**DOI**:10.3969/j.issn.1671-0673. 2025. 04. 013

# 分布式强化学习驱动的量子编译自动调优方法

刘 毅, 朱 雨, 许瑾晨, 杜启明, 连 航, 涂 政
(信息工程大学,河南 郑州 450001)

**摘要**:针对强化学习模型应用到量子编译自动调优领域开销大的问题,提出一种分布式强化学习(DRL)驱动的量子编译自动调优方法,通过将经验生成与智能体训练解耦,基于分布式集群实现了并行经验生成。该方法通过建立具有稠密奖励特性的量子编译马尔可夫决策过程(MDP)模型,设计经验生成与智能体训练的解耦机制,结合动态经验加载策略,在保证优化效果的同时提升训练效率。实验结果表明,分布式训练框架训练耗时减少 54.6%;优化性能方面,智能体在测试集77.3%的量子线路上表现优于Qiskit-O3编译器,对未见过的Shor算法线路平均减少 17.4%量子门数量。

**关键词**:强化学习;分布式系统;深度Q网络;量子编译优化;量子编译自动调优
**中图分类号**:TP311.5 **文献标识码**:A **文章编号**:1671-0673(2025)04-0462-08

## A Distributed Reinforcement Learning Driven Approach to Automatic Tuning of Quantum Compilation

LIU Yi, ZHU Yu, XU Jinchen, DU Qiming, LIAN Hang, TU Zheng
(*Information Engineering University*, *Zhengzhou* 450001, *China*)

**Abstract:** Aiming at the problem of high overhead when applying reinforcement learning models to the field of automatic tuning of quantum compilation, a distributed reinforcement learning (DRL) driven automatic tuning method for quantum compilation is proposed. By decoupling experience generation from agent training, parallel experience generation is achieved based on a distributed cluster. In this method, a markov decision process (MDP) model of quantum compilation is established with the characteristic of dense rewards, a decoupling mechanism for experience generation and agent trainingis is designed, and a dynamic experience loading strategy is combined to improve the training efficiency while ensuring the optimization effect. Experiments demonstrate a 54.6% reduction in training time compared to baseline methods. In terms of optimization performance, the agent performs better than the Qiskit-O3 compiler on 77.3% of the quantum circuits in the test set, and the number of quantum gates of the unseen Shor algorithm circuits is reduced by an average of 17.4%.

**Key words:** reinforcement learning; distributed system; deep Q-network; quantum compilation optimization; auto-tuning quantum compilation

量子计算作为一种新兴的计算范式,凭借其独特的并行计算能力,在解决复杂计算问题方面展现出巨大潜力。近期,随着量子计算机的飞速落地,量子计算进入噪声中规模(Noisy Intermediate Scale Quantum, NISQ)阶段。量子编译器作为连接量子算法与量子硬件的关键工具,其优化性能直接影响量子计算的效率和可靠性。在量子编译过程中,优化技术的执行顺序对最终编译结果

**收稿日期**:2025-03-11;**修回日期**:2025-04-11
**作者简介**:刘 毅(1998—),男,硕士生,主要研究方向为量子计算。E-mail:yd14737156215@163.com

具有较大影响。然而，现有量子编译器，如Qiskit[1]、Tket[2]等，普遍采用固定优化策略或依赖经验配置，难以适应不同量子程序的动态优化需求。

在经典编译领域，基于强化学习（Reinforcement Learning, RL）[3]的编译优化方法已取得显著进展[4]。这类方法通过建立编译过程与马尔可夫决策过程（Markov Decision Process, MDP）[5]的映射关系，利用智能体自主学习优化策略。受此启发，Quetschlich 等[6]在量子编译领域，利用近端策略优化[7]（Proximal Policy Optimization, PPO）算法，实现了跨编译器优化序列的自动生成。然而，该方法面临训练效率的严峻挑战：Quetschlich 等[8]的另一项实验研究表明单次50量子比特编译线路需要上百秒，而RL训练过程需上万次线路编译迭代，总训练时间达到上千小时。

针对上述问题，本文提出一种分布式强化学习（Distributed Reinforcement Learning, DRL）驱动的量子编译自动调优方法，旨在解决强化学习模型应用于量子编译自动调优领域开销大的问题。以深度Q网络（Deep Q-Network, DQN）[9]算法作为基础算法，通过建立具有稠密奖励特性的量子编译马尔可夫决策过程模型，设计经验生成与智能体训练的解耦机制，结合动态经验加载策略，在保证优化效果的同时提升训练效率。需要指出的是，当前存在如MQTPredictor等的相关工作，旨在融合多个量子编译器的全阶段调优，本文方法相比其则侧重于通过自动调优技术，来增强特定编译器优化阶段的优化能力，同时还借助分布式强化学习来加速训练速度，二者研究方向存在差异，难以直接进行对比实验。但本文方法在提升训练效率与优化性能方面展现出独特优势，后续通过相关实验验证了方法的有效性与创新性，为量子编译优化提供了新思路与技术途径。

