Reward Engineering for RL-Based Initial Qubit Mapping

From Reward–Metric Alignment to Compiled-Circuit Quality

Federico Lentini

Supervisor(s): Sebastian Feld[1], Akash Kundu[1], Matthijs Spaan[1]

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

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

Name of the student: Federico Lentini
Final project course: CSE3000 Research Project
Thesis committee: Matthijs Spaan, Sebastian Feld, Anna Lukina

Abstract

Quantum compilers are necessary to adapt quantum circuits to the connectivity and noise constraints of Noisy Intermediate-Scale Quantum devices. A key step in the compilation process is initial qubit mapping, where logical qubits are assigned to physical qubits. This assignment affects the overall reliability of the circuit, as mappings that place interacting qubits on non-adjacent physical qubits can require additional SWAP gates during routing. Recently, reinforcement learning techniques have been employed to tackle the initial qubit mapping problem. In this setting, the choice of reward function is particularly important, since it defines the objective the RL agent learns to optimize. This raises a central question for RL-based initial mapping: whether graph-level rewards can serve as useful proxies for downstream compiled-circuit quality. This work addresses this question by engineering reward functions based on hardware-distance and hardware-fidelity signals, including a hybrid reward that combines both, and evaluating how well they predict two downstream metrics: compiled SWAP count and estimated success probability (ESP). The same rewards are used to train action-masked PPO agents with terminal, shaped and (n)-step shaped variants on 5-qubit hardware topologies. Overall, the results show that graph-level rewards provide useful learning signals to guide RL-based initial mapping, but remain limited as proxies for full-compilation performance. This highlights the need for reward design that includes more routing-relevant information, such as repeated interactions and gate ordering, to better approximate the effects of routing on final compiled-circuit quality.

1 Introduction

As quantum technology continues to advance, quantum computation is emerging as a promising platform to solve problems that are infeasible for classical computers [1, 2, 3]. In particular, quantum computation has shown its potential in areas such as quantum simulation [4], optimization [5], and cryptography [6]. Current quantum processors belong to the Noisy Intermediate-Scale Quantum (NISQ) era [7]: devices with hundreds of qubits are available and can execute non-trivial quantum circuits.

However, current NISQ devices are still far from ideal hardware for executing quantum algorithms: two-qubit operations can be executed directly only between certain pairs of physical qubits, their fidelity can vary across the device, and qubit states can only be preserved for a limited time [7]. In contrast, quantum algorithms are typically designed in a hardware-agnostic way [8, 9], assuming that any gate operation can be executed perfectly and applied between arbitrary pairs of qubits.

These hardware limitations create a gap between quantum algorithms and their execution on real devices. A quantum compiler [10, 11] is therefore needed to translate a quantum algorithm to an executable circuit on a real device. The compilation process is usually divided into three stages: initial mapping [12], routing [13], and scheduling [14].

The initial mapping problem consists of assigning each logical qubit in the quantum circuit to a physical qubit on the target device [15]. The assignment must respect the device's coupling graph, which defines the hardware connectivity: a two-qubit gate can be executed directly only if the corresponding physical qubits in the coupling graph are connected by an edge. If two interacting logical qubits are mapped to non-adjacent physical qubits, the compiler must insert SWAP gates so that a gate operation can be performed between them.

However, SWAP insertion is very costly: quantum operations are inherently error-prone [16], and SWAP gates add extra two-qubit operations (3 CNOT gates) to the circuit, therefore increasing its accumulated error. Additional SWAP gates can also significantly increase the depth and duration of the compiled circuit, making the computation more exposed to decoherence [13].

Finding an initial mapping that minimizes the number of SWAP gates added during routing is therefore crucial to improving the efficiency and reliability of compiled circuits on NISQ devices [17]. In addition, initial mappings should also favor interactions that use higher-fidelity hardware connections to improve the overall reliability of the circuit.

Despite its importance, there is no consensus on an optimal strategy for solving the initial mapping problem across different circuits, hardware topologies, and optimization objectives. The problem is combinatorial in nature and can be related to the subgraph isomorphism problem, which is NP-complete [15]. In practice, heuristic-based approaches such as SABRE [18] have been used to address mapping and routing. These methods are effective, but they rely on handcrafted cost functions and may not generalize equally well across different circuit families, hardware topologies, and optimization objectives.

In this work, we investigate Reinforcement Learning (RL) [19] for the initial mapping pass. In RL-based initial mapping, the reward function defines the objective that the agent learns to optimize; however, this reward is typically computed before full compilation and therefore acts only as a proxy for downstream compiled-circuit quality. This raises a central question: whether such rewards are aligned with downstream compiled-circuit metrics, and how agents trained with these rewards perform against established baselines with respect to the objectives they are meant to optimize. In this work, we evaluate this question for two downstream objectives: reducing compiled SWAP count, as a measure of routing overhead, and improving the estimated success probability (ESP) of the compiled circuit.

Existing RL approaches to qubit mapping typically focus on the learning architecture, routing strategy, or a specific optimization objective, while the choice of reward formulation itself is rarely studied systematically [20, 21, 22, 23]. We address this gap by engineering and evaluating graph-level reward functions computed before compilation and designed to target routing overhead, circuit reliability, and their trade-off.

The main contributions of this work can be summarized as follows:

- We formulate graph-level reward design as a reward–metric alignment problem and systematically evaluate how different pre-compilation reward signals relate to downstream compiled-circuit quality, specifically routing overhead and estimated circuit reliability. In doing so, we aim to provide an initial step toward a broader framework for designing, comparing, and analyzing reward functions for RL-based initial qubit mapping.
- We study a distance-based reward and a fidelity-aware reward, inspired by [21] and [20], respectively. We fur-

ther propose four additional graph-level rewards: a hybrid distance–fidelity reward, a direct adjacency reward, and two interaction-centrality variants based on degree and closeness. We also study the effect of reward timing, by comparing terminal, shaped, and (n)-step shaped variants.

- We evaluate the proposed rewards at both the proxy and learning level. Reward quality is assessed through reward–metric correlation analysis, while trained PPO agents are benchmarked against random mapping and SABRE, across multiple 5-qubit topologies.

The remainder of this paper is organized into the following sections. Section 2 introduces the theoretical background necessary for this work. Section 3 provides an overview of previous attempts to solve the initial mapping problem. Section 4 describes the experimental setup, including the RL environment, reward formulations, and evaluation metrics. Section 5 presents the results on the considered 5-qubit hardware topologies. Finally, Section 6 discusses the implications of the findings while Section 7 addresses ethical considerations, reproducibility, and limitations. Section 8 summarizes the main conclusions and outlines directions for future research.

2 Background

This section introduces the notation and concepts used throughout the paper. It first reviews the basic quantum-computing concepts needed for this work, together with the hardware constraints of Noisy Intermediate-Scale Quantum (NISQ) devices. It then formalizes the initial mapping problem and introduces reinforcement learning as the framework used in this work.

2.1 Near-term quantum computation

We begin with an overview of fundamental quantum-computing concepts. The discussion is intentionally brief; a more complete treatment of quantum computation can be found in Ref. [1]. We then discuss the hardware limitations of NISQ devices that motivate the initial mapping problem.

Qubit. In classical computation, the fundamental unit of information is the bit, which can take one of two values, 0 or 1. In quantum computation, the corresponding unit is the qubit, which has two computational basis states, denoted by $|0\rangle$ and $|1\rangle$, but can also be in a superposition of them:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,,$$

where $\alpha, \beta \in \mathbb{C}$ are probability amplitudes satisfying $|\alpha|^2 + |\beta|^2 = 1$.

Quantum gate. A quantum gate is an operation applied to one or more qubits to transform their quantum state. Single-qubit gates act on one qubit, while two-qubit gates act on pairs of qubits. Two-qubit gates are especially relevant in this work because their execution is constrained by the hardware connectivity of the device. A common two-qubit gate is the controlled-NOT (CNOT), which flips a target qubit depending on the state of a control qubit. Another key example is the SWAP gate, typically decomposed into three CNOT gates, which exchanges the quantum states of two qubits.

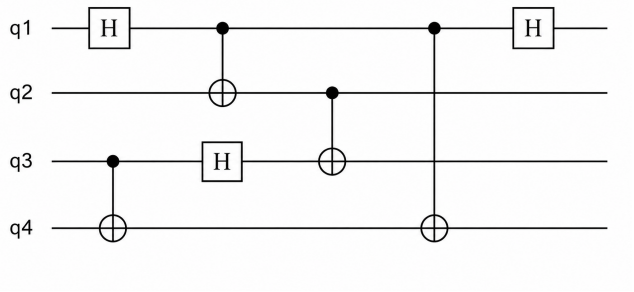


Figure 1: Example 4-qubit quantum circuit. Hadamard gates are shown as H boxes. CNOT gates are represented by a filled control dot connected to a target $\oplus$.

