计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
2025,34(12):215−227 [doi: 10.15888/j.cnki.csa.010036] [CSTR: 32024.14.csa.010036]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

分层拓扑-蚁群融合算法在 NISQ 量子电路调度优化中的应用①

李　晖[1,2], 王杰鹏[1], 韩子傲[1], 姬迎松[1], 陈禹彤[3]

[1](哈尔滨商业大学 计算机与信息工程学院, 哈尔滨 150028)
[2](哈尔滨商业大学 黑龙江省电子商务与信息处理重点实验室, 哈尔滨 150028)
[3](北京理工大学 信息与电子学院, 北京 100081)
通信作者: 王杰鹏, E-mail: ekkr18510@163.com

摘　要: 针对传统量子电路调度算法在调度质量、多样性以及对复杂量子电路结构适应性等方面存在的问题. 本文设计了层级量子电路调度-蚁群 (layered quantum circuit scheduling and ant colony optimization, LQCS-ACO) 融合算法. 将层级量子电路调度与蚁群算法相结合, 综合考虑量子门间的层依赖关系、SWAP 门操作数、CNOT 门数量以及电路层数等多种因素, 设计适合量子电路层结构的适应度函数. 同时采用基于量子电路层反馈动态信息素调整机制, 降低算法陷入局部最优的风险. 引入层间信息素传递方法, 考虑量子电路层影响, 完成调度信息的层间传递, 实现整体调度方案效率和全局优化效果提升. 实验结果表明, 在 2QAN 量子电路调度过程中, LQCS-ACO 相较于 2QAN 电路、HQAA 算法、ACO 算法, SWAP 门插入数量分别减少 5.8%、8.3% 和 7.3%, CNOT 门数量分别减少 8.2%、9.2% 和 33.8%.

关键词: 量子电路调度; 层级量子电路调度-蚁群融合算法; 2QAN 电路; 逐层冲突解决; 层次拓扑调度

引用格式: 李晖,王杰鹏,韩子傲,姬迎松,陈禹彤.分层拓扑-蚁群融合算法在 NISQ 量子电路调度优化中的应用.计算机系统应用,2025,34(12): 215–227. http://www.c-s-a.org.cn/1003-3254/10036.html

Application of Layered-topology-ant-colony Fusion Algorithm in NISQ Quantum Circuit Scheduling Optimization

LI Hui[1,2], WANG Jie-Peng[1], HAN Zi-Ao[1], JI Ying-Song[1], CHEN Yu-Tong[3]

[1](School of Computer and Information Engineering, Harbin University of Commerce, Harbin 150028, China)
[2](Heilongjiang Provincial Key Laboratory of Electronic Commerce and Information Processing, Harbin University of Commerce, Harbin 150028, China)
[3](School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, China)

Abstract: To solve the problems in the traditional quantum circuit scheduling algorithms in terms of scheduling quality, diversity, and adaptability to complex quantum circuit structures, this study proposes a layered quantum circuit scheduling and ant colony optimization (LQCS-ACO) fusion algorithm. By combining the layered quantum circuit scheduling and ant colony algorithm, and comprehensively considering multiple factors such as the layer dependency relationship between quantum gates, the number of SWAP gate operations, the number of CNOT gates, and the number of circuit layers, a fitness function suitable for the quantum circuit layer structure is designed. Meanwhile, a dynamic pheromone adjustment mechanism based on the feedback of quantum circuit layers is adopted to reduce the risk of the algorithm falling into a local optimum. Meanwhile, an inter-layer pheromone transmission method is introduced. By considering the

① 基金项目: 黑龙江省自然科学基金 (LH2024F042); 黑龙江省普通本科高等学校青年创新人才培养计划 (UNPYSCT-2020212); 哈尔滨商业大学“青年科研创新人才”培育计划 (2023-KYYWF-0983)
收稿时间: 2025-05-11; 修改时间: 2025-06-24; 采用时间: 2025-07-29; csa 在线出版时间: 2025-11-04
CNKI 网络首发时间: 2025-11-05

influence of quantum circuit layers, the inter-layer transmission of scheduling information is completed, and the efficiency of the overall scheduling scheme and the global optimization effect are improved. Experimental results show that in the 2QAN quantum circuit scheduling process, compared with the 2QAN circuit, HQAA algorithm, and ACO algorithm, the number of inserted SWAP gates of LQCS-ACO is reduced by 5.8%, 8.3%, and 7.3% respectively, and the number of CNOT gates decreases by 8.2%, 9.2%, and 33.8% respectively.

Key words: quantum circuit scheduling; layered quantum circuit scheduling and ant colony optimization fusion algorithm; 2QAN circuit; layer-by-layer conflict resolution; layered topology scheduling

量子计算作为一种新兴的计算模式, 近年来得到了广泛的关注. 得益于量子比特在量子纠缠等量子力学效应上的独特优势, 量子计算具有解决当前棘手的中大规模问题 (如大数分解、组合优化、量子化学模拟等) 的潜力. 与在二进制/逻辑位上运行的经典计算机不同, 量子计算机基于多值比特 (量子比特) 运行, 这些量子比特可以表示许多不同的逻辑状态 (量子态)[1]. 尽管现在正处于嘈杂中尺度量子 (noisy intermediate-scale quantum, NISQ) 时代, NISQ 设备的量子位数不足进行量子纠错, 但已足够解决远超出传统计算机能力的实际问题[2−4]. 量子计算机虽拥有强大的潜力, 但受限于量子比特数和退相干效应, 目前实际应用仍面临诸多挑战, 其中最为突出的挑战之一便是如何高效地设计和调度量子电路. 量子电路通过一些量子门按照特定的顺序对量子比特进行操作, 从而实现各种量子算法和计算任务. 量子电路或量子算法在量子计算设备上执行要考虑到 3 个主要约束: 分别是逻辑依赖性, 即满足算法中固有操作顺序; 任何量子比特都不能同时参与一个以上的门; 双量子比特门只能在物理连接或者相互作用的量子比特之间实现. 要解决这些问题涉及映射、调度、纠错等方面的工作. 其中量子电路调度算法关键的作用. 量子电路调度算法根据量子计算的特点和硬件约束, 找到最优或较优的量子门操作顺序和量子比特分配方案, 以提高量子计算的效率和性能[5−10].

