模型构建：R-GCN 状态编码器的数学分析

我们在 UNIQ-RL 框架中设计的状态编码器，其核心是一个**关系图卷积网络 (Relational Graph Convolutional Network, R-GCN)**。它的作用是将我们在 Prompt 1.1 中定义的、包含多种类型节点和边的 DOC 状态图，编码成一系列固定维度的、信息丰富的嵌入向量 (Embeddings)。

1. 输入层：特征投影 (Input Feature Projection)

我们的 DOC 状态图包含三种不同类型的节点（门、量子比特、QPU），它们的初始特征向量维度也各不相同。为了在网络中进行统一处理，我们直先需要一个输入层，将这些不同维度的初始特征向量 **x_v** 投影到一个统一的、更高维度的隐藏空间，维度为 d_model。

对于图中的每一个节点 v，其第 0 层的嵌入向量 **h_v^(0)** 计算如下：

h_v^(0) = W_type(v) x_v

其中：

- **x_v** 是我们在上一步中定义的节点 v 的初始特征向量。
- type(v) 代表节点 v 的类型（即 Gate, Qubit, 或 QPU）。
- **W_type(v)** 是一个可学习的投影矩阵。我们为每一种节点类型都设置一个独立的投影矩阵（即 **W_Gate**, **W_Qubit**, **W_QPU**）。这使得网络可以学习到如何为不同类型的实体提取最有用的初始信息。

2. 核心：R-GCN 的信息传播层 (R-GCN Message Passing Layers)

在输入投影之后，我们会堆叠 L 个 R-GCN 层来进行信息传播和特征学习。对于网络中的第 k 层（从 k = 0 到 L - 1），节点 v 的嵌入向量 **h_v^(k+1)** 由其自身以及其所有邻居节点在第 k 层的嵌入向量更新而来。

其核心的消息传递更新法则 (message-passing update rule) 的数学形式如下：

h_v^(k+1) = σ(∑_{r∈R} ∑_{u∈N_r(v)} (1/c_{v,r}) W_r^(k) h_u^(k) + W_0^(k) h_v^(k))

让我们来详细解析这个公式中的每一个部分：

- **h_v^(k)** ∈ ℝ**^d_model** 是节点 v 在第 k 层的嵌入向量。
- σ 是一个非线性激活函数，例如 ReLU(x) = max(0, x)。
- R 是我们 DOC 状态图中所有关系类型的集合。具体来说，R = {depends-on, acts-on, assigned-to, communicates-with}。
- N_r(v) 是在关系 r 下，节点 v 的邻居节点的集合。
- **W_r^(k)** ∈ ℝ**^(d_model×d_model)** 是 R-GCN 的核心。它是一个与特定关系类型 r 相关的、可学习的权重矩阵。这意味着，从"依赖"关系传递的信息，和从"分配"关系传来的信息，会经过完全不同的线性变换。这使得模型能够学习到不同关系的独特语义。
- **W_0^(k)** ∈ ℝ**^(d_model×d_model)** 是一个用于自身环路 (self-loop) 的权重矩阵，负责变换节点 v 自身在上一层的信息。
- c_{v,r} 是一个归一化常数，通常取 |N_r(v)|，用于防止因为节点度数不同而导致的梯度爆炸或消失问题。

具体实现细节（参数共享）：

为了防止在关系类型很多时模型参数过于庞大，R-GCN 通常采用一种名为基分解 (basis decomposition) 的正则化技术。即，将每个 **W_r^(k)** 分解为一组共享的基矩阵 **B_b^(k)** 的线性组合：

W_r^(k) = ∑_{b=1}^B a_{r,b}^(k) B_b^(k)

其中只有系数 a_{r,b}^(k) 是针对每个关系可学习的。这极大地减少了模型的参数，同时增强了模型的泛化能力。

3. 输出层 (Output Layer)

经过 L 层 R-GCN 的信息传播后，我们得到了图中每个节点的最终嵌入向量 **h_v^(L)** ∈ ℝ**^d_model**。

这些最终的嵌入向量，就是我们 R-GCN 状态编码器的输出。它们不再是孤立的、原始的节点特征，而是经过深度加工的、富含了关于其自身属性、领域结构以及在整个 DOC 问题中所扮演角色的高级表示。这些高质量的表示将被直接传递给下一阶段的 Attention 策略网络，作为其决策的依据。

