts in the bundle were not executed. Finally, author designations and reference metadata in this manuscript are placeholders pending verification before submission.

B. Future Scope

Four extensions follow directly from the limitations. First, replacing the synthetic calibration sampler with archived calibration tables from public devices, and ultimately executing the compiled circuits on hardware, would test whether the ESP ranking validated here in simulation survives correlated and time-varying noise. Second, the per-step feature computation should move to a compiled kernel and the policy should be given a small number of sampled rollouts by default, since the best-of-4 variant already shows that cheap diversity pays. Third, the action space can be widened beyond single SWAPs to bridge gates and to the joint selection of layout and routing moves, which is where the largest error reductions in Table VI originated. Fourth, the preference-conditioned policy is a natural host for a learned scalarisation that targets a Chebyshev rather than a weighted-sum criterion, which would reach the non-convex parts of the Pareto front that the present sweep cannot.

References

[1] J. Preskill, “Quantum computing in the NISQ era and beyond,” *Quantum*, vol. 2, p. 79, Aug. 2018.

[2] A. Javadi-Abhari *et al.*, “Quantum computing with Qiskit,” arXiv:2405.08810, 2024.

[3] G. Li, Y. Ding, and J. Xie, “Tackling the qubit mapping problem for NISQ-era quantum devices,” in *Proc. 24th Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, Providence, RI, USA, 2019, pp. 1001–1014.

[4] P. Murali, J. M. Baker, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, “Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers,” in *Proc. ASPLOS*, 2019, pp. 1015–1029.

[5] S. S. Tannu and M. K. Qureshi, “Not all qubits are created equal: A case for variability-aware policies for NISQ-era quantum computers,” in *Proc. ASPLOS*, 2019, pp. 987–999.

[6] S. Nishio, Y. Pan, T. Satoh, H. Amano, and R. Van Meter, “Extracting success from IBM’s 20-qubit machines using error-aware compilation,” *ACM J. Emerg. Technol. Comput. Syst.*, vol. 16, no. 3, art. 32, 2020.

[7] M. G. Pozzi, S. J. Herbert, A. Sengupta, and R. D. Mullins, “Using reinforcement learning to perform qubit routing in quantum compilers,” *ACM Trans. Quantum Comput.*, vol. 3, no. 2, art. 10, 2022.

[8] A. Sinha, U. Azad, and H. Singh, “Qubit routing using graph neural network aided Monte Carlo tree search,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 9, 2022, pp. 9935–9943.

[9] N. Quetschlich, L. Burgholzer, and R. Wille, “Compiler optimization for quantum computing using reinforcement learning,” in *Proc. 60th ACM/IEEE Design Autom. Conf. (DAC)*, 2023, pp. 1–6.

[10] T. Fösel, M. Y. Niu, F. Marquardt, and L. Li, “Quantum circuit optimization with deep reinforcement learning,” arXiv:2103.07585, 2021.

[11] Z. Li, J. Peng, Y. Mei, S. Lin, Y. Wu, O. Padon, and Z. Jia, “Quarl: A learning-based quantum circuit optimizer,” *Proc. ACM Program. Lang.*, vol. 8, no. OOPSLA1, art. 128, 2024.

[12] D. Kremer, V. Villar, H. Paik, I. Duran, I. Faro, and J. Cruz-Benito, “Practical and efficient quantum circuit synthesis and transpiling with reinforcement learning,” arXiv:2405.13196, 2024.

[13] A. Zulehner, A. Paler, and R. Wille, “An efficient methodology for mapping quantum circuits to the IBM QX architectures,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 7, pp. 1226–1236, Jul. 2019.

[14] S. Niu, A. Suau, G. Staffelbach, and A. Todri-Sanial, “A hardware-aware heuristic for the qubit mapping problem in the NISQ era,” *IEEE Trans. Quantum Eng.*, vol. 1, art. 3101614, 2020.

[15] N. Quetschlich, L. Burgholzer, and R. Wille, “MQT Bench: Benchmarking software and design automation tools for quantum computing,” *Quantum*, vol. 7, p. 1062, 2023.

[16] P. D. Nation and M. Treinish, “Suppressing quantum circuit errors due to system variability,” *PRX Quantum*, vol. 4, no. 1, art. 010327, 2023.

[17] L. Lao, H. van Someren, I. Ashraf, and C. G. Almudever, “Timing and resource-aware mapping of quantum circuits to superconducting processors,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 2, pp. 359–371, Feb. 2022.

[18] A. Cowtan, S. Dilkes, R. Duncan, A. Krajenbrink, W. Simmons, and S. Sivarajah, “On the qubit routing problem,” in *Proc. 14th Conf. Theory*

Quantum Comput., Commun. Cryptogr. (TQC), LIPIcs vol. 135, 2019, pp. 5:1–5:32.

[19] A. Abels, D. Roijers, T. Lenaerts, A. Nowé, and D. Steckelmacher, “Dynamic weights in multi-objective deep reinforcement learning,” in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 11–20.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” arXiv:1707.06347, 2017.

[21] K. Miettinen, *Nonlinear Multiobjective Optimization*. Boston, MA, USA: Kluwer, 1999.

[22] A. Y. Ng, D. Harada, and S. Russell, “Policy invariance under reward transformations: Theory and application to reward shaping,” in *Proc. 16th Int. Conf. Mach. Learn. (ICML)*, 1999, pp. 278–287.

[23] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proc. 28th Annu. ACM Symp. Theory Comput. (STOC)*, 1996, pp. 212–219.

[24] B. Tan and J. Cong, “Optimal layout synthesis for quantum computing,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2020, pp. 1–9.

[25] Qiskit contributors, “Qiskit Aer: High-performance simulators for quantum circuits,” version 0.18, 2026. [Online]. Available: https://github.com/Qiskit/qiskit-aer