# 1　基础知识

## 1.1　强化学习与马尔可夫决策过程

RL是机器学习领域的重要范式，其核心思想是通过智能体与环境的持续交互来学习最优策略。作为人工智能领域的研究热点，RL在多个领域（如机器人控制、计算机网络等）展现了强大的潜力[10-12]。智能体在时刻$t$观测环境状态$s_t \in S$，根据当前策略选择动作$a_t \in A$，执行动作后获得即时奖励$r_t \in R$并转移到新状态$s'_{t+1}$。该过程可形式化为MDP四元组$(S, A, S', R)$，其中：$S$为状态空间；$A$为动作空间；$S'$为状态转换后的状态；$R: S \times A \to R$为奖励函数。

MDP的核心特性是“马尔可夫性”，即未来状态仅依赖于当前状态和动作，而与历史状态无关。这一特性使得RL能够通过迭代优化策略，逐步逼近最优解。RL的目标是最大化累积奖励，智能体通过试错和策略调整，最终找到最优行为模式。

## 1.2　深度Q网络及其改进

Q-learning[13]是经典的基于动作价值函数（也称为Q函数）的RL算法，通过迭代更新动作价值函数$Q(s, a)$逼近最优策略。Q函数（动作价值函数）表示在状态$s$下执行动作$a$后，智能体在未来可能获得的累积奖励的期望值，其数学表达式如式（1）所示：

$$Q(s,a) = E\left[ R_t + \gamma \max_{a'} Q(s',a') \middle| s_t = s, a_t = a \right] \quad (1)$$

式中：$\gamma$是值折扣因子，用于权衡即时奖励和未来奖励的重要性；$R_t$是在时间步$t$获得的奖励。

DQN结合深度神经网络与经验回放机制，解决了传统Q-learning在高维状态空间下的维度灾难问题。其核心创新包括：1）经验回放：将交互经验$(S, A', S, R)$存储在回放缓冲区，训练时随机采样打破数据相关性；2）目标网络：使用独立的目标网络计算TD目标，缓解自举法带来的偏差积累。

Double DQN[14]通过解耦动作选择与价值评估，解决$Q$值过高估计问题。其目标$Q$值计算改进如式（2）所示：

$$y_t = r_t + \gamma Q'\left( s_{t+1}, \arg\max_a Q(s_{t+1}, a; w); w^- \right) \quad (2)$$

式中，$w$和$w^-$分别为策略网络$Q$和目标网络$Q'$参数。

Dueling DQN[15]创新性地将Q网络分解为状态价值函数$V(s)$和优势函数$A(s,a)$，其$Q$值计算公式如式（3）所示：

$$Q(s,a; w) = V(s; w) + A(s,a; w) - \frac{1}{|A|} \sum_{a'} A(s,a'; w) \quad (3)$$

该结构使网络能独立学习状态价值和动作优势，在量子编译优化中能有效区分离散优化动作的影响。

## 1.3　$\varepsilon$-greedy探索策略

RL技术领域中，探索—利用困境（Exploration-Exploitation Dilemma）是核心挑战之一。$\varepsilon$-greedy策略通过概率平衡随机探索与策略利用如式（4）：

$$a_t = \begin{cases} \arg\max_a Q(s_t, a), & \text{概率}(1-\varepsilon); \\ \text{随机动作}, & \text{概率}\varepsilon. \end{cases} \quad (4)$$

动态衰减的$\varepsilon$值常被采用，训练初期设置较高探索率（如$\varepsilon$=1.0），随训练进程线性衰减至下限（如$\varepsilon$=0.1），确保智能体从充分探索平稳过渡到策略利用。

### 1.4 量子编译自动调优

量子编译流程可概括为门综合、拓扑映射与动态优化3阶段协同作用。在门综合阶段，高阶量子门（如多比特控制门、复杂酉操作）被转换为目标硬件原生支持的基础门组，这一过程依赖数学分解理论（如SU(2)分解、门替换规则）实现量子操作的硬件兼容性表达，本质上是面向门集约束的技术适配选择。拓扑映射阶段则聚焦量子位的物理实现，基于芯片的耦合结构（如链式、网格拓扑）将逻辑量子位分配至物理位置，并通过路由算法（如SWAP门插入）解决双门操作的邻接性约束，属于硬件拓扑约束下的资源分配技术决策。

在优化阶段，量子编译器的性能高度不仅依赖于优化技术的强弱，还依赖于优化技术的执行顺序和组合。不同的量子线路可能对优化技术的响应不同，因此固定优化策略难以适应所有量子程序的动态需求。自动调优技术通过自适应调整优化技术的执行策略，使得量子编译器能够根据量子线路的特征动态选择最优的优化序列。

## 2 分布式强化学习驱动的量子编译自动调优方法设计

首先介绍基于DQN的量子编译自动调优方法设计，包括：量子编译的MDP建模、经验生成与智能体训练解耦以及智能体训练方式。

### 2.1 基于DQN的量子编译自动调优方法

图1展示了智能体采用DQN算法的训练流程。智能体训练采用$\varepsilon$-greedy策略生成其训练所需的经验（即训练数据），所有经验由经验回放池维护，经验回放池实际为一个具有容量上限的队列。经验生成与智能体训练存在强耦合关系，经验会随着智能体训练不断生成并保存至经验回放池中，超出经验回放池的经验样本会被舍弃。在训练收敛后，智能体可自主指导量子编译器选择最优化的编译技术序列。