模型构建：节点初始特征向量 (Initial Feature Vectors)

我们将为 DOC 状态图的三类节点——门（Gate）、量子比特（Qubit）和 QPU——分别设计其初始特征向量 **x_v**。

1. 门节点 (Gate Node) 特征向量, x_g

对于每一个尚未被调度的 CNOT 门 g ∈ G'，其初始特征向量旨在回答两个核心问题："这个门是什么？"以及"它现在可以被调度吗？"

Readiness Status (维度: 1):

- **物理意义：** 一个二元特征 {0, 1}，表示该门是否"就绪"。值为 1 意味着它的所有前驱依赖门（根据 precedence relations P）都已经被调度；值为 0 则表示尚未就绪。这是智能体判断一个调度动作是否合法的关键依据。
- **获取方式：** 在每个决策步骤，通过检查调度状态和电路的依赖关系图 (DAG) 动态计算得出。

Normalized Criticality (维度: 1):

- **物理意义：** 一个在 [0, 1] 区间的浮点数，代表该门在完整电路 DAG 中所处的深度，并由总电路深度进行归一化。这个特征为智能体提供了一个关于该门"紧迫性"或"关键性"的启发式信息。深度越大的门通常位于关键路径上，可能需要优先考虑。
- **获取方式：** 在任务开始前，根据完整的 CNOT 门集合 G 和依赖关系 P 一次性预计算得出。

最终门节点特征向量（总维度: 2）: **x_g = [is_ready, normalized_depth]**

2. 量子比特节点 (Qubit Node) 特征向量, x_q

对于每一个逻辑量子比特 q ∈ Q，其初始特征向量旨在回答："这个比特被分配了吗？分配在哪里？它在电路中有多重要？"

Mapping Status (维度: 1):

- **物理意义：** 一个二元特征 {0, 1}。值为 1 表示该量子比特已被分配到一个 QPU；值为 0 表示尚未分配。

- **获取方式：** 在每个决策步骤，从当前的部分分配方案 Π 中直接读取。

QPU Location (维度: |U|):

- **物理意义：** 一个 one-hot 编码的向量，其维度等于 QPU 的总数。向量中值为 1 的位置，指明了该量子比特被分配到了哪一个具体的 QPU。如果尚未分配，则为一个全零向量。
- **获取方式：** 从当前的部分分配方案 Π 中直接读取并编码。

Normalized Interaction Strength (维度: 1):

- **物理意义：** 一个在 [0, 1] 区间的浮点数。该值源自 UNIQ 的 Greedy-JIT 算法，代表该量子比特参与的所有 CNOT 门的总数（即其在"交互图"中的加权度 $W(q) = \sum_{j \in Q} w_{qj}$），并进行归一化。这个特征量化了该比特的"连接性"和"重要性"，是指导分配决策的关键信息。
- **获取方式：** 在任务开始前，根据完整的 CNOT 门集合 G 一次性预计算得出。

最终量子比特节点特征向量（总维度: 2 + |U|）: **x_q = [is_mapped, qpu_id_1, ..., qpu_id_{|U|}, norm_interaction_strength]** ——————————— one-hot location

3. QPU 节点 (QPU Node) 特征向量, x_u

对于每一个物理 QPU $u \in U$，其初始特征向量旨在回答："这个 QPU 在计算和通信方面的资源有多紧张？"

Qubit Load (维度: 1):

- **物理意义：** 一个在 [0, 1] 区间的浮点数，代表该 QPU 的计算量子比特容量的占用率。其值为（已分配到该 QPU 的量子比特数）/ Cap_u。
- **获取方式：** 在每个决策步骤，根据当前分配方案 Π 和静态的 QPU 容量 Cap_u 动态计算得出。

Communication Load (维度: 1):

