Elsayed Amer et al. *EPJ Quantum Technology* (2024) 11:19
https://doi.org/10.1140/epjqt/s40507-024-00225-1

EPJ Quantum Technology
a SpringerOpen Journal

RESEARCH Open Access

On the optimality of quantum circuit initial mapping using reinforcement learning

Norhan Elsayed Amer[1*], Walid Gomaa[1,2], Keiji Kimura[3], Kazunori Ueda[3] and Ahmed El-Mahdy[1,4,2]

*Correspondence:
norhan.elsayed@ejust.edu.eg
[1]Department of Computer Science and Engineering, Egypt-Japan University for Science and Technology, Alexandria, Egypt
Full list of author information is available at the end of the article

Abstract

Quantum circuit optimization is an inevitable task with the current noisy quantum backends. This task is considered non-trivial due to the varying circuits' complexities in addition to hardware-specific noise, topology, and limited connectivity. The currently available methods either rely on heuristics for circuit optimization tasks or reinforcement learning with complex unscalable neural networks such as transformers. In this paper, we are concerned with optimizing the initial logical-to-physical mapping selection. Specifically, we investigate whether a reinforcement learning agent with simple scalable neural network is capable of finding a near-optimal logical-to-physical mapping, that would decrease as much as possible additional CNOT gates, only from a fixed-length feature vector. To answer this question, we train a Maskable Proximal Policy Optimization agent to progressively take steps towards a near-optimal logical-to-physical mapping on a 20-qubit hardware architecture. Our results show that our agent coupled with a simple routing evaluation is capable of outperforming other available reinforcement learning and heuristics approaches on 12 out of 19 test benchmarks, achieving geometric mean improvements of 2.2% and 15% over the best available related work and two heuristics approaches, respectively. Additionally, our neural network model scales linearly as the number of qubits increases.

Keywords: Quantum computing; Controlled-NOT reduction; Proximal Policy Optimization; Classical optimization; Quantum circuit; Optimal initial logical-to-physical mapping; Qubit routing; Transpilation

1 Introduction

With the current rapid development in the quantum computing area, we are in the *Noisy Intermediate-Scale Quantum (NISQ)* era [1]. NISQ refers to noisy backends having total number of qubits up to few hundreds. While IBM has recently proposed a quantum chip with more than 1000 qubits [2], these qubits are still error prone and IBM is now shifting to designing fewer error-corrected qubits using a recent promising error-correcting codes [3].

In general, noise affects the reliability of executing a logical quantum circuit on a specific physical quantum processor, limiting the circuit size. Moreover, quantum processors

impose hardware-constraints on two-qubit gates applied to non-connected qubits. These result in additional 'swapping' gates, further affecting reliability.

Noise can be handled through both error correcting codes [3] and circuit optimization [4–9]. Error correcting code utilises many physical qubits for each logical bit to decrease the effective qubit error, at the expense of decreasing the available qubits. Circuit optimization utilizes many techniques to decrease the gate count of the circuit thus improving circuit fidelity. These optimization techniques include circuit depth reduction [4, 10, 11], gate cancellation and optimization [4, 12], and an initial logical-to-physical mapping selection [6, 7, 13] and routing [7, 9, 14–17]. Among which the initial mapping and routing are the most important problems.

This paper considers the initial mapping problem as there is limited research work, especially in utilizing reinforcement learning (RL) [13] for selecting a near-optimal logical-to-physical mapping, despite its importance. Additionally, reinforcement learning has been successful in several applications achieving human-level performance such as robotics [18], atari games [19], alpha-go [20], and alpha-zero [21]. Thus in this paper, we are concerned with utilizing reinforcement learning for selecting an optimal initial logical-to-physical mapping that would effectively reduce the number of additional two-qubit gates of a circuit to satisfy hardware constraints. This is accounted for by calculating the number of additional controlled-NOT (CNOT) gates, as it is the only two-qubit gate supported by IBM backends, after routing. Additional two-qubit gate optimization would decrease hardware-specific noise resulting from link errors connecting two qubits. Thus, two-qubit gates usually have higher error rates [22, 23] than single-qubit gates. This would also decrease qubit decoherence since we are decreasing additional depth.

1.1 Background

Logical quantum circuit is a quantum computation model consisting of a set of stages, where each stage consists of gates operating on distinct qubits. To execute a logical circuit on a specific hardware, the quantum compiler rewrites the circuit by transforming it using multiple passes, known as the *transpilation* process. According to Qiskit software [24], there are six passes, shown in Fig. 1: virtual optimization, gates decomposition, initial logical-to-physical placement, routing, translating to basis gates, and physical circuit optimization.

Virtual optimization optimizes the logical circuit itself before applying any hardware constraints. Gate decomposition decomposes three or more qubit gates into two-qubit

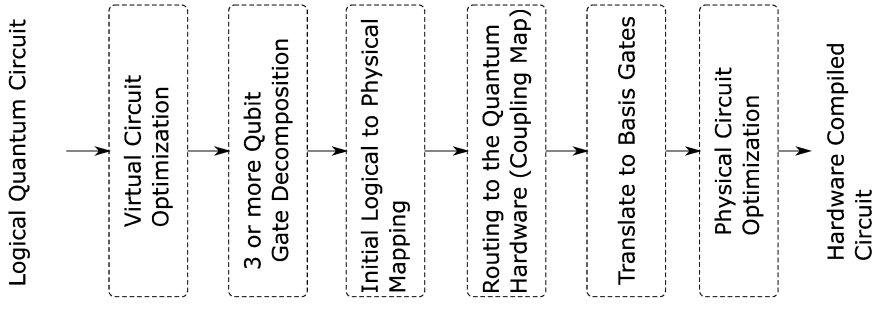
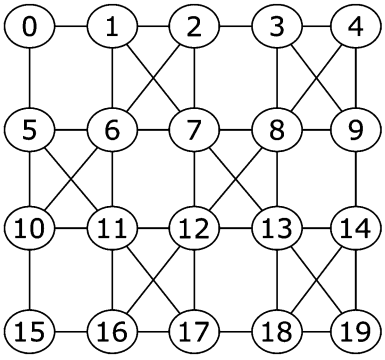


Figure 1 IBM Qiskit Transpiler Passes

Figure 2 Coupling Map for IBMQX20



gates. Initial logical-to-physical mapping permutes logical qubits while fixing physical ones. According to Qiskit [24], selecting a logical-to-physical mapping is crucial in reducing both additional gates and noise due to hardware constraints. Routing transforms the circuit by adding additional gates to satisfy limited hardware connectivity represented by the *coupling map* as shown for a 20-qubit hardware in Fig. 2. Nodes in the coupling map represent the physical qubits indices and edges represent the connectivity among qubits. Any two-qubit gate can only be executed if they are directly connected. This is achieved by adding swap or bridge gates which eventually increase the circuit gates. Finally, physical circuit optimization applies gate optimizations after the additional gates that were added by the routing stage.

Optimal initial mapping refers to the permutation of logical qubits that would have the least possible number of additional CNOTs for a given circuit after routing. As the number of qubits increases, obtaining an optimal initial mapping is of factorial complexity with respect to the number of qubits. Thus, current research studies [7, 13, 25] aim to select a *near-optimal initial mapping* that would decrease as much as possible additional CNOT gates. This would lead to the reduction of additional gate-specific errors, cross-talk among gates, circuit depth and qubit decoherence. This stresses the need for an intelligent approach capable of approximating the mapping between circuits' encodings and their initial mappings.