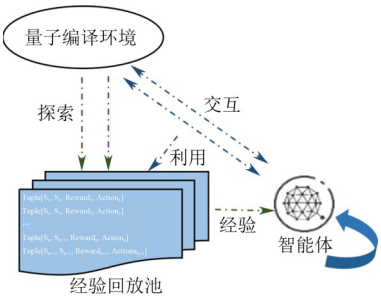



图1 基于DQN算法的智能体训练流程

### 2.2 量子编译的马尔科夫决策过程建模

量子编译序列优化问题天然符合MDP框架，其中状态空间由量子线路特征构成，动作空间对应编译器可执行的优化操作，如门替换、门重排等，奖励则表示执行某个动作后量子线路性能的提升程度。其中，量子线路会被转化为有向无环图（Driected Acyclic Graph, DAG），进而将量子线路状态表示为图数据。如图2所示，量子门被编码为图节点，量子门名称以One-Hot编码方式编码为节点属性。

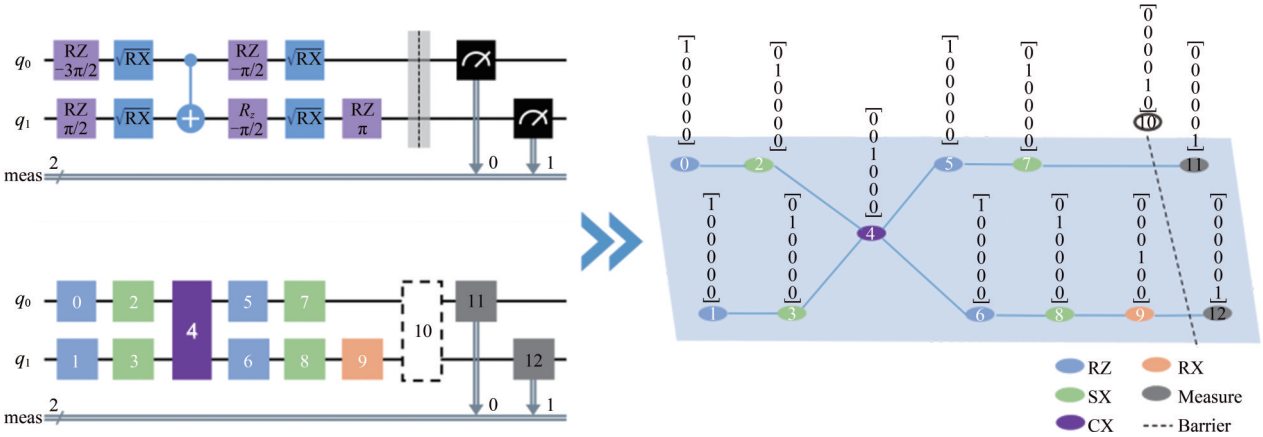



图2 线路状态编码

具体来说，图2中有6种不同的量子门类型，分别编为0到5。每个量子门类型对应一个长度为6的二进制向量，其中只有对应的位置为1，其余位置为0。例如：RZ门对应编号0，其One-Hot编码为[1, 0, 0, 0, 0, 0]；SX门对应编号1，其One-Hot编码为[0, 1, 0, 0, 0, 0]；CX门对应编号2，其One-Hot编

码为[0, 0, 1, 0, 0, 0]等。

量子门之间的依赖关系表示为图中的有向边。若一个量子门操作必须在另一个量子门操作之前执行，则在这两个节点之间添加一条有向边，指示它们的时序关系，例如节点0后为节点2，则两个节点的边可编码为[0, 2]。

算法1描述了量子编译优化过程建模为MDP后，量子编译器优化一条量子线路的过程。在编译过程中，量子编译环境会根据当前的量子线路状态生成一系列动作，并根据动作价值函数对这些动作进行排序，然后选择满足硬件约束条件的最优动作执行。该过程中生成的每一条经验中，每一个动作都能得到有效的奖励值，所以该MDP的奖励值是稠密的。其中，双重硬件约束验证包括门集约束验证和拓扑约束验证两个方面。门集约束验证主要检查量子线路中的所有量子门是否属于目标硬件支持的门集。若存在不支持的门，则验证失败。拓扑约束验证则是检查量子线路中的双量子比特门（如CNOT）是否满足硬件的量子比特拓扑连接结构。若存在不满足拓扑连接的门，则验证失败。