- **物理意义：** 一个在 [0, 1] 区间的浮点数，代表该 QPU 的通信量子比特在当前时间片的占用率。其值为（当前时间片 t 涉及该 QPU 的已生成或预留的 EPR 对数量）/ E_u。这个特征是判断是否能执行新的远程操作的关键依据，直接关系到 UNIQ 模型中的 EPR 库存 $s_{u,v,t}$ 和通信容量约束 E_u。
- **获取方式：** 在每个决策步骤，根据当前的 EPR 资源状态 R_{res} 和静态的通信量子比特数 E_u 动态计算得出。

最终 QPU 节点特征向量（总维度: 2）: **x_u = [qubit_load, communication_load]**

模型构建：处理混合动作空间的策略网络结构

1. 设计选择：为什么采用统一动作空间？

我们选择"同时评估所有动作"（即统一动作空间）的方案，而不是"分层决策"，主要基于以下两个关键原因：

1. **避免错误的信用分配 (Credit Assignment):** 在分层决策中，如果智能体最终得到了一个坏的奖励，它很难判断是第一层的"类型决策"（例如，选择"分配比特"）错了，还是第二层的"具体决策"（例如，分配了错误的比特）错了。这会严重干扰学习过程。

2. **实现端到端的权衡学习 (End-to-End Trade-off Learning):** DQC 优化的核心难点之一，就是在"继续分配更多比特以获得更全的布局信息"和"立即开始调度已就绪的门以推进度"之间做出权衡。统一动作空间允

许智能体在每一步都直接比较一个最佳分配动作和一个最佳调度动作的价值，从而能够端到端地、更深刻地学习到这种微妙的权衡策略。

2. 具体实现方案：带有类型编码的统一 Attention 网络

我们的方案可以分解为以下三个步骤：

步骤一：构建统一的候选动作集

在每个决策步骤 t，我们首先收集当前状态 s_t 下所有合法的动作，并将它们放入一个统一的集合 A(s_t) 中。

A(s_t) = A_map(s_t) ∪ A_schedule(s_t)

其中：

- A_map(s_t) = {(q, u) | 比特 q 未分配, QPU u 有容量}
- A_schedule(s_t) = {(g, t) | 门 g 已就绪, 时间片 t 资源充足}

步骤二：为不同类型的动作生成专属的"键"(Key)

这是本方案的核心技巧。虽然所有的动作都在一个集合里被评估，但我们使用不同的神经网络来为不同类型的动作生成它们的"键"向量。这使得网络可以学习到两种动作的独特语义，同时保证它们最终的"键"向量是在同一个维度空间中，可以相互比较。

- 对于一个"分配比特"动作 a = (q, u):
    - 我们使用一个专门的、可学习的多层感知机 (MLP)，记为 MLP_map。
    - 它的输入是 GNN 输出的量子比特节点 q 的嵌入向量 **h_q^(L)** 和 QPU 节点 u 的嵌入向量 **h_u^(L)** 的拼接。
    - 其"键"向量 **K_a** 的计算公式为： **K_a = MLP_map([h_q^(L), h_u^(L)])**
- 对于一个"调度门"动作 a = (g, t):
    - 我们使用另一个独立的、专门的 MLP，记为 MLP_schedule。
    - 它的输入是 GNN 输出的门节点 g 的嵌入向量 **h_g^(L)** 和一个代表时间片 t 的特征向量 **h_t**（例如，通过位置编码生成）的拼接。
    - 其"键"向量 **K_a** 的计算公式为： **K_a = MLP_schedule([h_g^(L), h_t])**

关键点: MLP_map 和 MLP_schedule 的输出维度是相同的（例如，都是 d_model 维），这保证了所有"键"向量的格式统一。

步骤三：执行统一的 Attention 计算

一旦为统一动作集 A(s_t) 中的每一个动作都生成了对应的"键"向量 **K_a**，接下来的流程就和我们之前讨论的完全一样了：

1. 从 GNN 的输出中聚合生成一个代表全局意图的查询向量 **Q**。
2. 计算 **Q** 与每一个 **K_a** 之间的注意力分数 e_a。
3. 通过一个带掩码的 Softmax 函数，将所有分数转换为一个覆盖了所有类型动作的统一概率分布 π(a|s_t)。
4. 从这个统一分布中采样一个动作执行。