1.2 Problem statement

In this paper, we only aim to optimize the number of gates regardless of hardware error rates due to the limited availability of open-access quantum backends. However, the same approach may be generalized to any cost function such as maximizing state fidelity by selecting qubits with minimum error rates. Specifically, the mathematical formulation of our problem can be defined as shown in Eq. (1),

$$j^* = \underset{j}{\operatorname{argmin}}\, C(m, j) \tag{1}$$

where j^* is the index to the mapping having minimum CNOT count, j is a running index for all permutations of logical qubits that is of factorial complexity with respect to the number of qubits and $C(m, j)$ is the total count of CNOTs after mapping a given circuit m to the permutation of logical qubits j. This can be easily solved by exhaustive search in the case of small number of qubits. For example, when we have a 5-qubit circuit, there

are only 120 combinations. However, as the number of qubits increases, finding an optimal mapping is considered an NP-hard problem [26]. Thus, deep reinforcement learning can be beneficial for initial near-optimal logical-to-physical mapping where it approximates similar states (circuits) using neural networks and effectively learns from environment interactions. This would provide the routing pass with an optimal initial mapping to start from that would eventually further reduce the number of additional CNOTs than when starting with a random non-optimal mapping. Additionally, selecting an initial near-optimal logical-to-physical mapping would optimize CNOT gate count required to satisfy the limited connectivity of hardware. As applying a CNOT gate to non-adjacent qubits due to limited connectivity would result in additional CNOT gates resulting from the decomposition of additional swap and bridge gates, which increase both depth and hardware-specific errors. This eventually reduces additional hardware-specific errors as two-qubit gates usually have higher error rates than single-qubit gates.

1.3 Contributions and paper outline

Some available methods rely mainly on heuristics (as detailed in Sect. 2) to optimize each circuit separately from the beginning. Another line of research utilized reinforcement learning [13] along with transformer neural network to approximate similar circuits to the same initial mappings. However, these approaches suffer from being time consuming and unscalable. Additionally, available circuit encoding approaches [13] generally rely on the depth of the circuit, that is, as the size of the circuit increases in terms of the number of qubits and depth, the feature vector length also increases, hindering the scalability of the approach as the transformer neural network is infeasible for long sequences.

In the current research, we utilize reinforcement learning along with simple neural network and fixed-length feature vector that will lead towards more efficient and scalable approach to find a near-optimal logical-to-physical mapping. Additionally, we extract statistical features from the circuit based on the interaction patterns between any two qubits, referred to as *edges feature vector* in the rest of the paper. Specifically, we consider the question: Can a reinforcement learning agent learn to select a near-optimal logical-to-physical mapping given edges feature vector? This would provide a scalable approach for circuit representation as the number of qubits and the circuit depth increase. This would also improve execution time as the reinforcement learning agent will approximate similar circuits to the same initial mappings instead of optimizing each circuit separately as in heuristics approaches. Additionally, it will better guide available qubit routing algorithms.

The results demonstrate that our reinforcement learning agent coupled with a simple routing evaluation is capable of generalizing to test benchmarks outperforming state-of-the-art approaches [9, 13, 25] in 12 out of 19 test benchmarks on a 20-qubit hardware architecture, shown in Fig. 2. Even for the remaining 7 circuits our method was either comparable with or ranked the second best achieving minimum number of CNOTs in 6 circuits among the three other related works. This emphasizes that our feature vector, allows scalability given its fixed-length to deeper circuits, is capable of outperforming other complex representations from related work [13] without constraining the circuit depth while fixing the number of qubits. So that our feature vector representation works for all qubits up to the maximum number of qubits supported by the hardware.

In summary, our contributions are as follows:

1. Proposing a generic reinforcement learning agent, that is not circuit-specific, for a 20-qubit hardware architecture capable of selecting a near-optimal initial mapping that further reduces the number of additional gates over the best available related work approaches.
2. Developing a fixed-length feature vector whose length is independent of the circuit depth relying only on all possible two-qubit combinations.
3. Demonstrating the generalizability and scalability of our agent to unseen benchmark circuits.

The rest of this paper is organized as follows. Section 2 surveys available state-of-the-art approaches. Section 3 presents the dataset collection, describes the reinforcement learning algorithm used, presents the reward scheme, and explains the experimental design. Section 4 compares and analyzes the performance of our RL agent in comparison to state-of-the-art work on test benchmarks. Finally, Sect. 5 summarizes the paper and provides future work directions.

2 Related work

In this section, we discuss available state-of-the-art research work regarding circuit optimizations. This is also summarized in Table 1.

Some research work did not consider the noise inherent in qubits and gates but mainly focused on reducing the number of gates and decreasing circuit depth given the backend's coupling map [5–9, 11, 14, 25]. Li et al. [7] introduced swap-based bidirectional heuristic search algorithm (*SABRE*). They proposed a routing and an initial layout selection methods. For the initial layout selection method, they scan the circuit bidirectionally (forward and backward) to reach the updated initial mapping. Cheng et al. [9] proposed both placement and routing methods. For the placement method, they utilized a nearest neighbor approach where they calculate the degree of activity (DoA) which is the total number of CNOT gates applied to each qubit. This is then sorted in descending order prioritizing qubits having more interactions to be placed first. One limitation is that their proposed nearest neighbor approach relies on the hardware being a 2D grid architecture, which is not usually the case as mentioned in [27]. Zhu et al. [25] proposed a heuristic approach for initial mapping selection. Specifically, they rely on an interaction graph of the quantum circuit that only considers the first occurrence of each two-qubit interactions. Edges of the interaction graph are weighted with the stage number at which the first occurrence took place. Our proposed feature representation is similar in the usage of the interaction graph; however, we extract statistical features from the whole circuit, not just the first occurrence of a two-qubit interaction. They consider an expansion from the center where

Table 1 Summary of related work approaches for initial logical-to-physical mapping selection

Reference	Proposed	Limitations
[5–9, 11, 14, 25]	-Noise unaware heuristics	-Circuit-specific lacking generalization and scalability
[22, 28–31]	–Noise aware heuristics	–Computationally expensive
[13]	–Transformer neural network for the Reinforce agent	–Inability to handle very long sequences due to the usage of transformer neural network model –Transformers are known for being computationally expensive and non-scalable

they map the former qubit of the first two-qubit gate to the center of the physical qubit in the coupling map. Then, they adopt a breadth-first search for selecting the next logical qubit based on the gate occurrence index to be mapped to available candidate neighbors. Fu et al. [11] proposed a combined heuristics approach for mapping and routing for depth reduction optimization. Specifically, they utilize a sliding window to divide the circuit to sub circuits where they rely on A^* search approach for depth optimization for each sub circuit. However, their approach is non-scalable having an exponential complexity of $O(\text{poly}(d)\exp(\text{poly}(S_d)))$, where d is the circuit depth and S_d is the window size. All the mentioned heuristic approaches are circuit-specific lacking generalization and scalability.