算法1　基于MDP的单次编译优化流程

输入：初始线路 $C_0$，基础门集 $G$，后端拓扑 $M$，最大迭代次数 $T$

输出：优化后的线路 $C^*$

/*环境初始化*/
初始化：$C \leftarrow C_0, t \leftarrow 0$
while $t < T$ do
　/*状态表征：提取线路特征向量*/
　$s_t \leftarrow \text{ExtractFeatures}(C)$
　/*动作空间生成与Q值排序*/
　$A \leftarrow \text{GenerateActions}(s_t)$
　$A_{\text{sorted}} \leftarrow \text{SortActionsByQValue}(s_t, A)$
　/*约束感知动作选择*/
　for $a \in A_{\text{sorted}}$ do
　　$C_{\text{candidate}} \leftarrow \text{ApplyAction}(C, a)$
　　if $C == C_{\text{candidate}}$ then　/*提前终止*/
　　break
　　/*双重硬件约束验证*/
　　if CheckGates ($C_{\text{candidate}}$, $G$)
　　and CheckCoupling ($C_{\text{candidate}}$, $M$)
　　/*接受合法动作*/
　　then
　　$r_t \leftarrow \text{Reward}(C, C_{\text{candidate}})$
　　$C \leftarrow C_{\text{candidate}}$
　　$t \leftarrow t + 1$
　　end if
　end for
end while
return $C^* \leftarrow C$

## 2.3　并行经验生成及训练

RL训练过程由智能体与环境交互过程中生成经验和智能体训练两部分组成。由于传统DQN通常采用 $\varepsilon$-greedy 策略，导致经验生成与智能体训练存在强耦合，使得整个训练过程的并行加速难以实现。因此，本方法将经验生成与智能体训练解耦，从而支持RL训练过程的并行。

### 2.3.1　并行生成经验

图3展示了经验生成与智能体训练的解耦过程。本方法采用分布式并行方式生成经验集作为总训练数据，其核心在于去除训练阶段智能体与量子编译环境交互步骤，将智能体训练拆分为训练数据生成和智能体迭代训练两个独立的部分。多个计算节点（Worker）同时运作，每个节点运行一个智能体，在不同的编译环境副本中交互，记录状态 $S$、下一步状态 $S'$、动作 $A$、奖励 $R$，形成四元组作为一条经验样本 $(S, S', A, R)$。形式上，假设有 $M$ 个计算节点 $\{W_1, W_2, \cdots, W_M\}$。若编译环境用函数 $E$ 表示，则智能体与编译环境交互可表示为 $E(S, A) = (S', R)$。

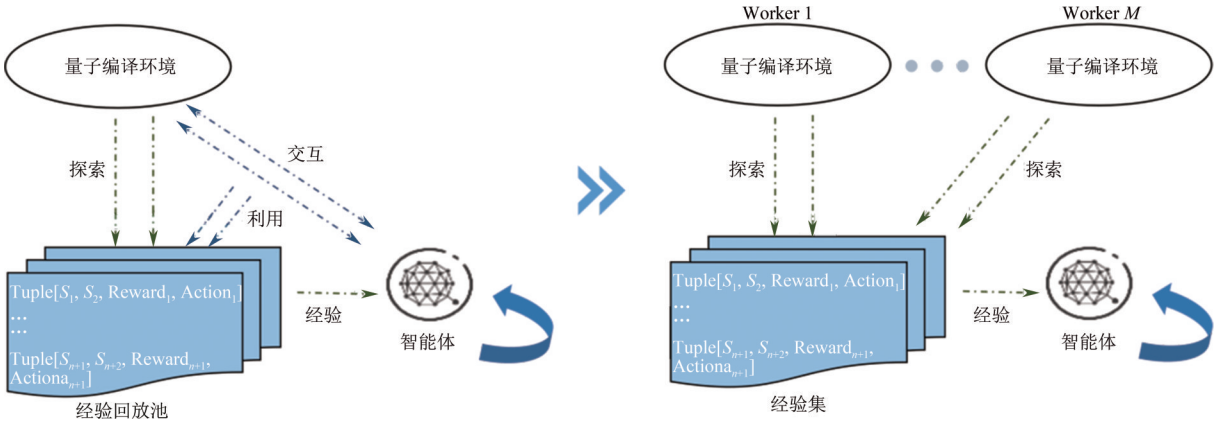



图3　经验生成与智能体训练解耦示意图

各计算节点上的智能体独立与量子编译环境交互，依据策略选动作，环境根据动作编译量子线路并返回新状态和奖励，智能体记录形成经验样本。所有节点的经验样本合并得到整个训练所需的经验集 $D = \bigcup_{i=1}^{M} D_i$。

2.3.2 经验集动态载入

图4展示了经验集切片及动态加载方法。由于经验集中的数据是智能体训练的所有经验，占用大量存储空间，所以将经验集一次性加载到内存中无论在研究过程中还是应用中都是不可行的。并且一次性加载到内存中的经验只有其中一小部分会被智能体训练所利用，其数据利用率不高。因此，采用动态加载经验的方法，避免了不必要的内存开销，从而提升内存资源利用效率。