随着量子计算硬件规模扩展与性能迭代, 量子电路面临更加复杂的应用问题, 如多任务并行调度、异构硬件适配以及实时性等. 为了提高量子计算的高效性和保真性, 研究人员聚焦量子电路调度算法优化, 通过最小化操作数以提升计算效率. Bhoumik 等人[11]提出了一种量子电路调度器, 通过对电路进行切割并优化子电路的调度, 最大化了整体保真度, 同时确保不超过硬件的最大执行时间. 该方法在不同基准电路上的保真度显著高于未切割电路在噪声最小的设备上执行的结果, 展示了在现有硬件限制下优化量子电路执行性能的潜力. Romero-Alvarez 等人[12]提出了一种量子电路调度技术, 通过将来自不同用户的电路合并执行, 以减少等待时间并优化量子计算机的使用. Guerreschi[13]提出了一种新的量子信息路由优化方法, 关注所有可能的路由操作, 优先选择对整个量子电路执行模式最有利的操作. 通过调度变分算法并量化制造决策对算法性能的影响, 尤其是在超导量子芯片中, 评估了不同量子比特频率对量子傅里叶变换性能的影响. Bahreini 等人[14]建立了一种混合整数非线性规划模型, 用于量子电路的布局和调度, 以最小化延迟, 并确定量子门的位置和操作顺序. Venturelli 等人[15]将量子电路编译问题转化为时间规划问题, 提出了一种方法用于编译满足最近邻约束的超导量子硬件架构, 特别针对 QAOA 电路进行优化. 通过初步实验, 验证了时间规划方法在量子电路编译中的可行性, 并展示了该方法在优化量子电路时具有较大的潜力. Whitney 等人[16]设计了一种量子电路的计算机辅助设计流程, 包括自动布局和控制逻辑提取. 通过提出两种多项式时间设计启发式算法, 并与现有的自动化网格布局和调度机制进行比较, 证明了基于数据流的启发式算法在延迟和管道化潜力方面具有更好的性能. Booth 等人[17]将约束编程与时间规划相结合, 用于量子电路编译问题中的任务调度. 通过将时间规划方法找到的解作为起始点, 结合 CP 模型进一步优化. Gong 等人[18]基于变分量子电路的量子卷积神经网络 (QCNN), 利用量子计算的优势提升图像分类任务的性能. Ruiz 等人[19]提出了一种基于深度强化学习的量子电路优化方法 Alpha Tensor-Quantum, 通过将量子电路优化问题转化为张量分解问题. Li 等人[20]提出了基于强化学习的量子电路优化器 Quarl, 通过将动作空间分解为门选择和变换选择的分层设计, 并利

用图神经网络进行状态表示, 解决了量子电路优化中动作空间大且变化、状态表示非均匀的问题. Li 等人[21]提出了一种基于离散粒子群优化 (DPSO) 和深度强化学习 (DRL) 的量子电路映射方法, 解决了量子电路映射中物理量子比特数量有限、连接稀疏导致的量子比特分配和门调度难题. Itoko 等人[22]将量子操作调度问题视为一种特殊类型的作业车间问题, 并通过约束编程进行建模, 考虑量子操作之间的可交换性.

避免交叉干扰是量子线路调度问题中不可忽视的一部分, Bhoumik 等人[23]提出了一种优化问题, 通过整数线性规划对多个量子线路进行调度, 确保量子比特的最近邻排列并最大化保真度, 同时避免交叉干扰. 实验表明, 该方法在 27 量子比特和 127 量子比特硬件设备上, 分别在量子比特和时间方面提供了 2 倍和 3 倍的利用率提升. Kurowski 等人[24]设计了一种基于量子门模型的调度方法, 通过量子近似优化算法解决作业车间调度问题 (JSSP). 该方法利用时间索引的 JSSP 实例表示构建成本哈密顿量, 并通过实验验证了 QAOA 在求解 JSSP 问题中的效率与准确性. Metodi 等人[25]提出并评估了一种物理操作调度器, 用于任意量子线路的调度. 该调度器通过接收电路和物理布局的描述, 输出一系列操作序列, 揭示电路中的通信需求和可用并行性, 并生成可直接在工具上模拟的量子汇编语言文件. Kan 等人[26]提出了 FitCut 方法, 通过将量子线路转化为加权图, 并采用基于社区的自底向上的方法, 根据每个工作节点的资源约束切割电路. 同时, FitCut 引入了一种调度算法, 优化了多节点系统中的资源利用, 显著提高了计算效率和资源利用率. Gokhale 等人[27]提出了一种利用三态量子比特 (qutrits) 的新方法, 通过对广义 Toffoli 门进行对数深度分解, 显著提高了量子线路的运行效率和成本.

尽管当前量子电路调度算法能够处理基本的门顺序优化, 但在应对较为复杂的电路和硬件约束时仍面临以下问题: (1) 层依赖关系处理不足, 全局优化问题受限; (2) 硬件拓扑适应性较差, 未能够优先调度相邻量子门导致过多的冗余门数; (3) 调度策略无法适应量子电路的量子特性. 基于此, 针对传统蚁群算法在量子电路调度中的容易陷入局部最优以及对量子门拓扑结构适应性较差的局限性, 提出了层级量子电路调度-蚁群融合算法并将其应用到 2QAN 电路. 设计适合量子电路层结构的适应度函数, 用于评估解的质量, 提出改进的信息素更新策略, 通过引入适合量子电路层结构的局部搜索算法, 提升蚁群算法在量子电路调度中的全局搜索能力和局部优化能力. 算法采用动态调整机制, 根据调度过程中的变化调整信息素的权重, 减少算法陷入局部最优解的可能, 提高调度效率.

1 2QAN 量子电路调度优化

1.1 量子电路层

定义 1. 量子电路层. 在同一时间步内可以并行执行且量子比特间无冲突的一组量子门集合.

设一个量子电路有 n 个量子比特, 第 l 层量子门集合为$G_l = \{g_{l,1}, g_{l,2}, \cdots, g_{l,k}\}$, 其中每个门$g_{l,k}$作用于一个或多个量子比特, 易知量子电路层有如下性质.

并行性: 任意$g_{l,i}, g_{l,j} \in G_l$, 同时$g_{l,i}$作用于量子比特集合 P_i, $g_{l,j}$作用于量子比特集合 P_j, 且$P_i \cap P_j = \varnothing$, 则$g_{l,i}, g_{l,j}$可以在层 l 框架下并行执行.

顺序性: 设任意$g_{l,i} \in G_l$, $g_{l+1,i} \in G_{l+1}$, $g_{l+1,i}$可以执行的必要条件是当且仅当$g_{l,i}$被执行. 由顺序性易知, 量子电路的总体操作为: $U = U_L \circ U_{L-1} \circ \cdots \circ U_1$, 其中$U_l$是第 l 层的整体操作, $\circ$表示操作的复合.

1.2 量子电路调度优化

量子电路调度优化主要是提高量子电路执行效率与可靠性. 本文主要考虑以下 3 个方面.