Quantum circuit. A quantum circuit represents a quantum algorithm as an ordered sequence of quantum gates applied to qubits, followed by an optional measurement. Figure 1 represents a 4-qubit circuit with Hadamard and CNOT gates.

NISQ devices. A quantum device is the physical hardware on which a quantum circuit is executed. Current devices operate in the NISQ regime, effectively imposing several hardware constraints. Similarly to [18], we consider the following limitations of NISQ devices:

1. **Restricted qubit connectivity.** NISQ devices are typically not all-to-all connected: two-qubit gates can only be executed directly between certain pairs of physical qubits.
2. **Limited coherence time.** A quantum state can only be preserved for a finite time before it is degraded by interactions with the environment. Longer circuits are therefore more exposed to decoherence-related errors.
3. **Non-uniform two-qubit gate fidelities.** Two-qubit gate operations are known to be error-prone. Additionally, the fidelities of these operations can vary across the device such that some physical connections are more reliable than others.

Restricted connectivity can be directly addressed during the qubit routing pass by inserting SWAP gates until the required two-qubit operation becomes executable. However, each additional SWAP introduces extra noise, increasing the accumulated error of the compiled circuit. SWAP insertion may also increase the depth and duration of the circuit, exposing the computation more strongly to decoherence. Non-uniform gate fidelities also imply that using more reliable physical connections is preferable.

Together, these limitations motivate reward objectives that account for both hardware distance and hardware fidelity, as well as evaluation metrics that reflect routing overhead and circuit reliability. Below, we formalize the initial mapping problem as an optimization problem over physical qubit assignments.

2.2 Formalization of the Initial Mapping Problem

The initial mapping problem consists of assigning the logical qubits of a quantum circuit to the physical qubits of a target device before routing is performed. We represent the target hardware as a *coupling graph* $G_H = (Q_P, E_H)$, where the vertices Q_P are physical qubits and the edges E_H are pairs

of physical qubits on which a two-qubit gate can be directly applied. Similarly, the circuit is represented by an *interaction graph* $G_I = (Q_L, E_I)$, where the vertices Q_L are logical qubits and an edge $(q_i, q_j) \in E_I$ indicates that logical qubits q_i and q_j interact through at least one two-qubit operation in the circuit.

An initial mapping is a function $\pi : Q_L \to Q_P$, which assigns each logical qubit to a physical qubit. We assume the number of logical and physical qubits is the same, so that π is a bijection.

Given a mapping π, the circuit interactions are embedded onto the hardware as

$$E_M(\pi) = \{(\pi(q_i), \pi(q_j)) : (q_i, q_j) \in E_I\}.$$

If all mapped interactions are contained in the hardware edge set, $E_M(\pi) \subseteq E_H$, then all two-qubit interactions in the circuit can be executed directly. On sparse hardware topologies this condition is generally not satisfied, and routing operations are required to make non-adjacent interactions executable.

The goal of initial mapping is therefore to find a mapping that minimizes a hardware-dependent cost over the mapped interactions. A general proxy formulation is

$$\pi^* = \arg\min_{\pi} \sum_{(q_i,q_j)\in E_I} C_H\big(\pi(q_i), \pi(q_j)\big).$$

where C_H assigns a hardware-dependent cost to placing interacting logical qubits q_i and q_j on physical qubits $\pi(q_i)$ and $\pi(q_j)$. In this work, the proposed reward functions can be interpreted as different choices of the hardware-dependent cost C_H.

2.3 Reinforcement Learning

Reinforcement Learning (RL) is a machine-learning framework in which an agent interacts with the environment by performing actions and receiving feedback in the form of state changes and rewards [19]. At each time step, the agent observes a state, selects an action, and receives a reward that provides feedback on the quality of that action. RL problems are commonly formalized as Markov Decision Processes (MDPs) [24], defined by a tuple $(\mathcal{S}, \mathcal{A}, T, r, \mu_0, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, and

$$T(s' \mid s, a) = \Pr(S_{t+1} = s' \mid S_t = s, A_t = a)$$

defines the transition probability of moving to state $s' \in \mathcal{S}$ after taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. μ_0 denotes the initial state distribution and $\gamma \in [0, 1]$ is the discount factor. The reward function is written as $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, or, more generally, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ if the reward depends on the resulting next state.

In the context of quantum compilation, RL is attractive because compilation tasks can often be framed as sequential decision-making problems. In this work, the agent constructs an initial mapping by assigning logical qubits to physical qubits based on a learned policy. The learned policy should select mappings that maximize the expected reward, where the reward is designed as a proxy for downstream compilation objectives such as reducing SWAP overhead or improving circuit reliability.

3 Related Work

Siraichi et al. [15] formalized the qubit allocation problem and proposed both an exact algorithm, which is exponential in runtime, and a heuristic solution. Li et al. [18] introduced SABRE, a SWAP-based bidirectional heuristic search algorithm that improves initial mapping quality using reverse traversal, and performs routing by selecting SWAP operations according to a hand-designed cost function. These heuristic methods provide strong practical baselines, but rely on manually designed objectives.

Reinforcement learning has been explored as an alternative to manually designed mapping and routing heuristics. Huang et al. [22] proposed an RL-based model for the initial mapping problem and combined it with the DEAR routing framework. Other works study mapping under specific hardware assumptions; for example, Liu et al. [23] applied deep reinforcement learning to map quantum circuits to two-dimensional nearest-neighbor architectures. These works demonstrate the potential of RL for qubit mapping, but they mainly study specific architectures or hardware assumptions rather than systematically comparing reward formulations.

Recent work has also considered reinforcement learning for qubit mapping as a combinatorial optimization problem. CO-MAP, proposed by Kulshrestha and Liu [21], formulates qubit allocation as a combinatorial optimization objective and trains an encoder–decoder policy network to produce logical-to-physical assignments. Its terminal reward is based on a SWAP-cost proxy derived from hardware distances, and the method is further improved using local-search post-processing. Oancea et al. [20] studied deep reinforcement learning for initial qubit mapping under fixed gate error rates, incorporating hardware fidelities directly into the reward function to optimize mappings for circuit reliability.

Overall, these works demonstrate that both heuristic and reinforcement-learning-based methods can improve qubit mapping and routing. However, most RL-based approaches focus on the policy architecture, the routing strategy, or performance under a single optimization objective. The reward formulation itself is usually treated as a design choice rather than as the main object of study: it remains unclear whether graph-level rewards computed before compilation can serve as useful proxies for downstream compiled-circuit metrics. This work addresses this gap by comparing multiple pre-compilation reward formulations, evaluating both their reward–metric alignment and the performance of PPO agents trained with them.

4 Methodology

This section presents the experimental methodology used to evaluate the reward formulations studied in this work. We first describe the RL environment and graph-level rewards used to train the agent. We then introduce the evaluation pipeline, including circuit generation, PPO training, downstream Qiskit compilation, and baseline comparison. Finally, we define the compiled-circuit metrics used to assess mapping quality.

4.1 RL Environment

We use the *Initial Mapping* environment from QGYM [25] to formulate the initial qubit mapping problem as a

reinforcement-learning task. In this environment the circuit is reduced to an interaction graph, where nodes represent logical qubits and edges indicate two-qubit interactions between them. This representation ignores single-qubit gates, gate order, and interaction frequency, as all edges are unweighted: it represents a simplified abstraction of the original circuit that retains only the connectivity information relevant to the initial mapping problem. The hardware is represented by a coupling graph, where nodes represent physical qubits and edges represent available hardware connections. Hardware edges are additionally associated with two-qubit gate fidelities.

To train the mapping policy, we use Maskable PPO [26], a variant of Proximal Policy Optimization (PPO) [27], implemented through Stable-Baselines3 and its action-masking extension [28]. Action masking is implemented such that once a physical qubit has already been assigned, it should no longer be available as a valid action. To isolate the effect of reward formulation, all agents are trained using the same Maskable PPO configuration; the full hyperparameter setup is reported in Appendix A. For the remainder of this paper, we refer to Maskable PPO agents simply as PPO agents.

Each training episode constructs one complete initial mapping by starting from an empty mapping and sequentially assigning each logical qubit to a physical qubit. After each assignment, the environment updates the partial mapping and proceeds to the next decision step until all logical qubits have been assigned, with rewards computed either at the end of the episode for terminal rewards or during the episode for shaped and (n)-step rewards.

4.2 Reward Formulations

This section presents the graph-level reward formulations used in this work. We study a distance-based reward and a fidelity-path reward inspired by [21] and [20], respectively. Building on these, we propose additional reward formulations, including a hybrid distance–fidelity reward, a direct adjacency reward, and interaction-centrality rewards based on degree and closeness ranking. The mathematical definitions of the latter two are given in Appendix B. In addition to the reward objective itself, we also study the effect of reward-signal timing by comparing terminal, shaped, and (n)-step shaped variants, whose definitions are given in Appendix C.