Other researchers improved on the previous noise-unaware methods [22, 28–31]. Niu et al. [29] adjusted the work by [7] to take into account hardware-specific noise. They proposed hardware-aware simulated annealing (*HSA*) method for initial mapping selection and hardware-aware heuristic mapping (*HA*) method for routing gates. Zhu et al. [15] were also inspired by the use of qubit routing to update the qubit initial placement in addition to the bidirectional scanning of a circuit proposed by [7]. Specifically, they relied on a reduced-form of a circuit in addition to appending its inverse to it that is then fed to their initial placement algorithm. Their algorithm relies on a noisy simulator to evaluate a given circuit with an initial mapping. Their initial placement algorithm has a worst case complexity of $O(I \cdot J \cdot |V|^6)$, where I is the outer loop number of iterations that provide a random starting mapping, J is the inner loop number of iterations required to update the given random starting mapping using qubit routing algorithm, and $|V|$ is the number of physical qubits. To benefit from different initial mappings generated from their initial placement algorithm, they propose a multi-agent qubit routing such that each agent starts from a different mapping of the same circuit. Tannu et al. [30] proposed variation-aware qubit movement and variation-aware qubit allocation. For the variation-aware qubit movement, their proposed method relies on Dijkstra's algorithm to calculate the distance between pairs of qubits such that paths with maximum failure rates, based on calibration data, are avoided. For the variation-aware qubit allocation, the initial mapping is assigned by allocating the most used qubits' pairs to high reliability links. Dury et al. [31] proposed a qubit allocation approach based on simulated annealing. Although these approaches consider hardware-noise, they are computationally expensive.

Other studies aimed to utilize reinforcement learning. Huang et al. [13] proposed reinforcement learning for initial logical-to-physical mapping in addition to a heuristics approach for qubit routing that utilizes A^* search. For the initial placement, they formulated the problem as sequence-to-sequence mapping where they utilized a transformer neural network for the Reinforce agent, a policy gradient reinforcement learning approach. Specifically, their input is a gate sequence, the enumeration of each two-qubit gate based on the layer it is executed in, and a layer sequence, the layer number such that it contains non-overlapping two-qubit gates. One limitation of their work is that they rely on the transformer neural network model due to its inability to handle very long sequences. The authors did not mention the number of neural network parameters used but transformers are known for being computationally expensive and non-scalable.

3 Models and methods

In this section, we first detail the dataset collection process that is composed of both benchmarks and random circuit generation. Second, we describe the reinforcement learning proximal policy optimization algorithm approach in addition to explaining its variant

that we employed in this paper. Third, we explain our reward scheme that we used during training our reinforcement learning agent. Fourth, we demonstrate the actor-critic neural network that was used to guide the agent by providing the actions' probability distribution and the value of the current state. Finally, we describe the methodology used.

3.1 Dataset collection

Our dataset consists of a combination of random circuits and benchmark circuits. For the random circuits, we generated circuits with qubits up to 20 qubits. Each circuit has a random depth of up to 1000. The random circuit consists of random IBM basis gates, gates supported by the hardware, that can be broadly divided into single qubit gates and two qubit gates. Single qubit gates are gates that can be executed directly on a qubit as they do not rely on the availability of connections between qubits, but are affected by noise from the gate itself in addition to noise inherent in the qubit it is executed on. These gates include identity, rotation Z, square-root NOT, and NOT. Two qubit gates are affected by both the availability of connections between the two involved qubits and the noise inherent in the link connection resulting from executing that gate. For the IBM quantum backend, the controlled NOT gate is the only two qubit gate supported as a basis gate such that a swap gate can be decomposed into its equivalence using three controlled NOT (CNOT) gates.

Benchmark circuits were collected from multiple sources: QASMBench [32], IBM QX-Circuit [33] and Munich Quantum Toolkit Benchmark Library (MQT) [34]. Gates in these circuits were decomposed into IBM basis gates. For benchmarks, we split circuits with depths larger than 1000 into smaller circuits in order to speedup the transpilation process during training. For benchmark training circuits, qubits were logically shuffled to enlarge the dataset. The total training set having both random and benchmark circuits consists of 84,400 circuits. We separate the same 19 benchmark circuits from training dataset as used in [13] for testing in order to compare with them. These 19 circuits were not used in training. These testing circuits were neither split nor shuffled. For the training and testing datasets, we only consider CNOT gates as single qubit gates will not affect the routing process.

3.2 Maskable Proximal Policy Optimization (Maskable PPO)

Maskable PPO [35] is a variant of the classical PPO [36] that masks invalid actions depending on the state. Classical PPO is a *policy gradient reinforcement learning* algorithm, which utilizes an actor-critic scheme, that updates the policy parameters to maximize the expected return. The actor is used to provide a probability distribution over the available actions while the critic provides an estimate of the expectation of rewards at a certain observation state. During training, PPO avoids large policy deviations by utilizing a clip ratio ϵ that maintains the change between the old and new policies to be in the range $[1-\epsilon, 1+\epsilon]$. The main goal for the reinforcement learning agent is to maximize the expected cumulative future rewards given a discounted factor of γ, ranging from 0 to 1, shown in Eq. (2), where $G(s_t, a_t)$ is the discounted cumulative rewards from time t of at certain state s_t, R is the reward for a given state and action pair.

$$\begin{aligned} G(s_t, a_t) &= R(s_t, a_t) + \gamma R(s_{t+1}, a_{t+1}) + \gamma^2 R(s_{t+2}, a_{t+2}) + \cdots \\ &= \sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k}) \end{aligned} \tag{2}$$

The critic neural network is trained to estimate this discounted cumulative return given a state. The difference between the actual discounted reward and the estimated discounted reward by the value neural network is known as the advantage as shown in Eq. (3), where $A(s_t, a_t)$ is the advantage at a certain timestep t and certain state s_t, $G(s_t, a_t)$ is the actual discounted cumulative rewards from time t of certain state s_t, and $V(s_t)$ is the critic's state value function (V) estimate of the discounted reward for state s_t at timestep t.

$$A(s_t, a_t) = G(s_t, a_t) - V(s_t) \tag{3}$$

This advantage demonstrates how effective the action the agent took was. Additionally, a ratio term, shown in Eq. (4), is calculated between the new policy and the old policy for an action given a state. This ratio term results in a value between 0 and 1 if the action for a state is less probable than in the old policy and a value larger than 1 if the action for a state is more probable than in the old policy.

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta\text{old}}(a_t|s_t)} \tag{4}$$

If the normalized advantage multiplied by the ratio term is more than $1 + \epsilon$ or lower than $1 - \epsilon$, the policy gradient step gets clipped according to Eq. (5).

$$L^{\text{Clip}}(\theta) = \hat{E}_t\big[\min\big(r_t(\theta)A(s_t, a_t), \text{clip}\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big)A(s_t, a_t)\big)\big] \tag{5}$$

This aids in preventing large updates to the policy as PPO learns from online samples so large updates can cause policy instability between different batches.

In this paper, we are concerned with qubit assignment problem. Thus, Maskable PPO would be of benefit as we fix a particular qubit at each step in an episode. This would prevent action loops if we used classical PPO that allows choosing any action at any step of an episode which would increase the number of steps required for each episode. Thus, the usage of Maskable PPO would result in fixed number of steps per episode equal to $n - 1$, where n is the number of qubits, allowing decreasing number of actions at each step. We employ Maskable PPO where valid actions rely on the current timestep of the agent. So, the action space is parameterized by the current timestep. Specifically, it can be demonstrated on a 5-qubit circuits as shown in Fig. 3, where we have $n - 1$ timesteps as depicted by the tree's height. At each tree level, there is a set of valid actions, where the action at certain tree level corresponds to certain logical qubit index. This is shown in Eq. (6), where LQI is the logical qubit index, a is the current action, level represent the current tree level number, n represents the number of qubits and k runs from 0 to the current step number.

$$LQI(a, \text{level}) = a - \left(\sum_{k=0}^{\text{level}-1} (n-k)\right) + \text{level} \tag{6}$$

Initially, we start with the corresponding mapping or identity mapping. After that we fix a qubit at each timestep by doing a swap between the logical qubit corresponding to the physical qubit (P) at the current timestep or level and the logical qubit (L) at the intended action (L_{LQI}). The total number of actions is $\frac{n(n+1)}{2} - 1$, where n is the number of qubits.