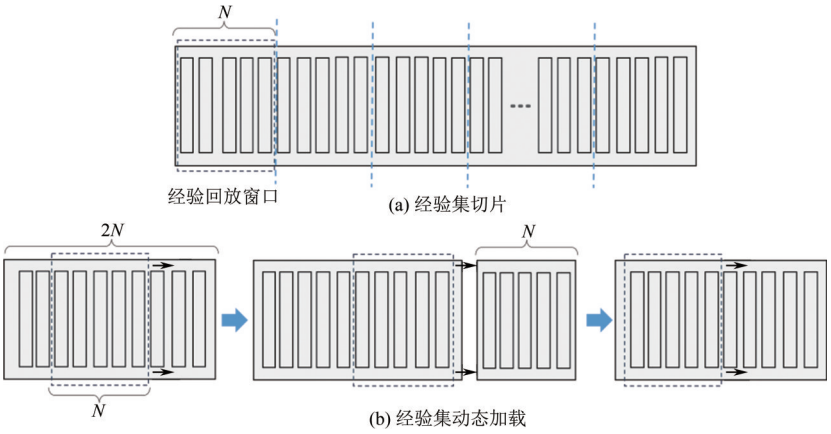



**图4 经验集切片及动态加载方法**

1）经验集切片处理。并行生成经验之后，首先对经验集进行切片并存储在外存中。切片大小与经验回放窗口相同（长度为 $N$），如图4(a)所示。这种切片处理方式将庞大的经验集分割成多个较小的片段，便于后续的管理和加载。

2）经验集动态加载机制。图4(b)为经验集动态加载概图。智能体加载 $2N$ 条经验样本，并在每次网络迭代时滑动窗口一次，滑动速度默认为1。经过 $N$ 次迭代后，内存中舍弃前 $N$ 个经验数据并加载下一个经验切片，将当前窗口（包含（$N$+1）~$2N$ 的经验）与新加载的切片拼接。

2.3.3 智能体训练

对于具有稠密奖励的MDP，DQN技术与基于Actor-Critic架构的PPO等算法相比，具有显著优势。DQN将每一步优化视为独立的训练样本，而PPO则将编译一次量子线路的所有优化步骤整合为一条训练样本。因此，DQN在数据利用率方面表现更为高效，使其在稠密奖励场景下更具适用性。为了进一步提升DQN的性能，本方法引入了D3QN（Dueling Double DQN）技术。D3QN结合了Dueling DQN和Double DQN的优势，通过将动作价值函数分解为状态价值函数和优势函数，能够更精确地评估动作的价值，同时有效缓解了Q值过高估计的问题，从而在稠密奖励环境中表现出更好的稳定性和收敛性。

传统DRL常依赖优先级经验回放[16]（Prioritized Experience Replay）技术筛选高价值样本，但本方法基于稠密奖励环境特性重新设计了训练范式：由于量子编译过程每个优化步骤均能获得细粒度奖励信号（如门数量减少、线路深度降低等），智能体无需复杂优先级计算即可高效学习。受益于稠密奖励的RL环境，智能体学习在保证学习效果的情况下，同样也减少优先级计算步骤，进一步降低训练耗时。在本方法中，只需按照算法2的流程对智能体进行训练。

算法2 Double DQN算法

输入：经验回放窗口 $D$，策略网络参数 $w$，目标网络参数 $w^-$，折扣因子 $\gamma$，学习率 $\alpha$

输出：更新后的策略网络（Policy Net）参数 $w$，目标网络（Target Net）参数 $w^-$

1. $Batch \leftarrow \mathrm{RandomSample}(D, N)$
2. /*初始化目标 $Q$ 值容器*/
3. $Q_{\mathrm{target}} \leftarrow \mathrm{EmptyArray}(N)$
4. /*前向传播计算 $Q$ 值*/
5. for $i$ = 1 to $N$ do
6. $s_i, a_i, r_i, s_i' \leftarrow Batch[i]$
7. /*策略网络计算当前 $Q$ 值—$q_i$*/
8. $q_i \leftarrow$ Policy Net($s_i$, $a_i$; $w$) #
9. /*目标网络计算未来 $Q$ 值—$q_i'$*/
10. $q_i' \leftarrow \mathrm{TargetNet}(s_i', \mathrm{argmax}\ \mathrm{PolicyNet}(s_i'; w); w^-)$
11. /*Bellman方程更新*/

12. $Q_{\text{target}}[i] \leftarrow r_i + \gamma * q'_i$
13. end for
14. /*计算时序差分误差*/
15. $Loss \leftarrow \text{MeanSquaredError}(q_i, Q_{\text{target}})$
16. $w \leftarrow \text{Optimizer}(w, \nabla Loss; \alpha)$
17. /*延迟更新目标网络*/
18. if $t$ mod $K$ == 0 then # 每$K$步同步参数
19. $w^- \leftarrow w$
20. end if
21. return $w, w^-$

## 3　实验结果及分析

### 3.1　实验环境及参数设置

本实验在配备两块 RTX 3090 24 GB GPU 和 AMD TR 3960X CPU 的服务器上进行，系统内存为 128 GB，操作系统为 Ubuntu 20.04.6。实验中使用了 CUDA 12.6 作为 GPU 加速框架，Qiskit 版本为 1.3.2，Pytorch 版本为 2.6.1，Python 版本为 3.10.14。