The reward formulations use the initial-mapping notation introduced in Section 2.2 and are structured following the reward-design framework of Ref. [29]. We additionally use $d_H(p_u, p_v)$ to denote the shortest-path distance in the hardware coupling graph, i.e., the minimum number of hardware edges connecting physical qubits p_u and p_v. For fidelity-aware rewards, each hardware edge $(p_u, p_v) \in E_H$ is assigned a two-qubit gate fidelity F_{uv}. Each reward assigns a scalar score to a mapping π, which PPO aims to maximize. All rewards are computed before compilation from the circuit interaction graph and hardware coupling graph, and can therefore be considered as graph-level rewards.

Distance reward. Inspired by the distance-based CO objective used in CO-MAP [21], the distance reward penalizes mappings in which interacting logical qubits are far apart on the hardware graph:

$$R_{\mathrm{dist},p}(\pi) = -\sum_{(q_i,q_j)\in E_I} \left(d_H(\pi(q_i),\pi(q_j)) - 1\right)^p .$$

This reward acts as a simple SWAP-cost proxy: adjacent physical qubits have zero cost, since $d_H = 1$, while longer hardware distances are expected to require more routing operations and therefore receive larger penalties. The distance reward is therefore designed primarily to minimize the routing overhead, which is measured by the compiled SWAP count. The exponent p controls how strongly long-distance interactions are penalized; in Section 5.1, we compare $p = 1$ and $p = 2$, corresponding to linear and quadratic distance penalties, respectively.

Fidelity-path reward. Inspired by Oancea et al. [20], the fidelity-path reward incorporates hardware reliability by assigning each hardware edge $e \in E_H$ an additive log-fidelity cost $c_e = -\log(F_e)$, where $F_e \in [0, 1]$ is the two-qubit gate fidelity. Since fidelities multiply along a path, while log-costs add, minimizing the accumulated log-cost is equivalent to maximizing the path fidelity.

For two physical qubits $p, q \in Q_P$, the best path cost is defined as

$$C_{\mathrm{path}}(p, q) = \min_{P:p\to q} \sum_{e\in P} -\log(F_e),$$

where P ranges over all paths from p to q in the hardware graph, and $e \in P$ denotes the hardware edges used by that path. The quantity $C_{\mathrm{path}}(p, q)$ is therefore the minimum accumulated log-error cost between physical qubits p and q.

The fidelity-path reward is then

$$R_{\mathrm{fid}}(\pi) = -\sum_{(q_i,q_j)\in E_I} C_{\mathrm{path}}(\pi(q_i), \pi(q_j)).$$

Here, $C_{\mathrm{path}}(\pi(q_i), \pi(q_j))$ is the minimum log-error path cost between the physical qubits assigned to the interacting logical qubits q_i and q_j. This reward therefore favors mappings that place interacting logical qubits on, or near, high-fidelity hardware paths, with the objective of improving the estimated reliability of the compiled circuit.

Hybrid distance-fidelity reward. The hybrid reward combines the two graph-level signals used above: hardware distance and hardware path quality. First, the distance and fidelity-path costs are normalized as

$$\tilde{d}_{ij}(\pi) = \frac{d_H(\pi(q_i), \pi(q_j)) - 1}{d_{\max} - 1},$$

$$\tilde{e}_{ij}(\pi) = \frac{C_{\mathrm{path}}(\pi(q_i), \pi(q_j))}{C_{\max}}.$$

Here, $\tilde{d}_{ij}(\pi)$ is the normalized distance cost for interaction edge (q_i, q_j), and $\tilde{e}_{ij}(\pi)$ is the normalized fidelity-path cost for the same interaction. The value $d_{\max} = \max_{p,q\in Q_P} d_H(p, q)$ is the maximum shortest-path distance between any two physical qubits in the hardware graph, and $C_{\max} = \max_{p,q\in Q_P} C_{\mathrm{path}}(p, q)$ is the maximum fidelity-path cost between any two physical qubits. These normalizations put the distance and fidelity terms on comparable scales.

The hybrid reward is defined as

$$R_{\mathrm{hybrid},\alpha}(\pi) = -\sum_{(q_i,q_j)\in E_I} \left[\alpha\tilde{d}_{ij}(\pi) + (1-\alpha)\tilde{e}_{ij}(\pi)\right].$$

Here, $\alpha \in [0, 1]$ controls the trade-off between the distance term and the fidelity term. At $\alpha = 1$, the reward targets only routing overhead by minimizing normalized hardware distance, while at $\alpha = 0$, it targets circuit reliability by minimizing normalized fidelity-path cost. Intermediate values combine both objectives. In the experiments, $\alpha = 0.5$ is used to give equal weight to routing distance and hardware-edge reliability.

4.3 Experimental Pipeline

For each training run, the QGYM initial-mapping environment described in Section 4.1 receives two graph inputs. The first input is a circuit interaction graph, obtained by generating a random 5-qubit quantum circuit with Qiskit [30] from a fixed distribution with circuit depth 8 and two-qubit gate probability 0.6, and then converted into a binary interaction graph following the convention introduced in Section 4.1. The second input is a hardware coupling graph, chosen from five 5-qubit hardware topologies: line, ring, star, tree, and partial mesh. These topologies are used to evaluate the reward formulations across different connectivity structures, rather than tying the observed PPO performance to a single hardware graph. The graph structures of these topologies are specified in Appendix E. For each hardware graph, the two-qubit gate fidelities are sampled uniformly from [0.85, 0.99].

For each reward formulation, a separate PPO agent is trained. To account for stochasticity in RL training, each topology–reward configuration is repeated over PPO seeds 0, 1, 2, 3 and 4. Thus, each reward formulation is evaluated over $5 \times 5 = 25$ topology–seed instances. In the experiments, each agent is trained for 250,000 environment timesteps. Since each 5-qubit episode consists of five assignment steps, this corresponds to approximately 50,000 complete mapping episodes per trained agent. The fixed experimental configuration is summarized in Appendix D.

After training, each agent is evaluated on a fixed set of 100 randomly generated circuits from the same distribution. In addition to this evaluation, we test the trained agents on structured benchmark circuits from MQT Bench [31], namely `ghz`, `graphstate`, `qft`, `qaoa`, and `wstate` (see Section 5.4). These circuits are used to assess whether policies trained on randomly generated circuits generalize to more structured circuit families with different interaction patterns.

For each evaluation circuit the PPO outputs an initial logical-to-physical mapping, which is converted into a Qiskit `initial_layout`. The original Qiskit circuit is then transpiled using the PPO produced initial layout, routed with SABRE and scheduled with ALAP [32]. The resulting compiled circuit is then used to compute the downstream evaluation metrics described in Section 4.5.

4.4 Baselines

The compilation performance of the PPO mappings is compared against three baselines: random mapping, SABRE [18], and an exhaustive oracle. Random mapping samples a valid logical-to-physical bijection uniformly at random and serves as an uninformed baseline, while SABRE provides a strong heuristic baseline used in practical quantum compilation.

The exhaustive oracle is used as a metric-specific brute-force reference. Since all experiments use 5-qubit circuits and 5-qubit hardware graphs, all $5! = 120$ valid initial mappings can be compiled with the Qiskit pipeline defined in Section 4.3 and evaluated exhaustively. The oracle is then selected separately for each metric: the SWAP oracle chooses the mapping with the lowest compiled SWAP count, while the Log ESP oracle chooses the mapping with the highest Log ESP. The oracle is not intended as a scalable compilation method, but provides a reference for the best achievable initial-mapping performance under the fixed compilation pipeline in the restricted 5-qubit setting.

The exhaustive mapping sweep is also used to evaluate the reward–metric alignment of Section 5.1. For each circuit–topology pair, all $5! = 120$ valid mappings are assigned both a graph-level reward score and a compiled-circuit metric obtained after Qiskit transpilation. For each reward–metric pair, we compare the ranking induced by the reward score with the ranking induced by the compiled metric using Spearman rank correlation [33].

4.5 Evaluation Metrics

The first evaluation metric we consider is the compiled SWAP count, defined as the number of routing SWAPs inserted during compilation:

$$N_{\mathrm{SWAP}}(C_{\mathrm{comp}}) = \left|\{g \in C_{\mathrm{comp}} \ : \ g \text{ is a SWAP gate}\}\right|.$$

where C_{comp} denotes the compiled circuit obtained after the full compilation pass. Since the input circuits used in this work do not contain SWAP gates, this quantity corresponds directly to the number of SWAP gates inserted during routing. Lower compiled SWAP count indicates that the initial mapping placed interacting qubits in a way that required fewer routing operations. This metric therefore directly measures SWAP-related routing overhead.