(1) 门级优化: 减少量子门总数, 合并可以组合的量子门操作, 消除冗余门操作或者将复杂门分解成基本门集.

(2) 并行优化: 在考虑硬件拓扑约束情况下最大化量子门并行执行.

(3) 深度优化: 在硬件约束情况下, 尽可能地减少量子电路深度, 深度减少即执行时间减少.

图 1 展示了量子电路优化前后对比图, 所举例子为 5 量子位哈密顿电路编译网络架构. 其中, 节点表示量子位, 边表示量子位之间的连接性. 量子电路图中水平线表示一个量子比特, 虚线划分电路层结构, 同一层内门操作可以并行执行.

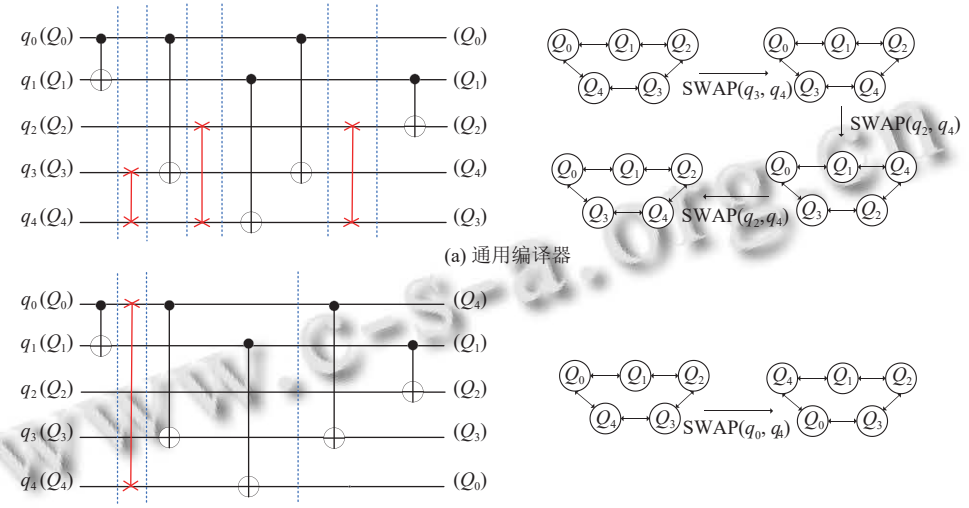
设 q 表示逻辑量子位, Q 表示物理量子位, 图 1(a) 中左图逻辑电路的物理架构可以用图 1(a) 中右图表示, 电路门序列形式为:

$$C = ((q_0, q_1), (q_0, q_3), (q_1, q_4), (q_0, q_3), (q_1, q_2))$$

图 1(a) 中, 描述了门依赖性的通用编译器的编译

结果, 插入 3 个 SWAP 门 (1 个 SWAP 门相当于 3 个 CNOT 门, 因此每插入一个 SWAP 门深度增加 3). 并输出具有 3 个 SWAP 操作和 5 个 CNOT 操作并且$D = 13$的电路.

图 1(b) 利用哈密顿模拟问题中灵活算子排列, 相比之下, 考虑运算符排列灵活性的编译器仅使用 1 个 SWAP 门, 进行调度优化后电路只有 8 个双量子位门即 1 个 SWAP 操作和 5 个 CNOT 操作且$D = 7$, 同时这个 SWAP 门可以与电路中其他门组合, 可以进一步减少门数和电路深度.



(a) 通用编译器

(b) 调度优化后编译结果

图 1 量子电路调度优化前后对比

1.3 2QAN 量子门调度策略

2QAN 是一种优化 2-local 量子比特哈密顿模拟问题量子电路的编译器. 由于量子硬件存在连接性限制, 传统的调度方法难以充分优化电路. 2QAN 的门调度利用哈密顿量中算子排列的灵活性, 提出了一种混合调度方法, 对于初始量子比特映射中的最近邻电路门, 采用图着色算法进行调度; 对于其他量子比特映射, 在考虑 SWAP 门和电路门依赖关系的基础上, 按照“尽可能晚 (ALAP)”的原则进行调度, 即先寻找在当前映射下量子比特为最近邻且空闲的电路门进行调度, 再寻找满足条件的 SWAP 门, 每插入一个 SWAP 门就更新量子比特映射, 直到所有门都被调度完成, 最后反转调度顺序.

和多数传统量子调度策略类似, 2QAN 的电路调度策略构建于一维量子数组之上. 在实施分层调度的过程中, 它采用串行操作模式, 这使得量子计算的并行特性未能得到充分发挥. 不仅如此, 2QAN 还忽视了量子电路运行时的一个重要事实, 即层内操作具备进一步并行化的潜力, 从而限制了整体调度效率的提升. 在针对调度质量、多样性以及对复杂量子电路结构的适应性方面问题, 本文设计了层级量子电路调度-蚁群融合算法. 通过结合量子电路的层级结构和蚁群算法的优势, 力求在量子电路调度问题中实现更出色的优化效果.

2 层级量子电路调度-蚁群融合算法

2.1 蚁群算法

蚁群算法 (ant colony optimization, ACO) 是一种受生物启发的优化算法[28–33], 其核心灵感来源于自然界蚂蚁觅食过程中寻找到最短路径的行为. 真实的蚂蚁群体在寻找食物时, 会在路径上释放一种化学物质, 称为信息素 (pheromone). 其他蚂蚁可以感知到这些信息素的浓度, 倾向于沿着信息素浓度较高的路径行进, 从而形成群体的协作效应. 这一现象导致信息素较高的路径会吸引更多的蚂蚁选择, 路径逐渐强化, 最终整个蚁群能够找到一条有效的最短路径.

蚂蚁的移动规则是基于信息素浓度和启发因子的权衡. 对于个体, 它从节点 i 移动到节点 j 的概率 p_{ij} 由式 (1) 决定.

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}, & j\in k_{\text{allowed}}\\ 0, & j\in k_{\text{unallowed}}\end{cases} \tag{1}$$

其中, τ_{ij}表示节点 i 和节点 j 之间的路径上的信息素浓

度, η_{ij}表示启发因子.

在每次迭代结束后, 个体会更新路径上的信息素浓度. 更新规则主要包含信息素的挥发和增加.

信息素挥发: 信息素会随着时间的推移逐渐减少, 防止算法陷入局部最优.

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\tau_{ij}(t,t+1) \tag{2}$$

其中, ρ 是信息素挥发率, 取值范围为 (0, 1].

信息素增加: 在每次迭代中, 优质解上的路径信息素浓度增加.