方案总结

这个方案的优势在于：

- **结构优雅:** 保持了单一策略网络的简洁性，避免了分层决策的复杂性。
- **语义丰富:** 通过不同动作类型设计专属的编码网络(MLP_map, MLP_schedule)，使得模型能够学习到两种决策的内在差异。
- **决策力强:** 允许智能体在每一步都进行全局权衡，直接比较"分配一个关键比特"和"调度一个瓶颈门"的收益，从而学习到更深刻、更高效的调度策略。

模型构建：即时奖励函数的具体数学形式

我们将即时奖励函数 R_immediate 定义为三个核心奖励组件的加权和。对于在状态 s 执行动作 a 并转移到状态 s' 的每一步，智能体都会收到以下奖励：

R_immediate(s, a, s') = w_alloc · R_alloc(a) + w_schedule · R_schedule(a) + w_progress · R_progress(a)

其中，w 是各项的权重系数。下面我们来详细定义每一个组件。

A. 分配奖励组件 (R_alloc)

该组件仅在智能体执行分配比特动作 a = (q, u) 时被触发，否则为 0。它的目标是引导智能体做出能降低通信成本的优质决策。

其数学形式定义为：

R_alloc(q, u) = ∑_{q_j∈MappedPartners(q)} w_{qj} · (r_colocate · δ(Π(q_j), u) - r_separate · (1 - δ(Π(q_j), u)))

- MappedPartners(q): 电路中与比特 q 有 CNOT 关系、且已经被分配的比特集合。
- w_{qj}: 比特 q 和 q_j 之间的交互强度（即 CNOT 数量），直接源自 UNIQ 的模型。
- Π(q_j): 比特 q_j 当前所在的 QPU。
- δ(·,·): 克罗内克 δ 函数。当两个 QPU 相同时 δ(Π(q_j), u) = 1，触发共置奖励；不同时为 0。
- C_{u,Π(q_j)}: 将比特 q 分配到 QPU u 后，与比特 q_j 之间产生的通信成本，源自 UNIQ 的模型。
- r_colocate, r_separate: 可调的奖励常数，分别代表共置的基础奖励值和分离的基础惩罚系数。

B. 调度奖励组件 (R_schedule)

该组件仅在智能体执行调度门动作 a = (g, t) 时被触发，否则为 0。它的目标是引导智能体形成紧凑、高效的调度方案。

其数学形式定义为：

R_schedule(g, t) = r_time/t + r_parallel_epr · I(is_parallel_epr(g, t))

- r_time/t: "争分夺秒"奖励。将门 g 调度在越早的时间片 t，获得的奖励越高。r_time 是一个奖励常数。
- I(·): 指示函数 (Indicator Function)。当条件为真时为 1，否则为 0。
- is_parallel_epr(g, t): 一个布尔函数，判断远程门 g 在时间片 t 安排 EPR 生成任务时，是否实现了与其它 EPR 任务的并行生成（根据资源状态 R_res 判断）。
- r_parallel_epr: 成功实现 EPR 并行生成时的额外奖励常数，鼓励智能体学习 UNIQ 的核心优化技巧之一。