The second main metric we consider is the estimated success probability (ESP), which estimates the reliability of the compiled circuit. Following [34], we compute the two-qubit ESP of the compiled circuit as

$$\mathrm{ESP}(C_{\mathrm{comp}}) = \prod_{g \in \mathcal{G}_{2q}(C_{\mathrm{comp}})} F_{e(g)}.$$

where $\mathcal{G}_{2q}(C_{\mathrm{comp}})$ is the set of two-qubit gates in the compiled circuit and $F_{e(g)}$ is the fidelity of the hardware edge used by gate g.

ESP depends on both the fidelities of the hardware edges used and the number of two-qubit gates in the compiled circuit. Using higher-fidelity hardware connections increases ESP, while routing overhead can decrease it because inserted SWAP gates add extra two-qubit operations to the product.

For numerical stability, we report the logarithm of the ESP, which avoids very small product values and converts the product of fidelities into a sum:

$$\log \mathrm{ESP}(C_{\mathrm{comp}}) = \sum_{g \in \mathcal{G}_{2q}(C_{\mathrm{comp}})} \log F_{e(g)}.$$

Since $F_e \leq 1$, $\log \mathrm{ESP} \leq 0$. Higher values are better, with values closer to zero corresponding to higher estimated reliability. For relative comparisons, we use the log-error cost $C_{\log}(C_{\mathrm{comp}}) = -\log \mathrm{ESP}(C_{\mathrm{comp}})$, where lower values are better.

Together, compiled SWAP count and Log ESP represent the two guiding objectives of this work: reducing routing overhead and improving estimated circuit reliability.

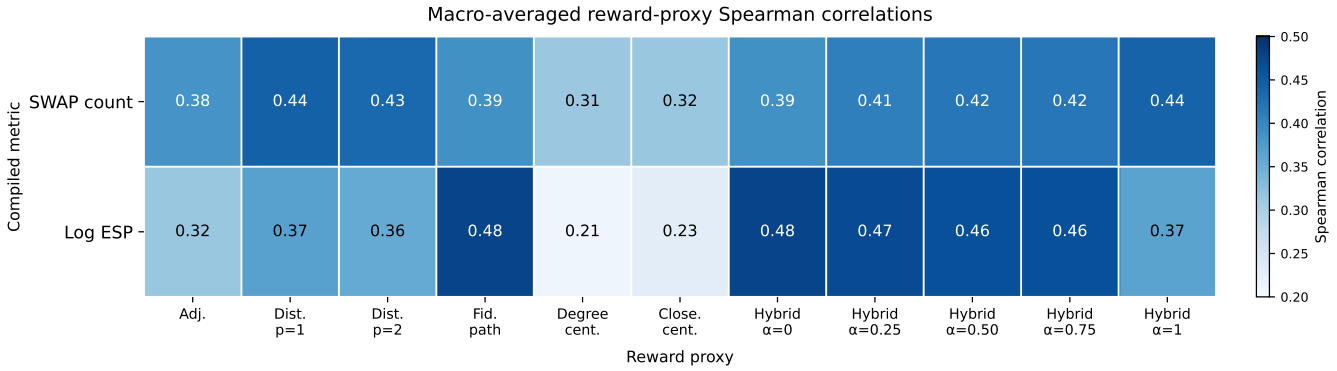
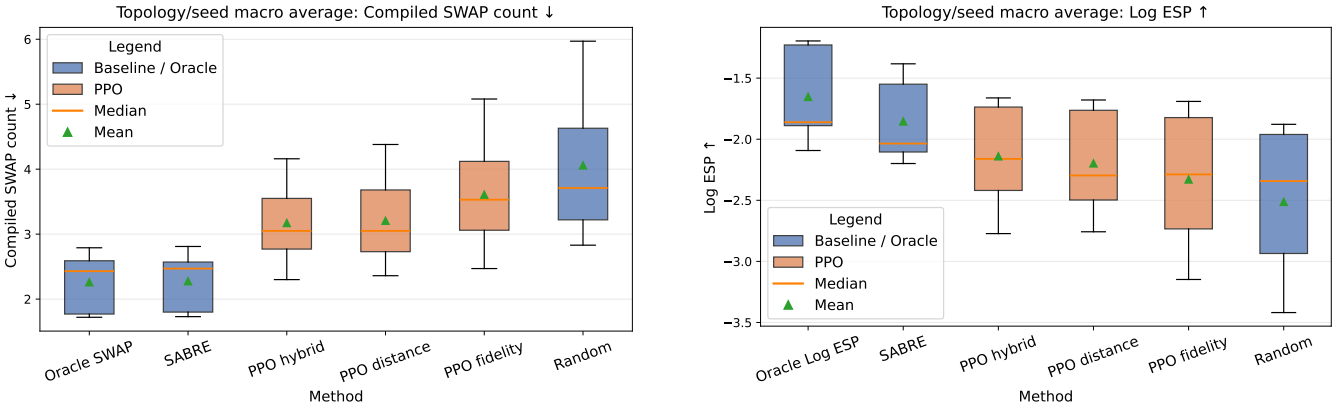


Figure 2: Macro-averaged Spearman correlations between reward scores and downstream compiled metrics over exhaustive 5-qubit mapping sweeps. Distance-based rewards align most strongly with routing-related metrics, while fidelity-path and hybrid rewards align more strongly with Log ESP.



(a) Compiled SWAP count. Lower is better.

(b) Log ESP. Higher is better.

Figure 3: Terminal reward performance across topology–seed instances. PPO improves over random mappings, but remains worse than SABRE and the oracle baselines.

5 Results

This section evaluates the rewards as graph-level proxies and PPO training signals. We first analyze reward–metric alignment, then compare terminal and shaped PPO variants against baselines, and finally test generalization on MQT Bench circuits.

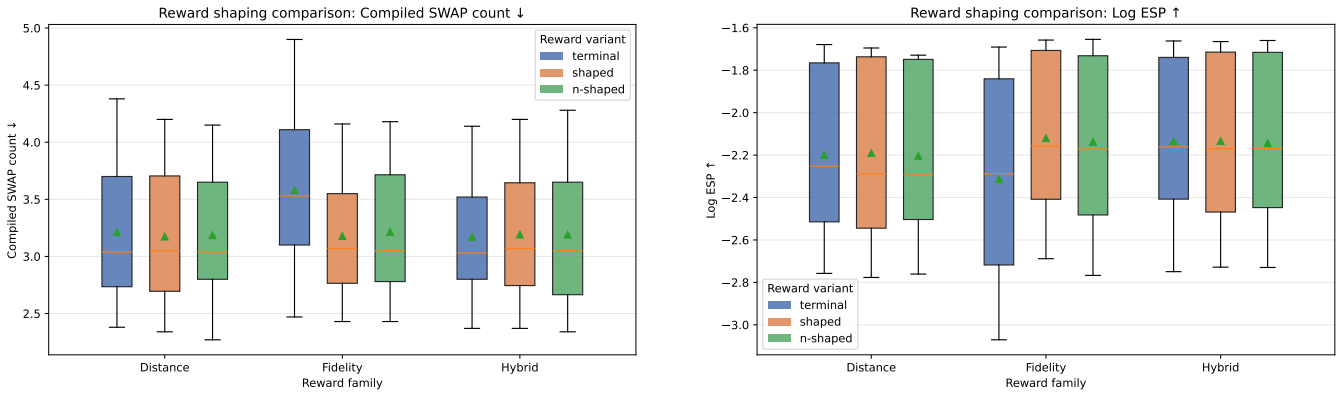
5.1 Reward–Compiled Metric Alignment

Before training PPO agents, we first analyze reward–metric alignment using the exhaustive mapping sweep described in Section 4.4. Figure 2 shows the resulting Spearman correlations, macro-averaged across hardware topologies.

The linear distance reward correlates most strongly with compiled SWAP count ($\rho = 0.44$), while the fidelity-path reward correlates most strongly with Log ESP ($\rho = 0.48$). Hybrid rewards interpolate between these objectives: increasing α improves correlation with compiled SWAP count, while decreasing α improves correlation with Log ESP. The balanced hybrid reward with $\alpha = 0.5$ provides an intermediate trade-off, showing a correlation of $\rho = 0.42$ with compiled SWAP count and $\rho = 0.46$ with Log ESP. Direct-adjacency and centrality-based rewards show weaker correlations overall, suggesting their signals are too indirect to reliably predict downstream compiled-circuit quality in this setting.