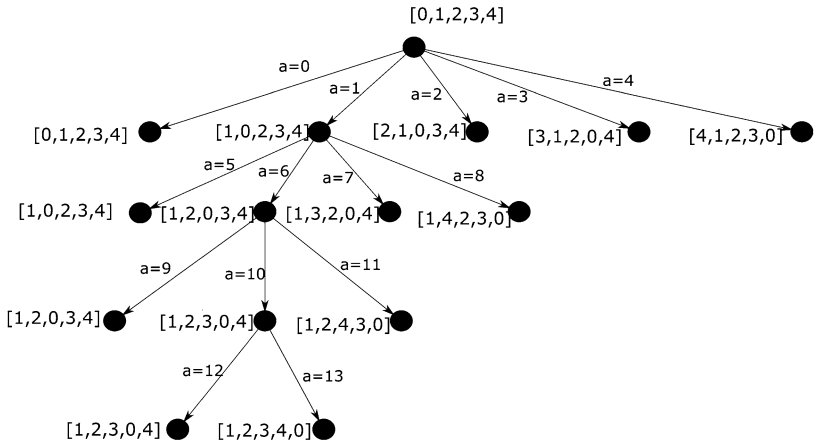


Figure 3 Action space example for 5-qubit

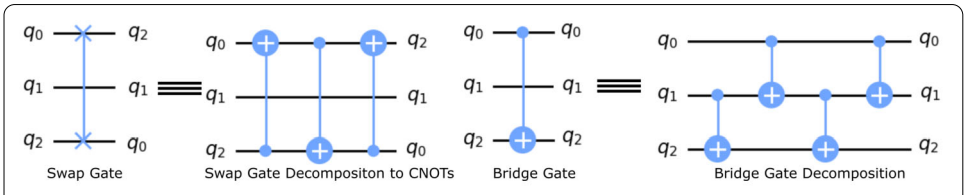


Figure 4 Swap and bridge gate decomposition into CNOTs

An example scenario, shown in Fig. 3 of the reinforcement learning agent is simplified as follows:

1. Agent starts with identity mapping
   Mapping = $[L_0 : P_0, L_1 : P_1, L_2 : P_2, L_3 : P_3, L_4 : P_4]$, level = 0
2. Agent takes action 1 (swap logical qubit at P_0 and $L_1 = L_{1-0+0}$)
   Mapping = $[L_1 : P_0, L_0 : P_1, L_2 : P_2, L_3 : P_3, L_4 : P_4]$, level = 1
3. Agent takes action 6 (swap logical qubit at P_1 and $L_2 = L_{6-5+1}$)
   Mapping = $[L_1 : P_0, L_2 : P_1, L_0 : P_2, L_3 : P_3, L_4 : P_4]$, level = 2
4. Agent takes action 10 (swap logical qubit at P_2 and $L_3 = L_{10-9+2}$)
   Mapping = $[L_1 : P_0, L_2 : P_1, L_3 : P_2, L_0 : P_3, L_4 : P_4]$, level = 3
5. Agent takes action 13 (swap logical qubit at P_3 and $L_4 = L_{13-12+3}$)
   Mapping = $[L_1 : P_0, L_2 : P_1, L_3 : P_2, L_4 : P_3, L_0 : P_4]$, level = 4

3.3 Reward

Our goal is to maximize the expected cumulative reward such that the agent takes steps towards the mapping with the minimum additional CNOT gates. Additional CNOT gates are the decomposition of higher level additional gates such as the Swap gate and Bridge gate, shown in Fig. 4. The Swap gate is decomposed into three CNOT gates, where it swaps two qubits in order to be able to perform a CNOT gate applied on non-adjacent qubits. The Bridge gate performs the CNOT gate between non-adjacent qubits while keeping the two qubits in their original position, where it can only be performed if the two qubits are at a distance of 2 in the coupling map. It is decomposed into four CNOT gates.

Algorithm 1 Reward Scheme

1: **function** REWARD(LogicalCNOT, CorrespondingCount, Suboptimal, CNOTCount)
2: **if** $\frac{\text{CNOTCount} - \text{LogicalCNOT}}{\text{LogicalCNOT}} < \text{threshold}$ **then**:
3: reward $\leftarrow$ Const
4: Suboptimal $\leftarrow$ CNOTCount
5: **else if** Suboptimal > CNOTCount **then**
6: reward $\leftarrow \frac{\text{Suboptimal}}{\text{CNOTCount}} + \frac{\text{CorrespondingCount}}{\text{CNOTCount}}$
7: Suboptimal $\leftarrow$ CNOTCount
8: **else if** Suboptimal < CNOTCount **then**
9: reward $\leftarrow -\frac{\text{CNOTCount}}{\text{Suboptimal}}$
10: **else**
11: reward $\leftarrow 0$
12: **end if**
13: **return** Suboptimal, reward
14: **end function**

For each step, our agent obtains an immediate reward. The immediate reward helps the agent learn faster because if the agent will wait till the end of the episode, given all possible action combinations at each step in an episode, then it would take a long time for the agent to receive a positive reward. The immediate reward scheme function, is shown in Algorithm 1, that is invoked at each step in the episode. The function parameters are the *LogicalCNOT* represents the number of CNOTs in the logical circuit before any mapping or routing, *CorrespondingCount* represents the number of CNOTs after mapping the circuit to the corresponding mapping or identity mapping followed by routing to satisfy the hardware constraints, and *Suboptimal* initially is equal to the *CorrespondingCount* but after that it is updated to the least CNOT counts found until the current timestep. The *Suboptimal* is used to represent the best found CNOT count values across the episode, since finding the absolute optimal is of factorial complexity with respect to the number of qubits. The *CNOTCount* is the additional number of CNOTs after mapping the logical circuit to the mapping at the current step followed by the routing.

Lines 2–12 represent the reward given at each step in an episode. The *CNOTCount* is compared with the increase rate in CNOTs over the *LogicalCNOT*; if it is less than a *threshold*, set to 0.5 based on trial and error to compromise between the agent receiving a positive reward and an acceptable number of additional CNOT gates, then we give an immediate reward represented by *Const*, set to 100 based on trial and error, otherwise it is a small positive, a small negative or a zero reward. If the *CNOTCount* of the current mapping is less than the counts of the best seen mapping (*Suboptimal*) for the circuit, we give a positive reward consisting of the ratio between the *Suboptimal* and *CNOTCount* that is added to the ratio of *CorrespondingCount* and *CNOTCount*. This is to quantify a reward based on how much the current mapping is better than both the initial mapping from the initial step and the best seen mapping till the current step. If the *CNOTCount* of the current mapping is greater than the *Suboptimal*, a negative reward is given with a ratio between the *CNOTCount* and the *Suboptimal*. If the *CNOTCount* is equal to *Suboptimal*, but the percentage increase between the starting value represented as *LogicalCNOT* and