C. 进度奖励组件 (R_progress)

该组件在智能体每执行一个动作时都会被触发，旨在鼓励智能体用尽可能少的步骤完成任务。

其数学形式定义为：

R_progress(a) = -r_step

- -r_step: 一个微小的负步进延罚。通过在每一步都施加一个小的负奖励，可以激励智能体找到完成调度的"最短路径"，避免冗余或无效的动作。

参数设定原则与初始值建议

奖励函数中的权重和常数是重要的超参数，需要通过实验进行微调。但我们可以根据一些原则来设定它们的初始值。

- **总体原则:** 即时奖励的绝对值应该远小于最终奖励 R_final 的期望值，以确保智能体不会"拉了芝麻丢了西瓜"，只顾优化眼前的小奖励而忽略了最终的全局目标。同时，各奖励组件的量级应大数相当，避免某个组件在学习初期完全主导梯度方向。
- **权重系数 (w):**
  - w_alloc 和 w_schedule: 这两个权重应反映我们对"通信成本"和"运行时间"的重视程度，可以参考最终目标函数中的 α 和 β。一个合理的初始设定是让它们处于同一量级，例如 w_alloc = 1.0, w_schedule = 1.0。
  - w_progress: 步进延罚是一个辅助项，其权重远小于其他项，例如 w_progress = 0.01。
- **奖励常数 (r):**
  - r_step: 应设为一个很小的正数，例如 r_step = 0.01。
  - r_colocate 和 r_separate: 这两项与通信成本相关。C_uv 本身有物理尺度（最短路径长度），我们可以设 r_separate = 1.0 使延罚与通信成本直接相关。共置奖励应是一个有意义的正反馈，可以设为 r_colocate = 0.5。
  - r_time: 为了让"争分夺秒"奖励在初期（t 较小）有一定影响力，可以设定 r_time = 1.0。
  - r_parallel_epr: 这是一个鼓励高级技巧的额外奖励，应该足够显著以被智能体注意到，可以设为 r_parallel_epr = 0.5。

模型构建：Critic 价值网络的具体架构

在 PPO 这种 Actor-Critic 框架中，Critic (评论家) 的角色是学习评估 Actor (演员) 所处的状态"好不好"。具体来说，它需要学习一个价值函数 V(s)，用于预测在当前状态 s 下，未来可能获得的总回报的期望值。这个价值估计是计算优势函数 (Advantage Function) 的基础，对于稳定和指导 Actor 的策略学习至关重要。

我们设计的 Critic 网络 V_φ（参数为 φ）的架构如下：

1. 输入的构建：全局图嵌入 (Global Graph Embedding)

Critic 评估的是整个状态的价值，而不是某个局部节点的价值。因此，它的输入不能是单个节点的嵌入向量，而必须是一个能够代表整个 DOC 状态图的全局表示。

我们通过一个读出 (Readout) 或池化 (Pooling) 的操作，来从 R-GCN 输出的所有节点嵌入向量 {**h_v^(L)**} 中，聚合生成一个单一的、固定维度的全局图嵌入向量 **h_G**。

一个简单而有效的方法是采用平均池化 (Mean Pooling)：

h_G = (1/|V'|) ∑_{v∈V'} h_v^(L)

其中：

- **h_v^(L)** 是 R-GCN 第 L 层输出的节点 v 的最终嵌入向量。
- V' 是图中所有门节点和量子比特节点的集合。我们通常不包含 QPU 节点，因为它们代表的是静态的硬件信息，而状态的价值多地由动态的、未完成的任务（门和比特）决定。
- **h_G** ∈ ℝ**^d_model** 就是最终代表整个图状态的向量，它将被作为 Critic 网络的输入。

2. 价值网络的具体架构：多层感知机 (MLP)

在获得了代表全局状态的嵌入向量 **h_G** 后，我们使用一个标准的多层感知机 (MLP) 来将这个高维的特征表示，映射到一个标量的价值估计值。

我们提出一个包含两个隐藏层的 MLP 结构：

- **输入层 (Input Layer):**
  - 接收全局图嵌入向量 **h_G**。
  - 维度: d_model（与 GNN 的隐藏维度相同，例如 128）。
- **隐藏层 1 (Hidden Layer 1):**
  - 全连接层，将输入从 d_model 维映射到更高维度（例如 256）。
  - 激活函数: ReLU (Rectified Linear Unit)。
  - 维度: 256
- **隐藏层 2 (Hidden Layer 2):**
  - 全连接层，保持维度不变。
  - 激活函数: ReLU。
  - 维度: 256
- **输出层 (Output Layer):**
  - 全连接层，将 256 维的特征映射到一个单一的标量值。
  - 激活函数: 线性激活 (Linear Activation)，即不使用激活函数。这一点至关重要，因为价值的取值范围是任意实数，而不是像分类问题一样被限制在 [0, 1] 区间。
  - 维度: 1

3. 完整的数学形式

整个 Critic 网络的计算过程可以表示为：

V_φ(s) = **W_3** · ReLU(**W_2** · ReLU(**W_1** · **h_G** + **b_1**) + **b_2**) + b_3