These results are consistent with the intended reward objectives described in Section 4.2: distance targets routing overhead, fidelity-path targets hardware reliability, and hybrid combines both. However, the fidelity reward alone is not expected to fully predict Log ESP: as discussed in Section 4.5, choosing high-fidelity connections may not directly improve Log ESP if they require many routing operations. The hybrid reward accounts for both effects and therefore also correlates strongly with Log ESP. Although the observed correlations are consistent with the reward design, all correlations remain moderate, with the highest value reaching only $\rho = 0.48$. This indicates that the graph-level rewards only partially predict downstream compiled-circuit quality. We discuss this further in Section 6.

This analysis characterizes the rewards as graph-level proxies, but does not yet show how useful they are as training signals for an RL agent. For the PPO experiments we focus on three representative reward functions: the linear distance reward $p = 1$, the fidelity-path reward, and the balanced hybrid reward with $\alpha = 0.5$.



(a) Compiled SWAP count. Lower is better.

(b) Log ESP. Higher is better.

Figure 4: Comparison of terminal, shaped, and n-shaped rewards for compiled SWAP count and Log ESP. Reward shaping has little effect for the distance and hybrid rewards, while the clearest improvement appears for the fidelity-based reward.

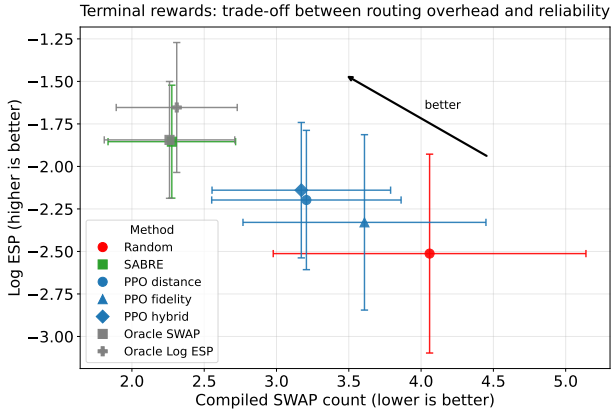


Figure 5: Trade-off comparison between compiled SWAP count and Log ESP. Lower compiled SWAP count and higher Log ESP are better, so points closer to the upper-left region correspond to better joint performance.

5.2 Terminal Reward Evaluation

We first evaluate PPO agents trained with terminal distance, fidelity-path, and hybrid rewards, following the experimental setup described in Section 4.3. Figures 3a and 3b compare PPO agents trained with terminal rewards against Random, SABRE, and the metric-specific oracle baselines.

Across both metrics, PPO improves over random mapping but remains worse than SABRE and the oracle baselines. For compiled SWAP count, the distance and hybrid rewards perform best among the PPO agents, reducing SWAP count by 19.6% and 20.3% relative to Random, respectively. The fidelity reward gives a smaller improvement of 10.4%, which is consistent with the reward–metric correlation analysis of Section 5.1.

For Log ESP, the hybrid reward gives the best mean terminal PPO performance. In terms of log-error cost, the hybrid reward improves by 14.0% relative to Random, compared with 11.6% for distance and 7.1% for fidelity. Figure 5 shows the terminal-reward results in terms of the joint trade-off between compiled SWAP count and Log ESP. Among the PPO agents, the hybrid reward lies closest to the upper-left region of the plot, while the fidelity-path reward is not the best-performing PPO method on either metric.

This is in apparent contrast with the reward–metric correlation analysis of Section 5.1, where the fidelity-path reward had the highest proxy correlation with Log ESP. This suggests that reward–metric alignment alone does not determine PPO performance: in the next section we evaluate whether changing the timing of the reward signal affects how effectively each reward can be learned.

5.3 Reward Shaping Evaluation

Figures 4a and 4b, together with Table 1, compare terminal, shaped, and (n)-step shaped variants for each reward family. The corresponding absolute mean performance and standard deviations are reported in Table 4 in Appendix F. Reward shaping has little effect on the distance and hybrid rewards, whose three variants achieve comparable performance on both metrics. This suggests that, for these reward families, the terminal signal is already sufficiently informative in the 5-qubit setting.

The clearest effect of reward shaping appears for the fidelity reward. In terms of log-error cost, terminal fidelity improves by only 7.1% relative to Random, while shaped and (n)-step shaped fidelity improve by 14.8% and 14.4%, respectively. The same trend is observed for compiled SWAP count. This indicates that the fidelity-path reward is sensitive to reward timing: although it is the most aligned graph-level proxy for Log ESP in the exhaustive analysis of Section 5.1, it is less effective when provided only as sparse terminal feedback.

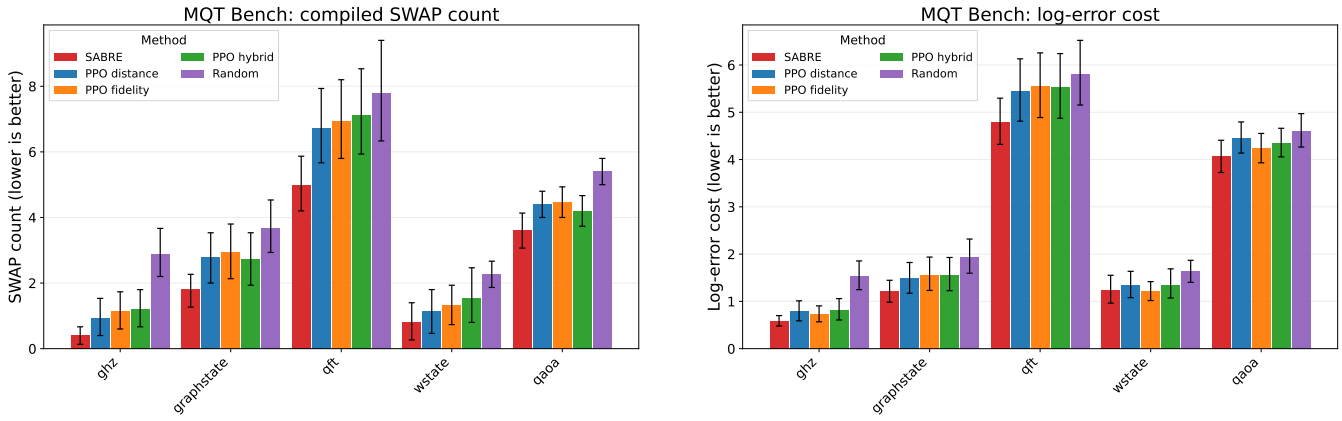
Overall, these results highlight the importance of reward timing for PPO learnability: a reward must not only align with the downstream metric, but also provide a training signal that can be effectively optimized by a sequential policy.

5.4 MQT Benchmark Evaluation

Using the MQT Bench setup described in Section 4.3, we evaluate whether the shaped PPO agents generalize beyond the random circuit distribution used during training. We focus on the shaped distance, fidelity-path, and hybrid rewards,

	Compiled SWAP count			Log-error cost		
	vs Random	vs SABRE	Gap to Oracle	vs Random	vs SABRE	Gap to Oracle
Distance terminal	+19.6%	−41.4%	42.9%	+11.6%	−18.7%	34.6%
Distance shaped	+20.3%	−40.2%	41.2%	+11.9%	−18.4%	34.0%
Distance n-shaped	+19.9%	−41.0%	41.9%	+11.4%	−19.0%	34.9%
Fidelity terminal	+10.4%	−59.4%	59.7%	+7.1%	−25.4%	40.8%
Fidelity shaped	+19.9%	−40.7%	41.9%	+14.8%	−14.6%	29.8%
Fidelity n-shaped	+19.3%	−41.9%	43.5%	+14.4%	−15.1%	30.9%
Hybrid terminal	+20.3%	−40.1%	41.4%	+14.0%	−15.7%	30.7%
Hybrid shaped	+20.1%	−40.4%	42.1%	+14.2%	−15.3%	30.7%
Hybrid n-shaped	+19.9%	−40.7%	41.8%	+13.9%	−15.7%	31.3%

Table 1: Relative performance of each PPO reward variant across topology–seed instances. The values show percentage changes in mean compiled SWAP count and mean log-error cost relative to Random, SABRE, and the metric-specific oracle. Positive values against Random or SABRE indicate improvement. The oracle gap measures the remaining percentage gap to the metric-specific oracle; lower is better.



(a) Compiled SWAP count. Lower is better.

(b) Log-error cost. Lower is better.

Figure 6: MQT Bench evaluation of SABRE and shaped PPO agents across benchmark families. Bars show the mean over hardware topologies and PPO seeds, while error bars indicate 95% bootstrapped confidence intervals for the mean. Across both metrics, SABRE performs best, Random performs worst, and the PPO variants achieve similar intermediate performance.

which provide the overall best mean PPO performance in Table 1. Figure 6 compares these agents with the SABRE and Random baselines on the MQT Bench circuits.

Across all benchmark families, SABRE achieves the lowest mean compiled SWAP count and log-error cost. The three PPO variants perform similarly to each other: no shaped reward gives a consistent advantage across benchmark families, with confidence intervals often overlapping.