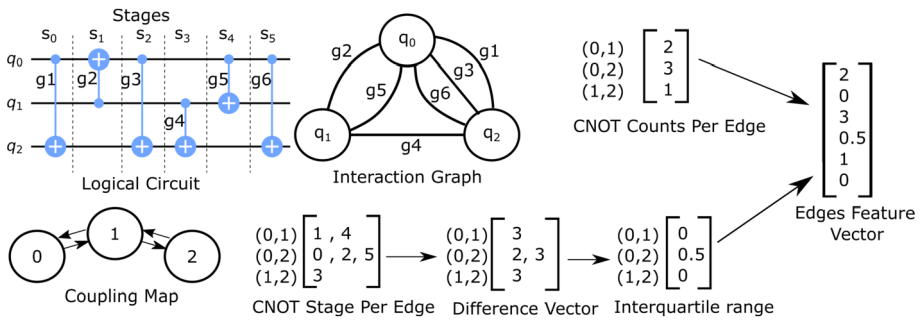


Figure 5 3-Qubit edges encoding example

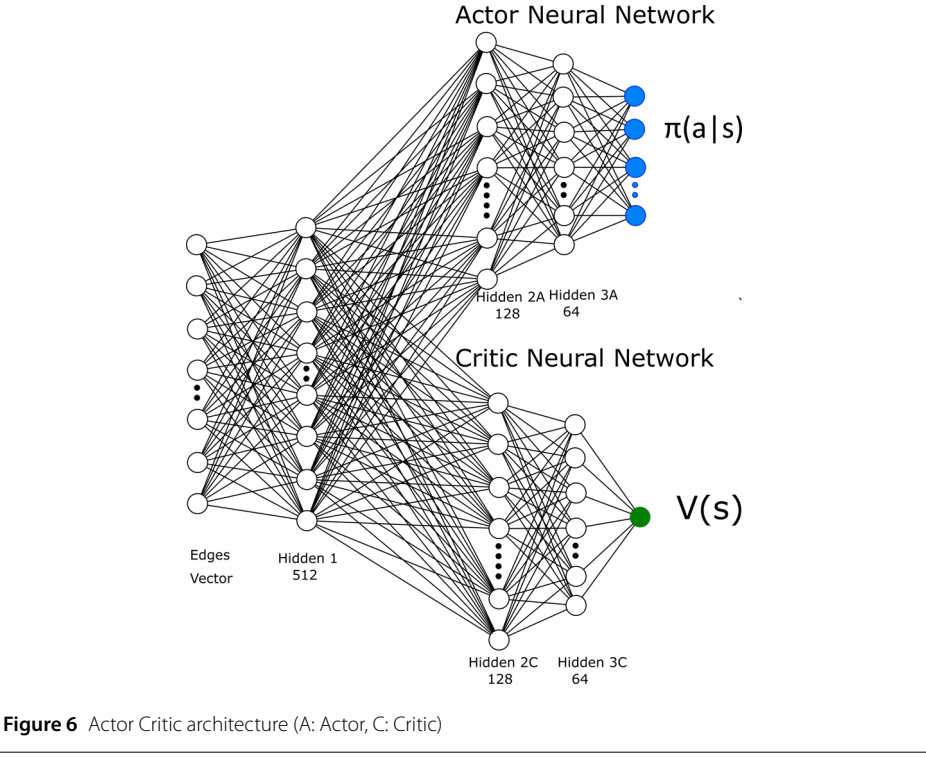


Figure 6 Actor Critic architecture (A: Actor, C: Critic)

the final value represented as *CNOTCount* is larger than the *threshold*, a reward of zero is given to the agent.

3.4 Proposed model

Selecting an optimal logical-to-physical mapping given a feature vector is a non-trivial task. Each circuit is considered a separate Markov Decision Process (MDP) so finding a feature vector representation such that similar circuits can be closely approximated by a neural network is essential. The best results were produced by extracting features from the interaction graph, as mentioned in [37], such as counting the number of interactions between two qubits along with the interquartile range of the difference vector between successive CNOTs indices, as shown in Fig. 5. This representation is scalable as it has a fixed-length.

For the RL agent, the best approach we found was having different sets of actions at each level of the tree, shown in Fig. 3, where we swap the qubit at the current tree level by the qubit indexed by the taken action. This is semantically correct as each action performs the same swap at each time it is taken since it is fixed per tree level.

Our model consists of three fully connected layers, shown in Fig. 6. The first fully connected layer, having 512 neurons, is shared between the actor and the critic neural networks. After that, they are split into two neural networks, where each of the actor and critic has two consecutive fully connected layers having 128 and 64 neurons, respectively. This is then followed by an output layer. All activation functions are rectified linear units (ReLU) except for the output layers. The mentioned hyperparameters were selected based on trial and error.

For the actor, the output layer has a number of neurons equal to all available actions so that the actor predicts the action accordingly after masking invalid ones. Specifically, an action mask is applied to the action logits, raw outputs of the final layer in the neural network before applying activation function, to set invalid actions to $-\infty$. So that after applying the softmax activation function, sampling of invalid actions will be 0.

For the critic, the output layer is composed of only one neuron with linear activation function, where it estimates the expected discounted summation of returns for a specific state.

The input to the actor-critic model is the edges vector, as demonstrated for a three-qubit circuit in Fig. 5. Every group of gates that operate on distinct qubits construct a stage such that the maximum number of stages represents the depth of the circuit. Our feature vector consists of the aggregation of CNOT gates on a certain edge in addition to the interquartile range of the difference vector between successor CNOTs based on its stage number. The edges feature vector has a length corresponding to all two-edge combinations multiplied by 2 as we consider two statistical features (CNOT counts per edge, interquartile range), resulting in a feature vector length of $2\binom{q}{2}$, such that if we have 20-qubit circuit, the length of the corresponding edges vector is 380.

The *CNOT Counts Per Edge* statistical feature is based on summing the number of interactions between each two qubits. The interquartile range is calculated by first finding the stage indices in which two-qubit interactions occurred between each pair of qubits, constructing the *CNOT Stage Per Edge Vector*. Then, we get the difference between successor CNOTs stages for each pair of qubits resulting in the *Difference Vector*, from which we estimate the interquartile range. The final *Edges Feature Vector* is constructed by merging both *CNOT Counts Per Edge Vector* and *Interquartile range Vector* row-wise.

3.5 Experimental design

We generated random circuits using the IBM Qiskit software [24]. We did not only rely on benchmark circuits so as to avoid bias to certain class of circuits. We trained our Maskable PPO agent using Stable-baselines3 [38] on our custom environment for a 20-qubit hardware, shown in Fig. 2. Specifically, the observation space consists of positive continuous values represented in the edges feature vector, as shown in Fig. 5. Thus the observation space shape has a vector length of $2\binom{q}{2}$, where q is the number of qubits, representing all possible two-qubit combinations for two statistical features.

The observation state is the edges encoding of a given circuit permuted to a given mapping constructing the edges feature vector. Initially and at the beginning of each episode,

the observation state is reset to the edges feature vector for the circuit at the identity mapping. After that the state gets updated based on the action taken that changes the mapping permutation resulting in the change of the circuit's edges feature vector. This would result in a scalable input representation that would only rely on the possible edges combinations without being constrained by the depth of the circuit. So, for 20 qubits, we have 209 possible actions. We utilize a learning rate of 0.0001 and a discount factor of 0.99. Since we are fixing a qubit at each tree level, the steps per episode is equal to $n-1$, where n is the number of qubits. So, the steps per episode is fixed to 19 for a 20-qubit circuit.