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t) \tag{3}$$

其中, m 表示蚂蚁数量, $\Delta\tau_{ij}(t)$表示的是第 k 个个体在路径 i 到 j 上释放的信息素增量.

$$\Delta\tau_{ij}^{k}(t,t+1)=\begin{cases}\dfrac{Q}{L_k}, & k\text{经过}(i,j)\\ 0, & k\text{不经过}(i,j)\end{cases} \tag{4}$$

其中, Q 是一个常数, 代表个体释放的信息素总量, L_k 是第 k 个个体找到的路径长度.

蚁群算法在量子电路调度中的优势主要体现在其并行搜索和动态优化特性. 量子电路调度需在硬件连通性限制下对量子门的执行顺序进行优化, 以降低电路深度并提高执行效率. 蚁群算法的自适应路径优化能力可以很好地应对量子门之间复杂的依赖关系, 通过信息素引导来识别最佳路径, 同时有效减少冲突和 SWAP 门的引入. 此外, 蚁群算法的局部更新策略使其能够灵活适应电路调度中不同层级的复杂度和依赖关系, 特别适合分层调度结构的量子电路优化. 然而, 由于依赖信息素的反馈和挥发过程, 蚁群算法在计算资源的需求上较高, 且在极大规模搜索空间中, 算法的收敛速度可能较慢.

2.2 LQCS-ACO 算法

LQCS-ACO 算法是在量子电路调度问题中提出的一种新的优化算法, 旨在通过层级量子电路调度与蚁群算法相结合, 提升量子电路调度的效率和优化质量.

2.2.1 适应度函数设计

为了充分评估基于层结构的量子电路调度方案, 适应度函数 F_{Layer} 综合考虑了电路层数、SWAP 门数量、CNOT 门数量等多个关键因素. 通过对各层的调度进行细致分析, 反映不同层级电路之间的协调性及执行效率, 从而有效减少量子门延迟、SWAP 门插入量以及总的电路执行时间. 具体而言, 适应度函数 F_{Layer} 通过式 (5) 计算:

$$F_{\text{Layer}}=[\alpha\cdot g_{\text{Layer}}(x)+\beta\cdot h_{\text{Q}}(x,f_{\text{SWAP}},f_{\text{CNOT}})]\cdot\left(\frac{N}{1+\omega_{\text{D}}}\right) \tag{5}$$

其中, F_{Layer} 为整体适应度函数输出, 数值越小表示调度方案越优秀. N 表示电路层数, 通过拓扑排序对当前调度方案中所有量子门分层后的层数统计得到. 层数越少, 表示量子电路调度方案的执行深度越小. f_{CNOT} 为 CNOT 门数量, 反映多量子比特门在调度方案中的出现情况. f_{SWAP} 是 SWAP 门数量, 用于评估满足邻近性约束所需的 SWAP 门操作数. SWAP 门用于交换两个物理量子比特的位置. 在非邻近量子比特之间执行 CNOT 操作时, 为了让量子比特变为邻近位置, 需要在其路径上引入若干个 SWAP 门, 使得操作符合量子电路的物理结构. SWAP 门的引入往往会增加量子电路的深度, 因此在调度算法中通常希望将 SWAP 数量最小化, 从而提升电路的执行效率. α, β与ω_{D}是加权参数, 用于平衡不同优化目标的权重. α为反映层结构优化对调度质量的影响, 主要衡量量子门的层次深度及调度并行度. β为影响 CNOT 和 SWAP 门数量的优化权重, 确保在 SWAP 插入过程中维持电路优化的稳定性. ω_{D}衡量了电路整体执行时间, 以确保整体优化目标.

此外, 适应度函数中还加入了两项改进特性, 以进一步提升评估的全面性.

(1) 层结构评估函数 $g_{\text{Layer}}(x)$: 用于量化各层电路结构之间的协调性. 其输入参数 x 表示当前的调度方案, 评估每层的量子门分布和资源占用情况. 具体形式如式 (6) 所示:

$$g_{\text{Layer}}(x)=\sum_{i=1}^{N}\left(\frac{1}{L_i}\cdot\sum_{j=1}^{L_i}\exp(-\gamma\cdot d_{ij})\right) \tag{6}$$

其中, L_i 是第 i 层中的量子门数量. d_{ij} 为第 i 层中第 j 个量子门之间的欧几里得距离, 距离越小, 意味着该层电路结构越紧凑, 资源利用率越高. γ为控制距离影响的参数, 数值越大, 则距离对评估的影响越明显.

(2) 量子特性评估函数 $h_{\text{Q}}(x,f_{\text{CNOT}},f_{\text{SWAP}})$: 结合量子行为特点, 评估当前调度方案的量子效能. 该项在算法运行中自适应调整, 结合 SWAP 门和 CNOT 门的数量, 计算优化的必要性.

$$h_{\text{Q}}(x,f_{\text{SWAP}},f_{\text{CNOT}})=\frac{\cos(\theta\cdot f_{\text{SWAP}})}{1+f_{\text{CNOT}}\cdot\exp(-\eta\cdot f_{\text{SWAP}})} \tag{7}$$

其中, θ 和 η 为调节参数, 用于控制 CNOT 门和 SWAP 门在调度方案中的重要性. h_Q 中的分母项对 CNOT 门和 SWAP 门的数量进行平衡, 当 CNOT 门数量相对较少时, 适应度函数优先评估 SWAP 门数量, 以保证高效的量子门调度.

适应度函数综合考虑 CNOT 门数量、SWAP 门数量和电路深度, 并采用动态权重调整机制, 使其能够根据调度阶段的不同需求进行优化, 以全面评估量子电路的执行效率和硬件资源消耗. 在初始阶段, 优化重点放在降低电路深度, 而在后期则更关注 CNOT 门和 SWAP 门的数量, 以精细控制硬件资源使用. 此灵活调整机制避免算法陷入局部最优, 使其适应复杂的量子电路调度需求, 实现全局优化.

2.2.2 路径移动

在 LQCS-ACO 算法中, 个体路径由量子电路的各个调度操作构成, 而每次路径选择的决策直接影响最终的调度结果. 不同于传统的蚁群算法, LQCS-ACO 针对量子电路层结构进行了特化设计, 确保个体能够在调度过程中适应电路层级的约束条件, 并根据量子比特之间的关系动态选择最优路径. 在路径选择过程中, 个体不仅依据当前的路径信息素浓度进行决策, 还结合量子电路的层结构进行局部优化, 避免无效的路径移动. 个体根据适应度函数的反馈信息调整路径选择策略, 以最大程度上减少 SWAP 门和 CNOT 门的数量, 并优化电路的执行时间和深度. 路径的移动和调整遵循了量子电路中层次化调度的原则, 确保了优化过程与实际硬件实现的紧密结合.