6 Discussion

The reward–metric correlations of Section 5.1 are consistent with the intended reward objectives, but even the strongest correlations remain moderate. This indicates that the rewards contain useful pre-compilation information, but are incomplete predictors of compiled-circuit quality. We explain this by noting that the final compiled metrics depend on circuit information not represented in the binary interaction graph, such as gate order, repeated interactions, and gate dependencies, all of which strongly affect routing and, consequently, the final compiled metrics. This limitation is related to reward quality itself and is independent of PPO learnability: even if PPO agents were to learn perfectly, a significant gap would still remain between what the graph-level rewards can predict and the quality of the final circuit. These findings highlight the need for richer circuit representations and more routing-aware rewards, which could help reduce this gap.

Beyond reward quality alone, the terminal PPO results further show that reward–metric alignment does not always directly translate into learned-policy performance. This is particularly evident for Log ESP: the terminal fidelity reward produces the worst PPO performance among the three rewards, despite fidelity-path having the highest proxy correlation with Log ESP. This suggests that reward design has two separate requirements: the reward must align with the downstream metric, but it must also be learnable by PPO as a sequential training signal.

The reward-shaping results support this interpretation, showing that the same underlying graph-level objective can

lead to different PPO performance depending on when feedback is provided during the mapping episode. This highlights the need for a broader reward-engineering framework for RL-based quantum compilation that considers reward timing and learnability alongside reward–metric alignment. We also highlight the performance of the (n)-step shaped reward, as it achieves performance comparable to the fully shaped reward while requiring fewer reward assignments. Although this provides limited benefit in the 5-qubit setting, it may offer a useful compromise between intermediate feedback and computational cost for larger hardware topologies.

Finally, the MQT Bench evaluation shows that the learned policies do not robustly generalize beyond the random circuit distribution used during training. No shaped PPO reward gives a consistent advantage across benchmark families: this suggests that the reward-specific trends observed on the random-circuit evaluation are not stable when moving to structured benchmark circuits. Training on more diverse circuit families and circuit distributions could help the agents learn mapping strategies that generalize more reliably beyond the random circuits used in this work.

7 Responsible Research

This section discusses the main ethical considerations, reproducibility measures, limitations, and responsible use of AI tools in this work.

7.1 Ethical Considerations

This work does not involve human participants, personal data, or user-facing systems, and therefore does not raise direct privacy or data-protection concerns. The main ethical consideration is computational cost: reinforcement-learning training can require many simulation episodes, and this cost is expected to increase for larger circuits and hardware topologies. This motivates the use of fixed training budgets and small benchmark instances in this work. More generally, we hope that the computational cost and energy consumption of RL training will be treated as explicit considerations in future research on quantum compilation. The (n)-step shaped reward explored here represents one possible direction for reducing computational cost while retaining informative intermediate feedback, although its practical benefits should be evaluated more thoroughly at larger scales.

More broadly, improvements in compilation can increase the reliability and efficiency of quantum computations, therefore supporting potentially beneficial applications in areas such as scientific simulation, optimization, and materials research. At the same time, more capable quantum computing systems may also create societal and security risks, most notably by threatening currently deployed cryptographic methods. Although this work does not directly address these applications or their consequences, future research should consider not only technical performance, but also the wider implications of enabling increasingly capable quantum systems.

7.2 Reproducibility

To support reproducibility, the main experimental choices are fixed and reported explicitly. The PPO hyperparameters are listed in Appendix A, while the circuit distribution, hardware topologies, training seeds, evaluation setup, and Qiskit compilation pipeline are summarized in Appendix D. In addition, the complete codebase is made publicly available at `https://github.com/Federico-tech/BSc-Thesis/`, including the exact reward implementations, experiment-running scripts, evaluation pipeline, data-analysis procedures, and figure-generation code. This allows the full experimental workflow to be inspected and rerun, from reward computation and PPO training to the reproduction of the reported results.

7.3 Limitations

The main limitation of this work is the restricted 5-qubit setting. This makes exhaustive evaluation of all (5! = 120) initial mappings possible and enables comparison with metric-specific oracle solutions, but it limits how far the observed reward correlations and PPO performance can be generalized to larger devices, where the mapping space grows factorially and routing behavior becomes substantially more complex. The experiments are also limited to five synthetic hardware topologies with two-qubit gate fidelities sampled from a fixed range, rather than calibration data from real quantum devices.

A second limitation concerns the circuit representation. The rewards are computed before compilation from binary interaction graphs, which indicate only whether two logical qubits interact. They therefore discard interaction frequency, gate order, single-qubit operations, and gate dependencies. As a result, the rewards cannot fully represent the routing decisions that determine downstream compiled-circuit quality.

PPO controls only the initial mapping, while routing and scheduling are subsequently performed by Qiskit using SABRE and ALAP. The reported compiled metrics therefore reflect the interaction between the learned mapping and this fixed compilation pipeline, rather than the quality of the mapping independently of the downstream compiler. Moreover, Log ESP is an estimated reliability metric based only on two-qubit gate fidelities and does not model effects such as single-qubit errors, readout errors, decoherence, or correlated noise.

Finally, no PPO hyperparameter tuning was performed. This was a deliberate choice to compare reward formulations under a fixed training configuration, but it means that the reported performance should not be interpreted as the best achievable result for each reward.

7.4 LLM usage

Claude 4.8 was used as an auxiliary tool during this project, in accordance with the responsible-use guidelines of the CSE3000 Research Project course. Its use was limited to three categories: reformatting and debugging implementation errors in the experimental Python code, reflecting on research ideas and reviewing the manuscript for clarity, consistency, and language. All responses and suggestions generated by Claude were carefully evaluated and fact-checked against existing literature and external sources before being incorporated into the project. At no point was the model used to replace the author's own critical thinking or decision-making. All scientific decisions in this work were made independently by the author. The author assumes full responsibility for the methodology, implementation, analysis, and conclusions presented in this work.

8 Conclusion

This work investigated reinforcement learning for the initial mapping pass, focusing on whether graph-level rewards computed before full compilation are aligned with downstream compiled-circuit metrics and how agents trained with these rewards perform against established baselines for the objectives they are intended to optimize. We engineered and evaluated graph-level reward functions designed to target routing overhead, estimated circuit reliability, and their trade-off, using compiled SWAP count and estimated success probability as downstream objectives. Reward quality was assessed through reward–metric correlation analysis, while action-masked PPO agents trained with terminal, shaped, and (n)-step shaped reward variants were benchmarked against Random, SABRE, and metric-specific oracle baselines across multiple 5-qubit topologies.

The results show that the proposed rewards capture useful but incomplete information about downstream compiled-circuit quality, with correlations reaching at most $\rho = 0.48$. PPO outperforms Random on average, but remains behind SABRE and generalizes only weakly to the structured MQT Bench circuits. Two main conclusions follow.

(i) Graph-level objectives are insufficient without routing awareness. Binary interaction graphs omit gate order, repeated interactions, and gate dependencies, all of which affect routing and the final compiled metrics. The reward–metric analysis therefore reveals a structural gap between what these graph-level rewards can predict before compilation and the quality of the circuit after routing. Closing this gap requires richer circuit representations and rewards that model routing behavior more directly.

(ii) Reward engineering must consider both reward quality and learnability. Reward quality and reward learnability are distinct but equally important: a reward must align with downstream compiled-circuit metrics, but it must also provide a feedback signal that PPO can optimize effectively.

Overall, the results point toward a broader framework for RL-based initial mapping that evaluates both the quality of the reward objective and its learnability. Future work should also incorporate interaction frequency and gate order into the circuit representation, investigate routing aware-rewards, and evaluate whether PPO can optimize these rewards on larger hardware topologies and more diverse circuit distributions.

References

[1] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[2] Youngseok Kim et al. "Evidence for the utility of quantum computing before fault tolerance". In: *Nature* 618.7965 (2023), pp. 500–505.

[3] Lov K. Grover. *A fast quantum mechanical algorithm for database search*. 1996. arXiv: quant-ph/9605043 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/9605043.

[4] I. M. Georgescu, S. Ashhab, and Franco Nori. "Quantum simulation". In: *Reviews of Modern Physics* 86.1 (Mar. 2014), pp. 153–185. ISSN: 1539-0756. DOI: 10.1103/revmodphys.86.153. URL: http://dx.doi.org/10.1103/RevModPhys.86.153.

[5] Amira Abbas et al. "Challenges and opportunities in quantum optimization". In: *Nature Reviews Physics* 6.12 (Oct. 2024), pp. 718–735. ISSN: 2522-5820. DOI: 10.1038/s42254-024-00770-9. URL: http://dx.doi.org/10.1038/s42254-024-00770-9.