其中，φ = {**W_1**, **b_1**, **W_2**, **b_2**, **W_3**, b_3} 是 Critic 网络所有可学习的权重和偏置参数。

总结：这个"图池化 + MLP"的架构，使得 Critic 能够有效地从 R-GCN 提供的、对 DQC 状态的深度理解中，学习到一个准确的价值函数。这个准确的价值估计，将为 Actor 网络的策略更新提供稳定可靠的梯度信号，是整个 UNIQ-RL 框架能够成功收效的关键保障之一。

模型构建：PPO 算法的核心超参数选择

下面是我们为 UNIQ-RL 框架推荐的核心超参数及其选择理由。

1. 学习率 (Learning Rate, η)

- **建议值: 1 × 10^-4（或 0.0001）**
- **选择理由：**
  - **作用:** 学习率控制了每次梯度下降时模型权重更新的步长。
  - **考量:** 我们的 DQC 优化问题是一个非常复杂的组合优化问题，其奖励"地形"可能非常崎岖。选择一个相对较小的学习率，可以确保训练过程更加稳定，避免因为某一批数据的梯度噪音过大而导致策略更新步子迈得太大，从而"学坏"或使性能崩溃。特别是在模仿学习预训练之后，我们希望模型在已有良好策略的基础上进行微调，而不是颠覆性的改变，因此小学习率是更稳妥的选择。

2. 折扣因子 (Discount Factor, γ)

- **建议值: 0.99**
- **选择理由：**
  - **作用:** 折扣因子决定了智能体对未来奖励的重视程度。γ 越接近 1，智能体越有"远见"；越接近 0，则越"短视"。
  - **考量:** 我们的任务是一个有限长度的回合（Episode），最终的性能由整个调度方案的总运行时间和总通信成本决定。这是一个典型的需要长远规划的问题。因此，我们选择一个非常接近 1 的 γ 值，以确保智能体能够充分地将未来的奖励考虑在内，学会做出对全局最优有利的决策，而不是只顾眼前的即时奖励。

3. GAE 参数 (Generalized Advantage Estimation, λ)

- **建议值: 0.95**
- **选择理由：**
  - **作用:** GAE 参数 λ 用于在计算优势函数时，平衡"偏差"和"方差"。λ = 0 对应低方差但高偏差的估计，而 λ = 1 对应高方差但低偏差的估计。
  - **考量:** 在复杂的环境中，奖励信号的噪音很大，导致高方差。λ = 0.95 是一个在学术界和工业界被广泛采用的"甜点值"，它能够在不过度增加偏差的情况下，显著降低优势函数估计的方差，从而使学习过程更加稳定、高效。

4. 裁剪系数 (Clipping Parameter, ε)

- **建议值: 0.2**
- **选择理由：**
  - **作用:** 这是 PPO 算法的核心。它通过限制新旧策略之间的变化幅度（概率比率被限制在 [1 - ε, 1 + ε] 区间内），来防止单次更新对策略网络造成过大的、破坏性的改变。

- **考量:** ε = 0.2 是 PPO 原始论文中推荐的、并在后续无数研究中被验证为非常鲁棒的一个值。它允许策略进行有意义的、足够大的改进，同时又有效地保证了训练的稳定性，是 PPO 算法成功的关键。

5. 批量大小 (Batch & Mini-batch Sizes)

- **数据收集批量大小 (Rollout Buffer Size):**
  - **建议值：4096 步**
  - **选择理由:** 这是在进行一次模型更新前，策略与环境交互的总步数。一个足够大的批量能够包含更多样化的经验，从而计算出更稳定、更准确的梯度，有助于提升训练稳定性。
- **小批量大小 (Mini-batch Size):**
  - **建议值：128 步**
  - **选择理由:** 在模型更新阶段，我们将收集到的 4096 步数据分成若干个小批量进行随机梯度下降。128 是一个在计算效率（能充分利用 GPU 并行能力）和梯度随机性之间取得良好平衡的常用值。

总结: 这套超参数组合是经过广泛验证的、适用于复杂决策问题的强大基准。在我们的研究中，我们将以此起点，在具体的 DQC 环境中进行实验，并根据实际的收效情况进行微调。