在基于层的量子电路调度中, 路径移动主要应用于在同一层内调度量子门, 并确保跨层调度时减少依赖关系和 SWAP 门的使用. 在路径移动过程中, 每个节点代表一个可执行的量子门, 个体在移动时需选择从一个量子门到下一个量子门. 路径的选择受信息素和启发式信息的共同影响, 以优化调度顺序并提高搜索效率.

在量子电路调度问题中, 每个逻辑量子比特需要满足物理量子比特的连接关系, 并且量子门之间存在着依赖关系, 这种关系决定了量子门的调度顺序. 在基于层结构的量子电路调度中, 每个个体代表了一个完整的调度方案, 即为所有量子门确定执行顺序的整体映射和策略规划. 每条路径对应于量子门在各层内的排列顺序, 即个体在搜索空间中选择的量子门执行序列. 个体通过蚁群算法进行路径选择, 其中每一步从一个量子门移动到下一个执行的量子门, 最终形成一条完整的执行路径. 具体而言, 每一层的路径选择为:

$$p_{ij}(t)=\frac{[\tau_{ij}^{l}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{k\in N_i}[\tau_{ij}^{l}(t)]^{\alpha}[\eta_{ij}]^{\beta}} \tag{8}$$

每个可执行量子门代表了一个节点. 其中 $p_{ij}(t)$ 代表了在第 t 次迭代时, 从节点 i 转移到节点 j 的概率, 这个概率值决定了个体选择路径的倾向性. $\tau_{ij}^{l}(t)$ 表示在第 l 层的第 t 次迭代时, 路径 (i, j) 上的信息素浓度. 信息素是一种虚拟的“标记”, 用来模拟蚂蚁在路径上留下的痕迹. 浓度越高, 表示此路径在过去的迭代中被更多次选择, 因此更有可能是高质量的路径. α 是信息素的权重系数, 决定信息素对路径选择的影响程度. 如果 α 值较大, 个体会更倾向于选择信息素浓度高的路径. β 是启发式信息的权重系数, 决定启发式信息对路径选择的影响. 如果 β 值较大, 个体会更倾向于选择启发式信息高的路径, 即那些物理距离更短、操作更简单的路径. N_i 是节点 i 的可选邻居节点集合, 即从节点 i 出发的所有可能路径. $\sum_{k\in N_i}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}$ 是归一化系数, 确保所有路径选择概率之和为 1. 这是因为个体只能选择一条路径, 因此需要将所有可能路径的权重归一化为一个概率分布. η_{ij} 为启发式信息. 如式 (9) 所示:

$$\eta_{ij}=\frac{1}{d_{ij}+\gamma\cdot\delta_{ij}} \tag{9}$$

其中, d_{ij} 是衡量量子比特之间的物理距离的距离参数. 较短的距离意味着操作效率更高, 因此启发式信息值会更高. γ 为控制常数, 用于调整路径中物理距离对调度的影响程度. δ 为量子门类型因子, 用于区分不同类型的量子门操作对路径选择的影响. 一般来说, SWAP 门的量子门类型因子大于 CNOT 门的量子门类型因子大于单量子比特门的量子门类型因子.

在实际操作中, 由于量子比特的物理邻接限制, 一些逻辑门操作由于不满足物理量子比特的结构, 所以需要插入额外的 SWAP 门来调整量子比特的位置. 因此, 可以通过调整节点转移概率, 优先选择那些有助于减少 SWAP 门插入的路径, 并通过调整信息素的浓度, 鼓励蚂蚁 (即量子电路调度方案) 更多地探索那些潜在的高效路径. 在每一轮迭代中, 调整节点转移概率:

$$p_{ij}(t+1)=p_{ij}(t)\cdot\left(1+\frac{\delta L_{i,j}}{L_{\text{avg}}}\right) \tag{10}$$

其中, $p_{ij}(t+1)$ 为更新后节点 i 与 j 的转移概率, $\delta L_{i,j}$ 为

当前路径与最优路径之间的差异, L_{avg} 为所有路径长度的平均值.

2.2.3 信息素更新

信息素更新策略在蚁群算法中起到了核心作用, 尤其在复杂的量子电路调度问题中, 它决定了算法在搜索空间中的探索效率和解的优劣. 在基于层结构的量子电路调度中, 每个路径选择的优化都依赖于个体在调度过程中留下的信息素痕迹, 因此, 信息素的更新策略直接影响了算法在解决量子门调度和优化问题上的效果. 通过调整信息素的浓度, 我们可以引导个体在后续迭代中更倾向于选择那些历史上表现优异的路径, 进而逐步趋向于全局最优解.

在量子电路调度问题中, 信息素的更新不仅是为了单纯地强化某条路径的选择, 也是为了适应量子电路的层结构特点. 量子电路由多个量子门层组成, 每一层内部可能有多个量子门并行执行, 而跨层之间则存在严格的依赖关系. 这就意味着, 调度算法不仅需要在层内优化门操作的执行顺序, 还要在层间优化整个量子电路的执行流程. 传统蚁群算法的信息素更新策略通常只适用于平面图结构, 未能充分利用量子电路的层次结构, 因此在处理复杂的量子电路调度时容易陷入局部最优解.

为了应对这一挑战, LQCS-ACO 算法对信息素更新策略进行了改进, 以更好地适应量子电路调度的特点. 首先提出了一种分层信息素更新的方法. 在这种策略下, 每一层的量子门调度都有独立的信息素浓度更新, 这意味着当某一层的调度路径被证明是高效的, 那么这一层的信息素浓度会更快地增强, 从而在下一次迭代中更优先选择这条路径. 具体而言, 将通过层内和层间的联合机制来进行信息素的更新, 使得信息素不仅能在同一层内传播, 同时能够跨层传递, 从而更好地优化量子电路调度.

(1) 初始化

信息素浓度初始化用于为算法提供初始搜索方向. 在量子电路调度中, 根据电路层 l 的深度对初始信息素进行微调, 使得信息素在更深的层次上略有增大. 这种初始化方式确保算法在初始阶段能够更快地聚焦于复杂的深层路径, 提高早期的搜索效率, 从而加快对有效调度方案的探索, 如式 (11) 所示:

$$\tau_{ij}^{l}(0)=\tau(0)\cdot\left(1+\frac{1}{\sqrt{l+1}}\right) \tag{11}$$

其中, $\tau(0)$是初始信息素浓度. 初始化信息素时, 本算法考虑当前层编号 l, 使得初始信息素在更深的层次上略有增加, 以适应更复杂的调度需求.