[6] Nicolas Gisin et al. "Quantum cryptography". In: *Reviews of Modern Physics* 74.1 (Mar. 2002), pp. 145–195. ISSN: 1539-0756. DOI: 10.1103/revmodphys.74.145. URL: http://dx.doi.org/10.1103/RevModPhys.74.145.

[7] John Preskill. "Quantum Computing in the NISQ Era and Beyond". In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. arXiv: 1801.00862 [quant-ph]. (Visited on 05/25/2026).

[8] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor". In: *Nature communications* 5.1 (2014), p. 4213.

[9] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. Nov. 2014. DOI: 10.48550/arXiv.1411.4028. arXiv: 1411.4028 [quant-ph]. (Visited on 05/25/2026).

[10] Frederic T Chong, Diana Franklin, and Margaret Martonosi. "Programming languages and compiler design for realistic quantum hardware". In: *Nature* 549.7671 (2017), pp. 180–187.

[11] Sumeet Khatri et al. "Quantum-assisted quantum compiling". In: *Quantum* 3 (May 2019), p. 140. ISSN: 2521-327X. DOI: 10.22331/q-2019-05-13-140. URL: http://dx.doi.org/10.22331/q-2019-05-13-140.

[12] Robert Wille, Stefan Hillmich, and Lukas Burgholzer. "Efficient and Correct Compilation of Quantum Circuits". In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2020, pp. 1–5. DOI: 10.1109/ISCAS45731.2020.9180791.

[13] Alexander Cowtan et al. "On the Qubit Routing Problem". en. In: vol. 135. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019, 5:1–5:32. DOI: 10.4230/LIPICS.TQC.2019.5. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.TQC.2019.5.

[14] Toshinari Itoko and Takashi Imamichi. *Scheduling of Operations in Quantum Compiler*. 2020. arXiv: 2011.04936 [quant-ph]. URL: https://arxiv.org/abs/2011.04936.

[15] Marcos Yukio Siraichi et al. "Qubit Allocation". In: *Proceedings of the 2018 International Symposium on Code Generation and Optimization*. CGO '18. New York, NY, USA: Association for Computing Machinery, Feb. 2018, pp. 113–125. ISBN: 978-1-4503-5617-6. DOI: 10.1145/3168822. (Visited on 06/01/2026).

[16] Diana Franklin and Frederic Chong. "Challenges in Reliable Quantum Computing". In: Jan. 2004, pp. 247–266. ISBN: 1-4020-8067-0. DOI: 10.1007/1-4020-8068-9_8.

[17] Prakash Murali et al. *Noise-Adaptive Compiler Mappings for Noisy Intermediate-Scale Quantum Computers*. 2019. arXiv: 1901.11054 [quant-ph]. URL: https://arxiv.org/abs/1901.11054.

[18] Gushu Li, Yufei Ding, and Yuan Xie. "Tackling the Qubit Mapping Problem for NISQ-Era Quantum Devices". In: *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS '19. Providence, RI, USA: Association for Computing Machinery, 2019, pp. 1001–1014. ISBN: 9781450362405. DOI: 10.1145/3297858.3304023. URL: https://doi.org/10.1145/3297858.3304023.

[19] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge, 1998.

[20] Rares Adrian Oancea et al. "Optimizing Initial Qubit Mappings Under Fixed Gate Error Rates Using Deep Reinforcement Learning". In: *International Conference on Innovations for Community Services*. Springer. 2025, pp. 189–208.

[21] Ankit Kulshrestha and Xiaoyuan Liu. *CO-MAP: A Reinforcement Learning Approach to the Qubit Allocation Problem*. 2026. DOI: 10.48550/ARXIV.2605.13638. (Visited on 06/01/2026).

[22] Ching-Yao Huang, Chi-Hsiang Lien, and Wai-Kei Mak. "Reinforcement Learning and DEAR Framework for Solving the Qubit Mapping Problem". In: *2022 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. Oct. 2022, pp. 1–9. (Visited on 05/25/2026).

[23] Yangzhi Li, Wen Liu, and Maoduo Li. "Deep Reinforcement Learning for Mapping Quantum Circuits to 2D Nearest-Neighbor Architectures". In: *Advanced Quantum Technologies* 7 (Jan. 2024). DOI: 10.1002/qute.202300289.

[24] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[25] Stan van der Linde et al. "qgym: A Gym for Training and Benchmarking RL-Based Quantum Compilation". In: *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Vol. 02. 2023, pp. 26–30. DOI: 10.1109/QCE57702.2023.10179.

[26] Shengyi Huang and Santiago Ontañón. "A Closer Look at Invalid Action Masking in Policy Gradient Algorithms". In: *The International FLAIRS Conference Proceedings* 35 (May 2022). ISSN: 2334-0762. DOI: 10.32473/flairs.v35i.130584. URL: http://dx.doi.org/10.32473/flairs.v35i.130584.

[27] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.

[28] Antonin Raffin et al. "Stable-baselines3: Reliable reinforcement learning implementations". In: *Journal of machine learning research* 22.268 (2021), pp. 1–8.

[29] Sinan Ibrahim et al. "Comprehensive Overview of Reward Engineering and Shaping in Advancing Reinforcement Learning Applications". In: *IEEE Access* 12 (2024), pp. 175473–175500. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3504735. (Visited on 05/10/2026).

[30] Ali Javadi-Abhari et al. *Quantum computing with Qiskit*. 2024. arXiv: 2405.08810 [quant-ph]. URL: https://arxiv.org/abs/2405.08810.

[31] Nils Quetschlich, Lukas Burgholzer, and Robert Wille. "MQT Bench: Benchmarking Software and Design Automation Tools for Quantum Computing". In: *Quantum* 7 (2023), p. 1062. ISSN: 2521-327X. DOI: 10.22331/q-2023-07-20-1062. URL: http://dx.doi.org/10.22331/q-2023-07-20-1062.

[32] IBM Quantum. *ALAPSchedule*. Qiskit 1.0 API reference. 2024. URL: https://quantum.cloud.ibm.com/docs/en/api/qiskit/1.0/qiskit.transpiler.passes.ALAPSchedule (visited on 06/11/2026).

[33] Charles Spearman. "The proof and measurement of association between two things." In: (1961).

[34] Nikiforos Paraskevopoulos et al. "SpinQ: Compilation Strategies for Scalable Spin-Qubit Architectures". In: *ACM Transactions on Quantum Computing* 5.1 (Dec. 2023), pp. 1–36. ISSN: 2643-6817. DOI: 10.1145/3624484. URL: http://dx.doi.org/10.1145/3624484.

A PPO hyperparameters

To isolate the effect of the reward formulation, all Maskable PPO agents were trained under the same fixed hyperparameter configuration. No hyperparameter tuning was performed, since the experiments are intended to compare reward formulations rather than optimize PPO performance. Table 2 summarizes the configuration used in the main experiments. Hyperparameters not explicitly specified were kept at their Stable-Baselines3 defaults.

Table 2: Maskable PPO training configuration used in the main experiments.

Hyperparameter	Value
Policy	MultiInputPolicy
Learning rate	3×10^{-4}
Rollout steps n_{steps}	2048
Batch size	64
Epochs n_{epochs}	10
Discount factor γ	0.99
GAE parameter λ	0.95
Clip range	0.2
Entropy coefficient	0.0
Value-function coefficient	0.5
Maximum gradient norm	0.5
Action masking	Enabled

B Additional Reward Formulations

This appendix presents the additional graph-level reward formulations used in the reward–metric alignment analysis of

Section 5.1. Although these rewards are not used in the main PPO experiments, they provide useful comparisons against the reward formulations discussed in the main text.

Direct adjacency reward. The direct adjacency reward counts how many logical interactions are mapped to directly connected physical qubits:

$$R_{\mathrm{adj}}(\pi) = \sum_{(q_i,q_j)\in E_I} \mathbb{I}[d_H(\pi(q_i),\pi(q_j)) = 1]\,.$$

Here, the indicator function $\mathbb{I}[\cdot]$ is equal to 1 if the condition is true and 0 otherwise. Therefore, this reward gives a positive contribution only when interacting logical qubits are mapped to adjacent physical qubits.

This reward encourages mappings in which interacting logical qubits can execute their two-qubit gates directly. Its limitation is that all non-adjacent placements are treated equally, regardless of whether the physical qubits are at distance two or farther apart.

Interaction-centrality rewards. We consider two interaction-centrality rewards: a degree-centrality reward and a closeness-centrality reward. Both are rank-based graph rewards based on the same assumption: logical qubits that are more central in the circuit interaction graph should be assigned to physical qubits that are more central in the hardware coupling graph.

For a graph $G = (V, E)$, degree centrality measures how many neighbours a node has:

$$c_{\mathrm{deg}}(v) = \deg_G(v),$$

where $\deg_G(v)$ denotes the degree of node v, i.e., the number of edges incident to v.