For testing, we start by the initial state of the logical circuit. At each timestep, the agent predicts an action based on the state and on valid actions at the current timestep; thus updating the current state to a new one. We map and route the circuit corresponding to the new state to get the total number of CNOTs after satisfying the hardware constraints. The routing pass is the most time-consuming task in our reinforcement learning approach; however, we plan as future work to speed up the routing pass by utilizing neural networks to approximate its cost. The maximum number of steps is $n-1$, where n is the number of qubits. Each of the $n-1$ steps represents a logical-to-physical qubits permutation for a given circuit among which we choose the permutation having the minimum total number of CNOTs. Since we formulate the problem as infinite horizon, our agent during training will favor long-term solutions. During decision making in the testing phase, we are no longer optimizing our model and we have a source of error from the usage of neural networks. Thus, our decision is based on the mapping having the minimum number of additional gates across the episode. This is represented in Eq. (7), where the near-optimal initial logical-to-physical mapping is the mapping having minimum number of CNOTs after routing among the $n-1$ possible mappings in an episode, where k is a running index to mappings that the agent take across an episode, C is the total number of CNOTs after mapping a given circuit m to permutation of logical qubits k followed by routing, and $j*$ is the index to the mapping having minimum number of CNOTs among the $n-1$ possible mappings which corresponds to the maximum steps per episode.

$$j* = \underset{k}{\operatorname{argmin}}\, C(m,k) \tag{7}$$

4 Results and discussion

4.1 Results

To evaluate the performance of our agent, we compare the additional number of CNOTs with Cheng et al. [9], Zhu et al. [25], and Huang et al. [13]. We also calculate the geometric mean of the ratio, shown in equation (8), between each approach with each of the related work [9, 13, 25] approaches. Equation (8) calculates the geometric mean across the 19 benchmark test circuits, k is the index to the benchmark test dataset running from 0 to 18, between the ratio of the number of additional gates between approaches i and j.

$$\text{GeoMean}(\text{approach}_i, \text{approach}_j) = \prod_{k=0}^{18} \frac{\text{approach}_j[k]}{\text{approach}_i[k]} \tag{8}$$

The geometric mean is used to better illustrate the global effect of an approach over another as we are dealing with ratios. We also compute the average number of additional

Table 2 Comparison for feature vector length for benchmark test circuits

Benchmark	Number of logical CNOTs	Feature Vector Length	
		Huang et al. [13]	RL
clip_206	14,772	14,772	380
cm85a_209	4986	4986	380
cycle10_2_110	2648	2648	380
dist_223	16,624	16,624	380
hwb6_56	2952	2952	380
hwb7_59	10,681	10,681	380
hwb8_113	30,372	30,372	380
mlp4_245	8232	8232	380
radd_250	1405	1405	380
rd73_252	2319	2319	380
rd84_253	5960	5960	380
root_255	7493	7493	380
sao2_257	16,864	16,864	380
sym10_262	28,084	28,084	380
sym9_148	9408	9408	380
sym9_193	15,232	15,232	380
Urf1_278	26,692	26,692	380
Urf2_277	10,066	10,066	380
Urf5_280	23,764	23,764	380

gates for [9, 13, 25] in comparison with our work. We calculate for each benchmark circuit the percentage of improvement over the best of related work [13] in terms of the total number of CNOTs after routing a given circuit from a given initial mapping.

Since the related work using reinforcement learning [13] is not open sourced, it can not be implemented faithfully. Therefore, we use the same test benchmarks along with the same T|ket⟩ compiler [39] using its default routing pass [40] to route the circuit given a certain initial mapping assuming no optimizations. We also took the results on the test benchmarks for the three related works [9, 13, 25] from [13].

Table 2 compares the feature vector length required for each of the benchmark circuits between our reinforcement learning approach with that of huang et al. [13], as this is the only related work approach utilizing neural networks. For all test benchmarks, our representation is more efficient and scalable over the related work approach [13]. Specifically, our reinforcement learning approach requires a fixed-length input of only 2 $\binom{q}{2}$, where q is the number of qubits, that requires minor preprocessing time to extract the statistical features; whereas, the feature vector representation by Huang et al. [13] has a variable-length equal to the number of logical CNOTs in a circuit.

Table 3 shows that our RL agent outperforms the related works in 12 out of 19 benchmark circuits achieving minimum number of additional CNOT gates among related work approaches [9, 13, 25], while utilizing a shorter feature vector length that is independent of the circuit size. This signifies that the reduction of the input complexity did not affect the agent's capability in finding a good initial mapping. The worst performance for our agent is for benchmark hwb6_56. This circuit has 7-qubit which is the least number of qubits among test benchmark circuits. Since the training dataset consists of circuits with high complexities, the neural network may overestimate circuits with low complexities as detailed in Sect. 4.2 for a 9-qubit model comparison. As represented by our results, our model tends to have better results as the complexity of the circuits increases. However, our result for this benchmark is still comparable with the other heuristics approaches. For the 6 out of the 7 remaining benchmark circuits, our method serves as either comparable with

Table 3 Benchmark test circuits for the 20-qubit model

Benchmark	Logical CNOTs#	Δ CNOTs after mapping and routing				Improv %
		Huang et al. [13]	Cheng et al. [9]	Zhu et al. [25]	RL	
clip_206	14,772	7959	9339	9444	**7572**	**1.7**
cm85a_209	4986	2706	3075	2907	**2322**	**4.99**
cycle10_2_110	2648	**1293**	1440	1611	1449	−3.99
dist_223	16,624	8082	10,668	10,569	**7791**	**1.17**
hwb6_56	2952	**1143**	1665	1740	1668	−12.82
hwb7_59	10,681	**4950**	5226	5619	5129	−1.14
hwb8_113	30,372	17,319	17,886	20,007	**16,905**	**0.86**
mlp4_245	8232	5511	**4881**	5862	4932	**4.21**
radd_250	1405	900	993	1065	**834**	**2.86**
rd73_252	2319	1323	1344	1272	**1239**	**2.30**
rd84_253	5960	3381	3912	3804	**2940**	**4.72**
root_255	7493	3531	4554	4164	**3315**	**1.95**
sao2_257	16,864	7812	11,268	9999	**7527**	**1.15**
sym10_262	28,084	**13,833**	16,899	17,079	14,340	−1.2
sym9_148	9408	3033	4200	3366	**2306**	**5.55**
sym9_193	15,232	8676	9252	9462	**8496**	**0.75**
Urf1_278	26,692	17,391	17,106	17,217	**16,959**	**0.97**
Urf2_277	10,066	6129	6168	**6108**	6805	−4.35
Urf5_280	23,764	**13,791**	15,261	15,201	13,845	−0.14
Geo. Mean		1	1.145	1.148	**0.978**	
of Addit. Gates		0.873	1	1.002	**0.854**	
		0.870	0.997	1	**0.852**	
Avg. # of		6777	7638.7	7710.3	**6651.2**	
Addit. Gates		±5347.24	±5753.6	±5979.35	**±5330.34**	

or ranked the second best approach having a low number of CNOTs. Furthermore, our method reduced the number of gates over the best available related work [13] achieving a geometric mean of 2.2% and outperformed the other two heuristics approaches [9, 25] achieving a geometric mean of 15%. Additionally, our method also achieved the least average number of additional CNOT gates among available related work, achieving an average additional number of gates of 6651.2 ± 5330.34. This implies that our RL agent is scalable and can perform well on large number of qubits with any depth given its fixed-length feature vector unlike the variable-length feature vector by Huang et al. [13] that increases proportionally as the number of two-qubit gates increases. Also, Huang et al. [13] employ a transformer neural network which is computationally expensive and unscalable due to its inability to handle long sequences.