在进行初始化之后, 需要根据量子门的依赖关系构建电路的有向无环图, 将量子门划分为多个调度层. 接着需要更新信息素. 基础信息素的更新可以用于调整蚂蚁在量子电路调度过程中选择路径的信息素浓度, 从而引导后续个体优先选择经过验证的高效路径. 通过定期更新信息素, 可以增强在同一层内具有较好调度性能的路径的信息素浓度.

$$\tau_{ij}^{l}(t+1)=(1-\rho_l)\cdot\tau_{ij}^{l}(t)+\lambda\cdot\Delta\tau_{ij}^{l} \tag{12}$$

其中, ρ_l是层 l 的信息素挥发率. 在传统蚁群算法中, 信息素的挥发率通常是一个固定值. 而对于量子电路来说, 通过引入基于调度性能的动态挥发率调整机制, 可以在不同的调度阶段自适应地改变信息素的挥发速度, 使得算法在全局搜索和局部优化之间取得更好的平衡, 从而有效减少 SWAP 门和 CNOT 门数量, 提升调度效率. 信息素浓度越高, 越倾向于将非相邻的量子门调度在一起, 从而尽量减少额外的 SWAP 门插入需求. $\Delta\tau_{ij}^{l}$是基于路径的调度效率计算的信息素增量, 可由式 (13) 计算获得.

$$\Delta\tau_{ij}^{l}=\frac{Q}{F_{\text{Layer}}+\varepsilon} \tag{13}$$

其中, Q 是信息素增量的常数因子, ε是防止除零的微小常数.

(2) 信息素更新

为提高优化质量, 算法采用基于适应度函数的动态信息素调节机制. 具体而言, 信息素增量不再按固定比例分配, 而是依据路径优化程度动态调整. 若某路径显著减少 SWAP 门插入数量、降低 CNOT 门成本或压缩电路层深度, 其适应度值更优, 从而获得更高的信息素增量. 这一策略避免了传统方法因路径长度差异导致的信息素偏差. 例如, 仅依赖路径长度可能导致短路径被过度选择, 但该路径可能因过多 SWAP 门插入而增加整体开销. 借助适应度函数调节信息素分配, 算法能精准识别高质量调度方案, 引导搜索向低资源消耗和高执行效率方向收敛.

量子电路的层次化特性要求算法兼顾层内优化与层间协调. 某一层的调度影响后续层的量子比特布局, 进而影响整体资源开销. 例如, 若前一层路径选择不当,

导致量子比特分散, 后续层可能需额外 SWAP 门调整布局, 增加资源消耗. 为此, 算法引入跨层信息素传递机制, 将前一层优化路径的信息素动态注入当前层的决策过程中. 这一机制并非简单的数值累加, 而是通过历史路径质量评估影响当前路径选择, 使个体更倾向于与前层优化模式兼容的调度方案.

此外, 信息素传递强度可根据电路深度自适应调整: 初始层保持较强探索能力, 而接近输出层时强化历史优质路径的引导, 以平衡探索与收敛. 通过该机制, 算法在全局视角下构建层间协同优化方案, 实现资源消耗与执行效率的最优平衡, 提高量子电路调度质量.

$$\tau_{ij}^{l+1}(t+1)=\tau_{ij}^{l}(t+1)+\theta\cdot\frac{\sum_{k=l}^{l}\tau_{ij}^{k}(t)}{l} \tag{14}$$

其中, θ 是层间信息素传播系数, 层间信息素传播帮助后续层更快速找到优化路径, 减少全局 SWAP 门数量. 结合以上公式, 可以得出层间信息素更新的综合公式为:

$$\tau_{ij}^{l}(t+1)=(1-\rho_l)\cdot\tau_{ij}^{l}(t)+\lambda\cdot\Delta\tau_{ij}^{l}+\theta\cdot\frac{\sum_{k=l}^{l}\tau_{ij}^{k}(t)}{l} \tag{15}$$

(3) 调整自适应

在传统的蚁群算法中, 信息素的更新机制通常依据个体在路径上走过的代价来进行. 最优路径的选择依赖于当前路径上的信息素浓度, 高浓度的路径往往被更多的个体选择, 导致信息素逐步向优路径集中. 这一机制在许多经典的优化问题中表现良好, 但在量子电路调度中, 由于电路结构的复杂性, 局部最优解的可能性更高. 在量子电路调度中, 量子比特的层级结构和相互依赖的量子门常导致路径选择不是简单的路径, 而是需要考虑路径间的相互作用. 例如, 量子电路中的 CNOT 门和 SWAP 门在不同层次之间的依赖关系、量子比特之间的连接约束都使得简单的路径选择无法高效地优化调度问题. 因此, 传统蚁群算法容易在局部最优解附近停滞, 导致收敛速度缓慢. 基于此, 本算法为了适应不同的层次结构, 引入动态自适应系数γ:

$$\gamma_l=\frac{1}{1+\mathrm{e}^{-k(l-L/2)}} \tag{16}$$

其中, k 是调整速率常数, L 为当前电路总层数. 该公式使用了一个 Sigmoid 函数, 根据当前层的深度 l 来动态调节系数. 通过式 (16) 可知, 当 l 靠近 $L/2$ 时, 调整系数增大, 反之调整系数减小. 层数接近 $L/2$ 时, 网络中的决策较为复杂, 层间关系较为紧密, 局部最优解的可能性较高. 通过增强中间层的信息素更新权重, 算法能够更好地进行全局搜索, 避免陷入局部最优解. 此时, 信息素挥发率ρ_l的计算公式可以得出:

$$\rho_l=\rho_0\cdot(1-\gamma_1) \tag{17}$$

其中, ρ_0为基础挥发率. 当γ_1较大时, 说明当前层处于电路的中间部分, 挥发率降低, 保留更多信息素以鼓励探索; 当γ_1较小 (如在初始或末尾层) 时, 挥发率增大以加速收敛.

2.2.4 终止条件

在量子电路调度问题中, 优化的核心目标是减少跨物理量子比特之间的 SWAP 门和 CNOT 门的数量, 从而降低调度成本, 缩短总的电路执行时间. 量子门的调度受到硬件拓扑结构和量子比特耦合限制的影响, 因此在实际应用中, 调度的效率直接关系到量子电路的运行性能.

基于信息素分布的收敛终止条件的核心理念在于通过度量信息素在各层的分布情况来判断当前解的稳定性. 当算法在多次迭代后, 某些路径上的信息素浓度趋于稳定, 而其他路径上的信息素则逐渐减弱, 这表明个体在解空间中的探索已经趋向集中, 所有个体开始选择相同或相似的调度路径. 这种现象通常意味着算法可能已经接近收敛点. 然而, 在传统蚁群算法中, 如果直接依赖这种信息素积累的特性, 可能会导致算法过早收敛, 从而错失更优的解. 因此, 本算法不仅关注各层之间的信息素浓度变化, 还引入了动态的终止判断机制. 有以下两种终止条件判断的方式.