训练参数方面，智能体的训练步数为 300 000 步，采用经验回放机制，经验回放窗口容量为 10 000。训练过程中使用了 $\varepsilon$-greedy 策略生成经验，初始探索率 $\varepsilon$ 为 1.0。每次训练的批量大小（Batch）为 64，学习率为 $1\times10^{-5}$，折扣因子 $\gamma$ 为 0.90，目标网络更新频率为 500 步。智能体网络架构采用图注意力网络（GAT Network[17]），并行生成经验的核数为 30。

此外，D3QN 算法使用 $\varepsilon$-greedy 策略生成经验，$\varepsilon$ 值会随着训练的进行逐渐减小，以平衡探索与利用。初始 $\varepsilon$ 值为 1.0，随着训练步数的增加，$\varepsilon$ 值逐渐衰减至 0.10，衰减率为 0.999 987。这种动态调整的探索策略有助于智能体在训练初期充分探索环境，而在训练后期逐渐稳定策略，提高优化效果。

关于奖励函数的设置，本实验将其设为量子线路门个数优化比，计算方式如式（5）所示：

$$\text{Reward}\left(S_t,S_{t+1}\right)=\frac{G\left(S_t\right)-G\left(S_{t+1}\right)}{G\left(S_t\right)} \tag{5}$$

式中，$G\left(S_t\right)$ 表示当前 $t$ 时刻量子线路的门个数。

### 3.2　实验设计

实验旨在验证 DRL 训练智能体在加速训练的同时保证效果，设计了下面 3 组实验：收敛性验证、训练时间对比和优化能力测试，并使用 MQT-Bench[18]基准测试集中的量子线路进行验证。

收敛性实验：在 30 万步训练中，每 100 步保存智能体模型参数（共 3 000 个）。训练后，用所有模型编译训练集线路，记录奖励值总和作为优化能力指标，绘制奖励值随步数变化曲线，分析智能体能力演化趋势。

训练时间对比实验：在相同硬件下，对比 DRL 与 D3QN 完成训练目标的时间，分析 DRL 的效率优势。

优化能力测试实验：以量子线路门个数为指标，评估智能体优化能力，门数越少效果越好。与 Qiskit-O3（采用串行流程，含冗余门删除、KAK 分解等技术）对比，评估动态优化策略有效性。Qiskit-O3 通过预定义串行流程优化线路，而本方法基于 RL 智能体动态选择优化技术顺序。

实验使用的量子线路来自 MQTBench 基准测试集，涵盖 9 种量子算法（2 ~ 30 量子比特）的 296 条线路。随机选择 112 条线路作为训练集，剩余 184 条线路作为测试集。此外，额外选择 Shor 算法线路作为未见过的测试集，用于评估智能体的泛化能力。所有线路均经过门综合与映射预处理，以确保满足门集约束和拓扑约束，保证线路的规范性。

### 3.3　收敛性验证及训练时间结果分析

#### 3.3.1　收敛性验证

图 5 展示了 DRL 和 D3QN 的奖励曲线，横坐标表示训练步数；纵坐标表示智能体编译训练集线路获得的奖励总和，用于衡量智能体的优化能力。奖励曲线的收敛表明智能体继续训练不会显著提升其能力。从图 5 可以看出，DRL 的收敛速度慢于 D3QN，但其奖励曲线逐渐向 D3QN 靠拢。在训练至 30 万步时，DRL 的奖励曲线几乎与 D3QN 重合，最终表现出明显的收敛趋势。

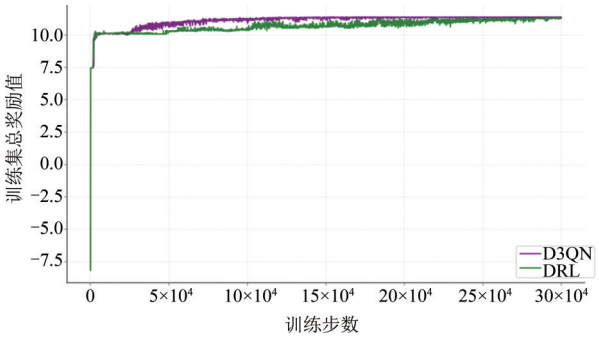



**图 5　DRL 与 D3QN 的奖励曲线**

D3QN 采用 $\varepsilon$-greedy 策略生成经验，随着训练的进行，$\varepsilon$ 逐渐衰减至最小值，导致在训练后期，经验回放池中的经验主要由智能体选择的动作构成，数据趋于稳定。因此，智能体的优化能力不再显著提升，训练过程收敛。而 DRL 采用分布式经验生成框架，多个智能体在不同环境副本中生成