We further compare the time required for a given circuit between a heuristic approach [9] and our RL approach with and without considering routing time, shown in Table 4, for the 2-D grid shown in Fig. 2. The heuristic time approach is comparable to our RL approach without considering routing time, which we aim to improve by employing a neural network as future work. Cheng et al. [9] approach first constructs the adjacency matrix by iterating over all two-qubit gates, represented by G. This is then utilized by the nearest neighbor algorithm to place qubits based on available unoccupied neighbors in addition to minimizing the interaction cost with placed qubits resulting in a cost of $O(n^3)$, where n is the number of qubits. Thus, Cheng et al. [9] approach, given circuit encoding and nearest neighbor algorithm, has a complexity of $O(n^3 + G)$.

Our RL approach has fixed steps per episode, totalling $n - 1$, where n is the number of qubits. At the first step in the episode, we encode the circuit by iterating over all gates resulting in a cost of $O(G)$, where G is the total number of two-qubit gates. For the remaining

Table 4 Time comparison in seconds between heuristics approach and RL agent inference per circuit

Benchmark	Qubits #	Logical Depth #	Logical CNOTs#	Time Per Circuit in Seconds		
				Cheng et al. [9]	RL Without Routing	RL With Routing
clip_206	14	12,028	14,772	6.15	5.56	53.31
cm85a_209	14	4256	4986	1.94	1.96	20.05
cycle10_2_110	12	2276	2648	1.08	1.11	11.62
dist_223	13	13,274	16,624	6.77	6.61	56.55
hwb6_56	7	2559	2952	1.32	1.85	10.9
hwb7_59	8	9112	10,681	4.4	4.77	37.04
hwb8_113	9	26,041	30,372	12.38	11.21	108.66
mlp4_245	16	6930	8232	4.02	3.33	34.02
radd_250	13	1210	1405	0.64	0.55	5.9
rd73_252	10	1963	2319	1	0.77	8.4
rd84_253	12	4917	5960	2.69	2.46	22.09
root_255	13	5965	7493	3.02	3.33	26.39
sao2_257	14	13,209	16,864	7.6	6.24	56.19
sym10_262	12	23,736	28,084	12.9	11.59	102.45
sym9_148	10	8062	9408	4.02	3.66	28.84
sym9_193	11	12,849	15,232	6.11	5.63	50.17
Urf1_278	9	22,307	26,692	10.5	10.1	96.32
Urf2_277	8	8312	10,066	3.85	4.18	39.31
Urf5_280	9	19,888	23,764	9.25	9.43	94.7

steps in the episode, we update the state based on the predicted action by only swapping two qubits in the encoding matrix resulting in a cost of $O(4n)$. Consequently, the total cost for our approach without routing is $O(n^2 + G)$ per episode. For the routing evaluation, we route the circuit at each step in the episode using the default routing algorithm by T|ket⟩ compiler [40], considerably faster than other compilers on large benchmarks, resulting in a cost of $O(n \cdot G)$ per episode. Thus, our RL approach along with the routing time evaluation is still considered scalable having a computational complexity of $O(n^2 + n \cdot G)$), where n is the number of qubits, G is the total number of gates.

4.2 Scalability

To evaluate the scalability of our model, we compare the number of parameters required by the reinforcement learning model between 9-qubit and 20-qubit hardware. As in the 20-qubit dataset, our training dataset consists of a combination between random and benchmark circuits totalling 53,610. Our testing dataset consists of 6 out of the 19 benchmarks mentioned in Table 3 that have qubits less than or equal 9-qubit. Similar to the coupling map of the 20-qubit hardware in Fig. 2, we utilize a 3 by 3 subset, totalling 9-qubit, from that coupling map. Additionally, we use the same neural network model as in Fig. 6. However, we utilize half the number of neurons in each of the three hidden layers. Specifically, for the shared layer between the actor and the critic the number of neurons is 256. After that each of the actor and critic has two consecutive fully connected layers having 64 and 32 neurons, respectively. Thus, the total number of parameters, excluding the input and the output layers, required for the 9-qubit model is 37,056 parameters, while that of the 20-qubit model is 147,840 parameters. Thus, our neural network model scales linearly as the number of qubits increases.

Table 5 represents the results for the six benchmarks achieving minimum number of additional CNOT gates outperforming the results mentioned in Table 3. This clarifies that

Table 5 Benchmark test circuits for the 9-qubit model

Benchmark	Logical CNOTs#	Δ CNOTs after mapping and routing
		RL
hwb6_56	2952	**819**
hwb7_59	10,681	**3486**
hwb8_113	30,372	**12,294**
Urf1_278	26,692	**13,227**
Urf2_277	10,066	**4056**
Urf5_280	23,764	**12,186**

the 12.82% drop in performance in hwb6_56 benchmark circuit is due to the training dataset of the 20-qubit model having high complexities.

5 Conclusion

In this paper, we utilized Maskable PPO reinforcement learning algorithm to train an agent to find the optimal logical-to-physical mapping of quantum circuits. Specifically, we only rely on the edges feature vector demonstrating its scalablility as it is depth independent. Experimental results show that it is possible to train a generic reinforcement learning agent coupled with simple routing evaluation capable of selecting a near-optimal mapping outperforming the available heuristics and related reinforcement learning approach on 12 out of 19 test benchmarks, achieving a geometric mean improvement over the best available related work with 2.2% and an average additional CNOT gates of 6651.2 ± 5330.34. We also outperformed two heuristics approaches achieving a geometric mean of 15%. In addition, we demonstrated that our neural network model scales linearly as the number of qubits increases. Thus, our results demonstrate the advantage of our approach over available related works in terms of the simple scalable neural network, compact circuit encoding, and fewer average additional CNOT gates. This would radically improve circuit reliability as the reduction of additional CNOT gates would reduce additional noise in addition to decreasing the depth of a given circuit. As future work, we plan to investigate an approach using either neural networks or heuristics to predict additional CNOT counts after routing given the logical circuit in order to speed up the routing evaluation complexity of our approach. Additionally, we plan to investigate multi-agent reinforcement learning to handle multi-objective optimizations such as reduction of additional gates and selecting reliable links based on the backend's noise.

Acknowledgements
Special thanks to Yasutaka Wada for his fruitful discussions regarding this work. This work was supported by JSPS Bilateral Program Number JPJSBP 120226002. This paper is based upon work supported by Science, Technology & Innovation Funding Authority (STDF) under grant number 46263.

Funding
Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB). This work was supported by JSPS Bilateral Program Number JPJSBP 120226002. This paper is based upon work supported by Science, Technology & Innovation Funding Authority (STDF) under grant number 46263.

Data Availability
No datasets were generated or analysed during the current study.

Declarations

Competing interests
The authors declare no competing interests.

Author contributions
AE formulated the problem of utilizing reinforcement learning for selecting a near-optimal initial logical-to-physical mapping. NE performed the data collection and code implementation. WG analyzed and interpreted the results. KK and KU contributed to problem positioning and presentation. All authors read and approved the final manuscript.