(1) 信息素分布多样性收敛

首先, 通过度量信息素分布的多样性来判断算法是否收敛. 当算法在多次迭代后, 各层间路径上的信息素浓度τ_{ij}^{l}的分布趋于稳定, 并且各层之间的信息素浓度差异减小到一定阈值以下时, 表明个体在寻找最佳调度路径的过程中已经趋向一致. 这意味着算法在当前的搜索空间中已经找到了相对最优的调度解, 从而可以终止迭代. 公式为:

$$D(\tau)=\frac{1}{L}\sum_{i=1}^{L}\frac{\sum_{(i,j)\in E_l}\left|\tau_{ij}^{l}-\bar{\tau}^{l}\right|}{|E_l|} \tag{18}$$

其中, $D(\tau)$表示所有层的信息素分布的多样性, E_l 表示层 l 中所有可行路径的集合, $|E_l|$表示层 l 中路径的数量, $\bar{\tau}^{l}$表示层 l 的平均信息素浓度. 通过计算每一层的路径信息素浓度与其平均值之间的差异, 可以度量当

前信息素分布的多样性. 当多样性趋近于阈值ε, 即:

$$D(\tau) < \varepsilon \tag{19}$$

即表明各路径的信息素浓度趋于一致, 算法已经收敛.

(2) 最大迭代次数

设定最大迭代次数为 $T_{\max}$, 当前迭代次数为 t, 则算法的终止条件可以定义如下:

$$t \geqslant T_{\max} \tag{20}$$

在蚁群算法中, 设定最大迭代次数的目的是控制计算开销, 防止算法陷入长时间的循环. 此外, 经过一定次数的迭代后, 算法通常已经接近全局最优解, 再继续迭代可能收效甚微. 通过设定最大迭代次数, 可以在保证解质量的前提下限制算法的运行时间, 尤其适用于资源受限的应用场景.

与传统蚁群算法相比, LQCS-ACO 在适应度函数中引入了层结构评估函数$g_{\text{Layer}}(x)$将量子电路中的层调度并行性与适应度函数相关联. 引入量子特性评估函数$h_{\text{Q}}(x, f_{\text{SWAP}}, f_{\text{CNOT}})$和系数$\left(\dfrac{N}{1+\omega_{\text{D}}}\right)$来区分 SWAP 门 (物理约束) 和 CNOT 门 (逻辑约束) 的代价并量化硬件开销.

在路径移动方面, 传统蚁群算法仅依赖于信息素和启发式距离, 无法适配量子电路中的分层调度. LQCS-ACO 在路径选择时, 优先选择同层的可并行门, 在启发式信息中融入层内物理距离 d_{ij} 和门类因子 δ (其中 SWAP 门的 δ 更高, 优先规避, 目的是减少 SWAP 门的开销), 这样使得 LQCS-ACO 的路径选择能够适配量子电路的分层并行架构.

LQCS-ACO 引入 Sigmoid 函数动态调整每层挥发率, 中间层挥发率低 (保留探索), 首尾层挥发率高 (加速收敛), 让信息素更新适配量子电路的分层优化, 此外 LQCS-ACO 引入层间信息素传播, 让当前层调度继承前层优化经验, 进一步减少全局 SWAP 门的开销, 提高优化效率. 而传统蚁群算法采用全局固定的挥发率, 无法适配量子电路分层优化的需求.

LQCS-ACO 算法的具体流程图如图 2 所示. LQCS-ACO 算法伪代码如算法 1 所示. LQCS-ACO 的空间复杂度为$\text{O}(N_g^2)$源于信息素矩阵存储, 时间复杂度为$\text{O}(max_iter \cdot n_{\text{ants}} \cdot N_g^2)$. 时间复杂度主要由迭代过程中蚂蚁构建的路径复杂度$\text{O}(n_{\text{ants}} \cdot N_g^2)$, 适应度函数复杂度$\text{O}(N_g^2/L)$和信息素更新的复杂度$\text{O}(N_g^2)$决定. 由此可见, 当量子门数 N_g 增大时, LQCS-ACO 的复杂度会呈平方级增长. 相较于传统蚁群算法, 其复杂度新增了量子电路层结构评估、门类型特性计算及跨层信息素传递等开销, 但通过动态权重与分层机制, 在复杂度可控的前提下, 在量子电路调度中实现了 SWAP/CNOT 门数量的显著优化. 不过在中大规模量子电路中需通过分层并行或稀疏存储优化等方法来降低计算成本.

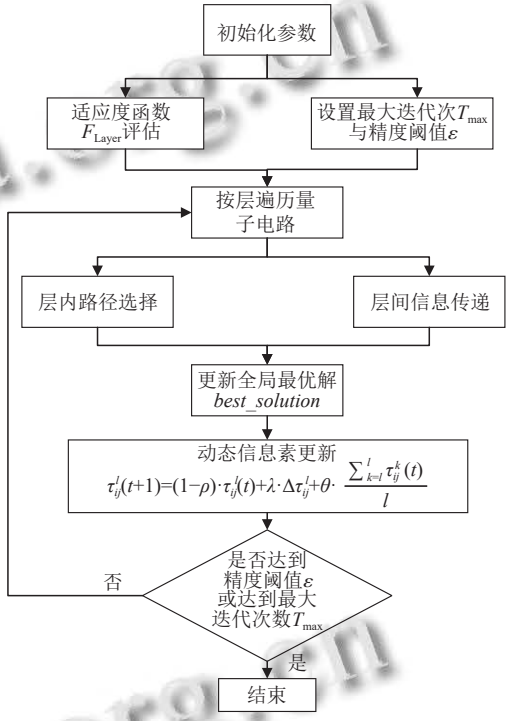


图 2 LQCS-ACO 算法流程图

算法 1. 层级量子电路调度-蚁群 (LQCS-ACO) 融合算法

输入: Q: 输入的量子电路; max_iter: 最大迭代次数; $cycle\_insts$: 当前周期的门列表; n_{ants}: 蚂蚁数量; ρ: 信息素挥发率.
输出: S*: 优化后的量子电路调度.