经验，这些经验随机且多样，导致经验回放窗口中的数据不稳定。这种不稳定性使得DRL智能体在训练过程中需要不断适应新数据，从而导致其收敛速度慢于D3QN。

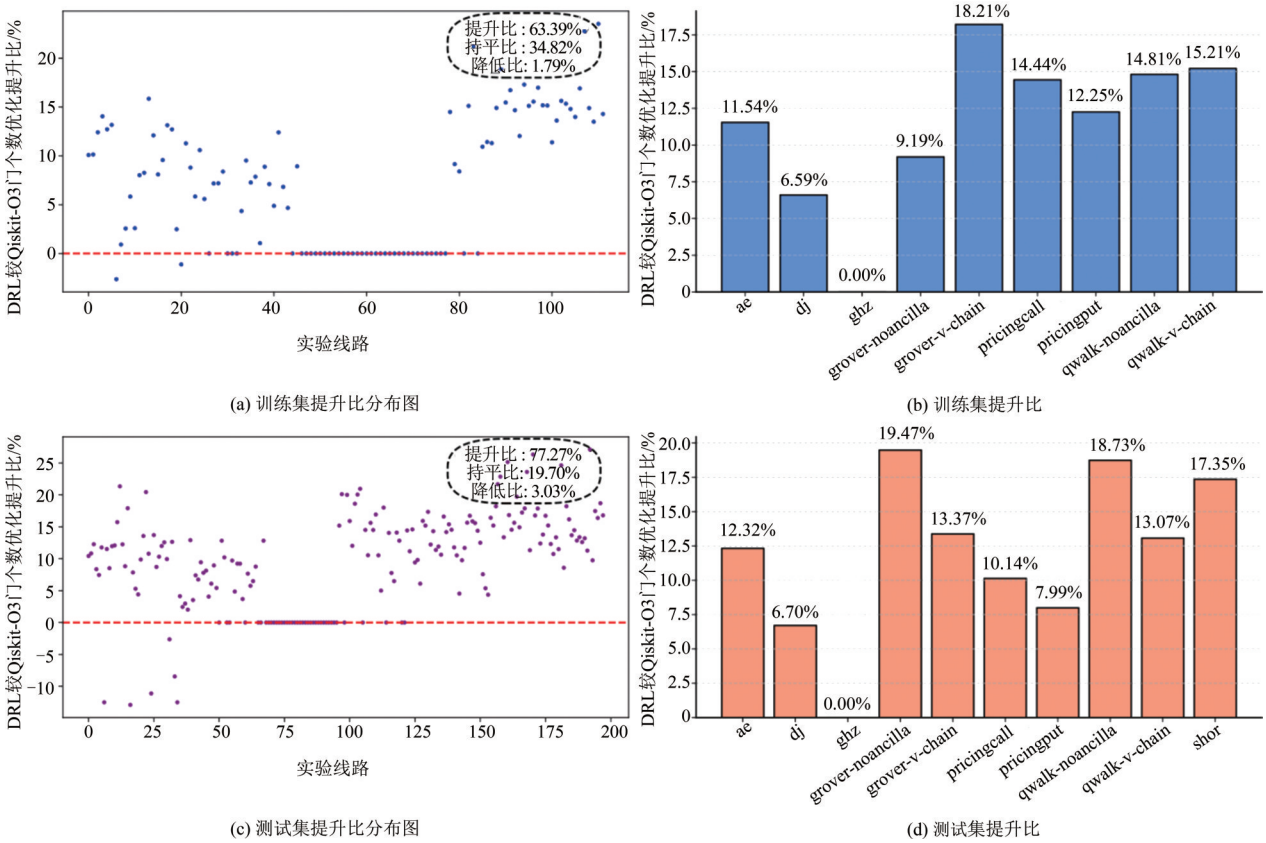
### 3.3.2 训练时间

DRL与D3QN训练30万步需要的时间，其中D3QN随机选择动作生成经验将经验回放池填充满时间 $T_1$ = 0.28 h，智能体采用 $\varepsilon$-greedy 策略训练时间 $T_2$ = 8.3 h；DRL并行生成经验时间 $T_3$ = 0.3 h，智能体训练时间 $T_4$ = 3.6 h。

实验结果表明，$T_3$略大于 $T_1$，这可能是由于并行生成经验集时涉及经验文件的读写操作所致。总体来看，DRL的总训练时间相较于D3QN分布式训练框架训练耗时减少54.6%。尽管DRL的收敛速度较慢，但其训练至30万步所需的时间远低于D3QN，显著提升了训练效率。

### 3.3.3 门个数优化性能

图6展示了采用相同优化技术时，智能体与Qiskit-O3的性能对比结果。其中，图6(a)与6(c)分别呈现了训练集和测试集的性能提升分布，每个数据点的横坐标对应量子线路索引，纵轴反映性能提升比率。数据点位于红色参考线上方时，表明智能体较基准编译器具有更优的优化效果。实验数据显示，在训练集和测试集中分别有63.4%和77.3%的量子线路经过智能体优化后展现出更优性能，仅1.8%和3%的线路优化效果不及Qiskit-O3。GHZ算法在训练集和测试集上提升比均为0，原因是由于Qiskit-O3优化技术不能优化GHZ线路。如图6(d)所示，智能体对于训练过程没有见过的Shor算法线路同样具有很好的表现，平均减少17.4%的量子门。实验数据表明，智能体具有良好的优化能力，并具有一定的泛化能力。