Closeness centrality instead measures how close a node is, on average, to all other nodes in the graph:

$$c_{\mathrm{close}}(v) = \frac{|V| - 1}{\sum_{u\in V,\, u\neq v} d_G(v,u)},$$

where $d_G(v,u)$ is the shortest-path distance between nodes v and u. Nodes with larger closeness centrality have shorter average graph distance to the rest of the graph.

For each logical qubit $q \in Q_L$, a centrality score $c_I(q)$ is computed on the interaction graph G_I. Similarly, for each physical qubit $p \in Q_P$, a centrality score $c_H(p)$ is computed on the hardware graph G_H. For the degree-centrality reward, c_I and c_H are degree-centrality scores, while for the closeness-centrality reward, they are closeness-centrality scores. The reward definition below is the same for both variants, as only the centrality score used to compute the ranks changes.

The centrality scores are converted into descending ranks. Let $\rho_I(q)$ denote the rank of logical qubit q, and let $\rho_H(p)$ denote the rank of physical qubit p, with rank 0 corresponding to the most central qubit. If multiple qubits have the same centrality score, they are assigned the same average rank.

Given an initial mapping π, the normalized rank mismatch for logical qubit q is

$$\Delta_{\mathrm{cent}}(q,\pi) = \frac{|\rho_I(q) - \rho_H(\pi(q))|}{n-1},$$

where $n = |Q_L| = |Q_P|$. The normalization by $n-1$ makes the maximum possible rank mismatch equal to 1.

The interaction-centrality reward is then defined as

$$R_{\mathrm{cent}}(\pi) = 1 - \frac{1}{n}\sum_{q\in Q_L} \Delta_{\mathrm{cent}}(q,\pi).$$

This reward is maximized when the rank ordering of logical-qubit centrality matches the rank ordering of physical-qubit centrality. It therefore does not depend on the absolute centrality values, but on how well the mapping preserves the relative centrality structure of the interaction and hardware graphs.

C Reward Timing Variants

The reward functions defined in Section 4.2 assign a graph-level score to a mapping. In addition to changing the objective itself, we study how the timing of the reward signal affects learning, following the reward-shaping perspective discussed in [29]. For each base reward R_{base}, we consider terminal, shaped, and (n)-step shaped variants.

Let π_t denote the partial mapping after t logical qubits have been assigned to physical qubits, and let π_n denote the completed mapping for a problem with n logical qubits. For partial mappings, $R_{\mathrm{base}}(\pi_t)$ is computed only over interaction edges whose two endpoints have already been assigned.

Terminal variant. In the terminal variant, the agent receives reward only after the complete mapping has been constructed:

$$r_t = \begin{cases} R_{\mathrm{base}}(\pi_n), & t = n, \\ 0, & t < n. \end{cases}$$

This gives feedback directly on the final mapping quality, but makes the reward signal sparse.

Shaped variant. In the shaped variant, the agent receives feedback after every assignment step. The reward is the difference between the score of the new partial mapping and the score of the previous partial mapping:

$$r_t = R_{\mathrm{base}}(\pi_t) - R_{\mathrm{base}}(\pi_{t-1}).$$

In other words, the agent is rewarded when an assignment improves the current partial mapping according to the chosen reward function. Over the full episode, these step-wise differences add up to the final score of the complete mapping, up to the initial score:

$$\sum_{t=1}^{n} r_t = R_{\mathrm{base}}(\pi_n) - R_{\mathrm{base}}(\pi_0).$$

Thus, shaped rewards provide feedback during the construction of the mapping, while keeping the same final objective.

(n)-step shaped variant. Finally, in the (n)-step shaped variant, feedback is emitted only every k assignment steps:

$$r_t = \begin{cases} R_{\mathrm{base}}(\pi_t) - R_{\mathrm{base}}(\pi_{t-k}), & t \equiv 0 \pmod{k}, \\ 0, & \text{otherwise.} \end{cases}$$

At the end of the episode, any remaining partial interval is also flushed so that the final completed mapping contributes to the reward. In this work, we use $k = 2$, meaning that the

agent receives intermediate feedback after every two logical-qubit assignments. This variant lies between terminal and fully shaped rewards: it provides more feedback than the terminal reward, but less frequent feedback than the shaped reward.

D Main Experimental Configuration

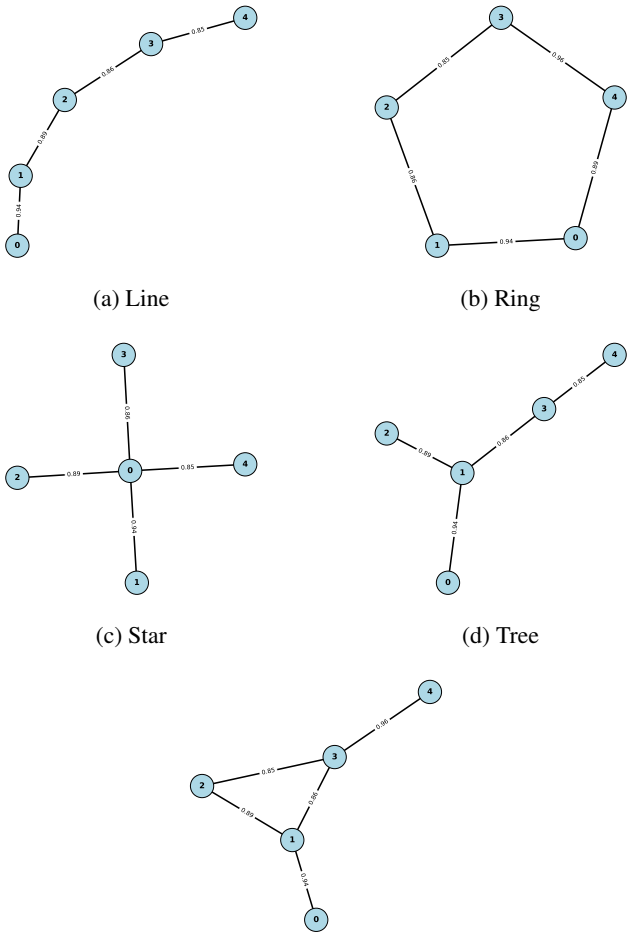
To ensure that differences in performance can be attributed to the reward formulation rather than to changes in the experimental setup, all main experiments use the same circuit distribution, hardware settings, training setup, and downstream compilation pipeline. Table 3 summarizes the fixed configuration used throughout the main experiments.

Table 3: Main experimental configuration used for training and evaluation.

Setting	Value
Number of qubits	5
Circuit depth	8
Two-qubit gate probability	0.6
Hardware topologies	line, ring, star, tree, partial mesh
Two-qubit fidelities	Uniform in $[0.85, 0.99]$
PPO seeds	0, 1, 2, 3, 4
Training timesteps	250,000
Evaluation circuits	100
Routing method	SABRE
Scheduling method	ALAP

E Hardware Topologies

The experiments use five synthetic 5-qubit hardware topologies: line, ring, star, tree, and partial mesh. Nodes represent physical qubits, while edges represent available two-qubit hardware connections.



(a) Line (b) Ring (c) Star (d) Tree (e) Partial mesh

Figure 7: Five 5-qubit hardware topologies used in the experiments.

F Absolute Compiled-Circuit Performance

Table 4 reports the absolute compiled-circuit performance corresponding to the relative comparisons presented in Table 1. Lower compiled SWAP count and higher Log ESP indicate better performance.

Table 4: Absolute compiled-circuit performance, reported as the mean $\pm$ standard deviation over topology–seed means.

Reward	**Variant**	**SWAP count**	**Log ESP**
Baselines			
	Random	4.06 ± 1.08	-2.51 ± 0.58
	SABRE	2.28 ± 0.44	-1.85 ± 0.33
PPO agents			
Distance	Terminal	3.21 ± 0.65	-2.20 ± 0.41
	Shaped	3.18 ± 0.65	-2.19 ± 0.42
	n-step shaped	3.20 ± 0.64	-2.20 ± 0.41
Fidelity	Terminal	3.61 ± 0.84	-2.33 ± 0.52
	Shaped	3.18 ± 0.58	-2.12 ± 0.39
	n-step shaped	3.21 ± 0.60	-2.13 ± 0.40
Hybrid	Terminal	3.17 ± 0.62	-2.14 ± 0.40
	Shaped	3.18 ± 0.62	-2.13 ± 0.40
	n-step shaped	3.19 ± 0.65	-2.14 ± 0.40
Metric-specific oracles			
	Oracle SWAP	$\mathbf{2.26 \pm 0.45}$	-1.84 ± 0.34
	Oracle Log ESP	2.31 ± 0.42	$\mathbf{-1.65 \pm 0.38}$