Author details
[1]Department of Computer Science and Engineering, Egypt-Japan University for Science and Technology, Alexandria, Egypt. [2]Department of Computer and Systems Engineering, Faculty of Engineering, Alexandria University, Alexandria, Egypt. [3]Waseda University, Tokyo, Japan. [4]School of Information Technology and Computer Science, Nile University, Cairo, Egypt.

Received: 14 December 2023 Accepted: 21 February 2024 Published online: 13 March 2024

References

1. Preskill J. Quantum computing in the NISQ era and beyond. Quantum. 2018;2:79.
2. Castelvecchi D. Ibm releases first-ever 1000-qubit quantum chip. Nature.
3. Bravyi S, Cross AW, Gambetta JM, Maslov D, Rall P, Yoder TJ. High-threshold and low-overhead fault-tolerant quantum memory. 2023. arXiv preprint arXiv:2308.07915.
4. Fösel T, Niu MY, Marquardt F, Li L. Quantum circuit optimization with deep reinforcement learning. 2021. arXiv preprint arXiv:2103.07585.
5. Siraichi MY, Santos VFD, Collange C, Pereira FMQ. Qubit allocation. In: Proceedings of the 2018 international symposium on code generation and optimization. 2018. p. 113–25.
6. Zulehner A, Paler A, Wille R. An efficient methodology for mapping quantum circuits to the ibm qx architectures. IEEE Trans Comput-Aided Des Integr Circuits Syst. 2018;38(7):1226–36.
7. Li G, Ding Y, Xie Y. 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. 2019. p. 1001–14.
8. Paler A. On the influence of initial qubit placement during nisq circuit compilation. In: International workshop on quantum technology and optimization problems. Berlin: Springer; 2019. p. 207–17.
9. Cheng X, Guan Z, Zhu P. Nearest neighbor transformation of quantum circuits in 2d architecture. IEEE Access. 2020;8:222466–75.
10. De Brugiere TG, Baboulin M, Valiron B, Martiel S, Allouche C. Reducing the depth of linear reversible quantum circuits. IEEE Trans Quantum Eng. 2021;2:1–22.
11. Fu H, Zhu M, Wu J, Xie W, Su Z, Li X-Y. Effective and efficient qubit mapper. In: 2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD). Los Alamitos: IEEE; 2023. p. 1–9.
12. Jang W, Terashi K, Saito M, Bauer CW, Nachman B, Iiyama Y, Kishimoto T, Okubo R, Sawada R, Tanaka J. Quantum gate pattern recognition and circuit optimization for scientific applications. In: EPJ Web of Conferences. vol. 251. EDP Sciences; 2021. p. 03023.
13. Huang C-Y, Lien C-H, Mak W-K. Reinforcement learning and dear framework for solving the qubit mapping problem. In: Proceedings of the 41st IEEE/ACM international conference on computer-aided design. 2022. p. 1–9.
14. Ren S, Chen K, Ghadermarzy N, Nguyen B, Huang Y, Ronagh P. Nuwa: a quantum circuit transpiler based on a finite-horizon heuristic for placement and routing. 2021. arXiv preprint arXiv:2110.00592.
15. Zhu P, Feng S, Guan Z, et al. A variation-aware quantum circuit mapping approach based on multi-agent cooperation. 2021. arXiv e-prints, 2111.
16. Sinha A, Azad U, Singh H. Qubit routing using graph neural network aided Monte Carlo tree search. In: Proceedings of the AAAI conference on artificial intelligence. vol. 36. 2022. p. 9935–43.
17. Pozzi MG, Herbert SJ, Sengupta A, Mullins RD. Using reinforcement learning to perform qubit routing in quantum compilers. ACM Trans Quantum Comput. 2022;3(2):1–25.
18. Peters J, Vijayakumar S, Schaal S. Reinforcement learning for humanoid robotics. In: Proceedings of the third IEEE-RAS international conference on humanoid robots. 2003. p. 1–20.
19. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M. Playing atari with deep reinforcement learning. 2013. arXiv preprint arXiv:1312.5602.
20. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M et al. Mastering the game of go with deep neural networks and tree search. Nature. 2016;529(7587):484–9.
21. Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, Lanctot M, Sifre L, Kumaran D, Graepel T, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. 2017. arXiv preprint arXiv:1712.01815.
22. Wilson E, Singh S, Mueller F. Just-in-time quantum circuit transpilation reduces noise. In: 2020 IEEE International Conference on Quantum Computing and Engineering (QCE). Los Alamitos: IEEE; 2020. p. 345–55.
23. Lao L, Korotkov A, Jiang Z, Mruczkiewicz W, O'Brien TE, Browne DE. Software mitigation of coherent two-qubit gate errors. Quantum Sci Technol. 2022;7(2):025021.
24. Anis MS et al. Qiskit: an open-source framework for quantum computing. 2021. https://doi.org/10.5281/zenodo.2573505.
25. Zhu P, Guan Z, Cheng X. A dynamic look-ahead heuristic for the qubit mapping problem of nisq computers. IEEE Trans Comput-Aided Des Integr Circuits Syst. 2020;39(12):4721–35.
26. Garey MR. Computers and intractability: a guide to the theory of np-completeness. New York: Freeman; Fundamental 1997.
27. IBM Quantum Processor Types. https://docs.quantum-computing.ibm.com/run/processor-types. Accessed: 2023-11-20.
28. Murali P, Baker JM, Javadi-Abhari A, Chong FT, Martonosi M. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In: Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems. 2019. p. 1015–29.

29. Niu S, Suau A, Staffelbach G, Todri-Sanial A. A hardware-aware heuristic for the qubit mapping problem in the nisq era. IEEE Trans Quantum Eng. 2020;1:1–14.
30. Tannu SS, Qureshi MK. Not all qubits are created equal: a case for variability-aware policies for nisq-era quantum computers. In: Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems. 2019. p. 987–99.
31. Dury B, Di Matteo O. A qubo formulation for qubit allocation. 2020. arXiv preprint arXiv:2009.00140.
32. Li A, Stein S, Krishnamoorthy S, Ang J. Qasmbench: a low-level quantum benchmark suite for nisq evaluation and simulation. ACM Trans Quantum Comput. 2023;4(2):1–26.
33. Kepler J. University Linz institute for integrated circuits. In: IIC JKU – IBMQX QASM circuits. 2019. https://github.com/iic-jku/ibm_qx_mapping/tree/master/examples. Last accessed: 14 December 2023.
34. Quetschlich N, Burgholzer L, Wille R. Mqt bench: benchmarking software and design automation tools for quantum computing. Quantum. 2023;7:1062.
35. Huang S, Ontañón S. A closer look at invalid action masking in policy gradient algorithms. 2020. arXiv e-prints, 2006.
36. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. 2017. arXiv preprint arXiv:1707.06347.
37. Ding Y, Chong FT. Quantum computer systems research for noisy intermediate-scale quantum computers.
38. Raffin A, Hill A, Gleave A, Kanervisto A, Ernestus M, Dormann N. Stable-baselines3: reliable reinforcement learning implementations. J Mach Learn Res. 2021;22(268):1–8.
39. Sivarajah S, Dilkes S, Cowtan A, Simmons W, Edgington A, Duncan R. tket: a retargetable compiler for nisq devices. Quantum Sci Technol. 2020;6(1):014003.
40. Cowtan A, Dilkes S, Duncan R, Krajenbrink A, Simmons W, Sivarajah S. On the qubit routing problem. 2019. arXiv preprint arXiv:1902.08091.

Publisher's Note