(a) 训练集提升比分布图　(b) 训练集提升比

(c) 测试集提升比分布图　(d) 测试集提升比

**图6　智能体较Qiskit-O3性能提升图**

## 4 结束语

提出了一种基于DRL的量子编译自动调优方法，通过将经验生成与智能体训练解耦，并利用分布式集群实现并行经验生成，提升了训练效率。创新性体现在：1)通过量子线路特征编码与硬件约束验证机制，构建稠密奖励函数空间；2)引入分布式经验生成框架，提升了训练过程的并行化能力；3)提出滑动窗口动态加载策略，缓解大规模经验存储与计算资源的矛盾。实验结果表明，该方法在保证优化效果的同时，训练得到的智能体在量子编译自动调优性能上优于主流编译器，并有

效缓解RL模型在量子编译领域应用时面临的高开销问题。此外,智能体对未见过的量子线路具有良好的泛化能力,进一步验证了该方法的鲁棒性。结果表明,DRL在量子编译优化领域具有广阔的应用前景。然而,本研究仍存在一定的局限性,例如其在不同量子编译架构上的适应性有待进一步探索。未来研究中,一方面,将继续拓展和完善本文方法在量子编译优化领域的应用,探索其对更大规模量子线路的优化效果,进一步提升量子编译效率与质量;另一方面,将深入研究如何使该方法更好地适应不同量子硬件架构,增强其通用性和灵活性。

**参考文献:**


[1] JAVADI-ABHARI A, TREINISH M, KRSULICH K, et al. Quantum computing with Qiskit[DB/OL]. (2024-06-19)[2025-03-09]. https://arxiv.org/abs/2405.08810

[2] SIVARAJAH S, DILKES S, COWTAN A, et al. T|ket: a retargetable compiler for NISQ devices[J]. Quantum Science and Technology, 2021,6(1):No.014003.

[3] WIERING M A, VAN OTTERLO M. Reinforce-ment learning[J]. Adaptation, Learning, and Optimization, 2012,12(3):No.729.

[4] HAJ-ALI A, HUANG Q J, XIANG J, et al. Auto-phase: juggling HLS phase orderings in random forests with deep reinforcement learning[J]. Proceedings of machine learning and systems, 2020,2:70-81.

[5] PUTERMAN M L. Markov decision processes[J]. Handbooks in Operations Research and Management Science, 1990,2:331-434.

[6] QUETSCHLICH N, BURGHOLZER L, WILLE R. MQT predictor: automatic device selection with device-specific circuit compilation for quantum computing[J]. ACM Transactions on Quantum Computing, 2025,6(1):1-26.

[7] SCHULMAN J, WOLSKI F, DHARIWAL P, et al. Proximal policy optimization algorithmst[DB/OL]. (2017-08-28)[2025-03-09].https://arxiv.org/abs/1707.06347.

[8] QUETSCHLICH N, BURGHOLZER L, WILLE R. Reducing the compilation time of quantum circuits using pre-compilation on the gate level[C]//Proceedings of the 2023 IEEE International Conference on Quantum Computing and Engineering. Piscataway, USA: IEEE, 2023:757-767.

[9] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Playing atari with deep reinforcement learning[DB/OL]. (2013-12-19)[2025-03-09]. https://arXiv.org/abs/1312.5602.

[10] 王泉德,王君豪,刘子航.基于动态势能奖励的双足机器人行走控制[J/OL].华中科技大学学报(自然科学版).[2025-04-11].https://doi.org/10.13245/j.hust.250078.

[11] 朱棣,申涓,李丹,等.一种面向网络节能的VNF容量调整方法[J].信息工程大学学报,2023,24(2):183-189.

[12] 陈博,孙鹏浩,兰巨龙,等.基于多智能体强化学习的域间多链路路由优化[J].信息工程大学学报,2022,23(6):641-647.

[13] WATKINS C J C H, DAYAN P. Q-learning[J]. Machine Learning, 1992,8(3):279-292.

[14] VAN HASSELT H, GUEZ A, SILVER D. Deep reinforcement learning with double Q-learning[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2016. DOI:10.1609/aaai.v30i1.10295

[15] WANG Z, SCHAUL T, HESSEL M, et al. Dueling network architectures for deep reinforcement learning[C]// Proceedings of The International Conference on Machine Learning. New York, USA: PMLR, 2016: 1995-2003.

[16] SCHAUL T, QUAN J, ANTONOGLOU I, et al. Prioritized experience replay[J]. Computer Science, 2015. DOI:10.48550/arXiv.1511.05952.

[17] VELIKOVI P, CUCURULL G, CASANOVA A, et al. Graph Attention Networks[J]. 2017. DOI: 10.48550/arXiv.1710.10903.

[18] QUETSCHLICH N, BURGHOLZER L, WILLE R. MQT bench: benchmarking software and design automation tools for quantum computing[J]. Quantum, 2023, 7: No. 1062.


(编辑:冯　春)