初始化信息素矩阵 τ 为 τ_0, 最优解 $best\_solution$ 为空集, 最优成本 $best\_cost$ 为 0

```
1. begin
2.   for iter = 1 to max_iter do:
3.     for each ant k = 1 to n_ants do:
4.       Initialize empty scheduling S_k
5.       while not all gates in Q are scheduled do:
6.         Select next executable gate g
7.         Add g to S_k
8.         Update dependency constraints
9.       Evaluate fitness of S_k using F_layer function
10.    if cost(S_k)<best_cost then:
11.      best_solution ← S_k
```

12. *best_cost* ← *cost*(*S_k*)
13. $\tau(g) \leftarrow (1-\rho) \times \tau(g) + \Delta\tau(g)$, where $\Delta\tau(g) = 1/cost(S\_k)$ for best ant path
14. Apply local search refinement on *best_solution*
15. **return** *best_solution*
16. **return** *S**
17. **end**

2.3 LQCS-ACO 与传统蚁群算法性能比较

如图 3 所示, 用 TSP 问题为测试基准 (城市数为 20, 种群数为 30, 评价指标为最短路径长度) 比较传统蚁群算法和 LQCS-ACO 的性能. LQCS-ACO 在很少的迭代次数 (约 10 次) 就找到了最优解, 而传统蚁群算法需要约 40 次迭代才能够达到与 LQCS-ACO 相同的最优解, 说明 LQCS-ACO 的适应性更强. LQCS-ACO 在第 10 次迭代前后有一次明显的跳跃寻找到最优解, 而传统蚁群算法没有明显的跳跃而是缓慢下降, 说明 LQCS-ACO 的抗局部最优能力更强.

3 实验设计与结果分析

本文选取了特定的问题实例, 并对每个实例进行了详细的实验评估, 具体方法参照了文献[34]. 对于每个实例, 本文提供得到的最优解, 重点关注插入的 SWAP 门数量和电路中的 CNOT 门数量, 较低的门数表示更好的性能. 这些指标直接影响量子电路实现的效率和实际应用, 因此是评估我们提出算法性能的重要依据. 基准测试为“FullPass”的 t|ket>编译器 (版本 0.11.0) 和 Qiskit 编译器 (版本 0.26.2, 优化级别为 3).

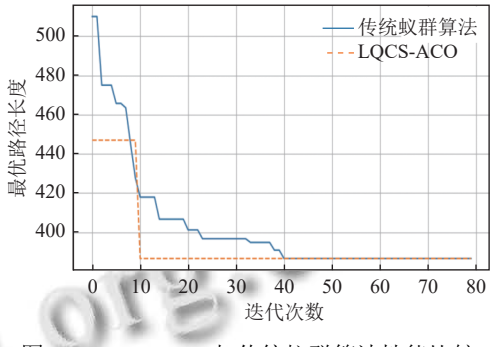


图 3 LQCS-ACO 与传统蚁群算法性能比较

LQCS-ACO 是在 Python 3.9 中实现, 执行所有编译的笔记本电脑配置为 Intel Core i7 处理器 (5.0 GHz 和 16 GB RAM).

如表 1 展示 4 种算法在基准量子电路上的比较, 相较于传统算法 IBM_QX 和 HQAA, ACO 和 LQCS-ACO 显著降低了所需附加门数. ACO 与 IBM_QX 和 HQAA 相比平均优化效果分别提升 82.42% 和 54.94%. 与 ACO 相比, LQCS-ACO 结合了层电路调度方法, 优化效果进一步提升. LQCS-ACO 与 IBM_QX、HQAA 和 ACO 相比优化效果分别提升 84.21%、64.39% 和 10.85%.

表 1 基准量子电路上不同算法优化 SWAP 门数比较

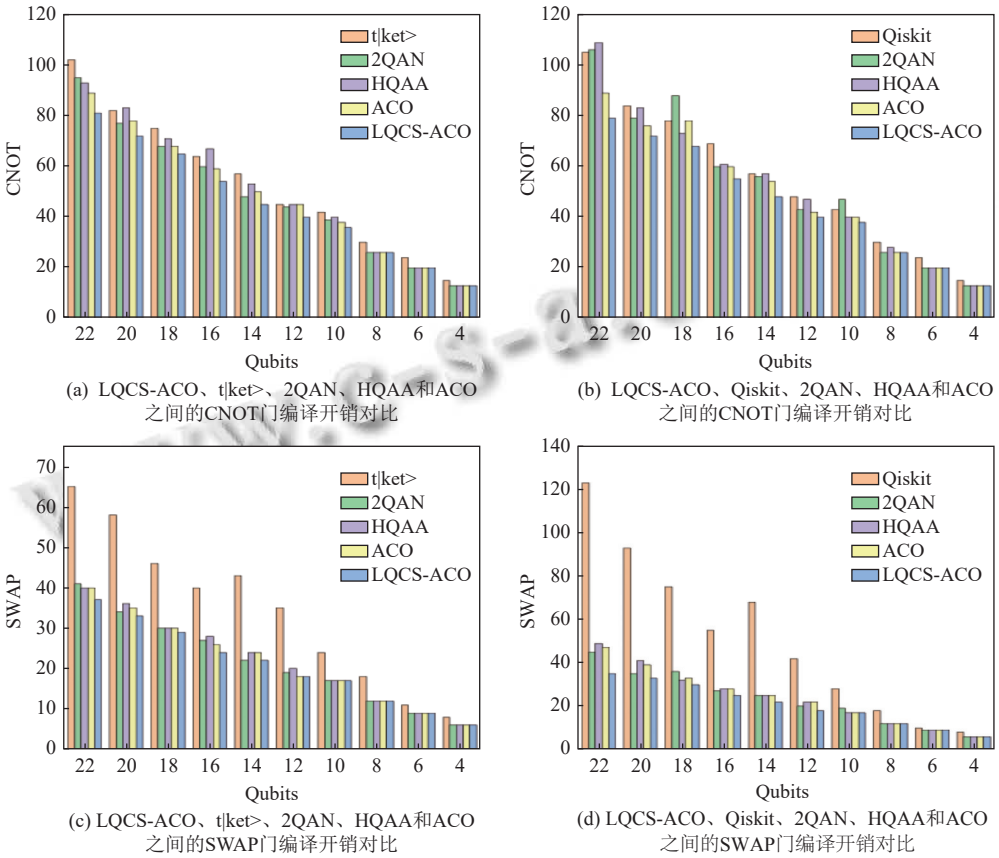
电路模型	比特数	初始门数	附加门数			
			IBM_QX	HQAA	ACO	LQCS-ACO
decod24-v2_43	4	52	19	12	3	3
4mod5-v1_22	5	21	19	13	4	4
mod5mils_65	5	35	22	16	6	6
alu-v0_27	5	36	28	16	4	4
4gt13_92	5	66	40	30	14	12
ising_model_10	10	480	22	13	4	4
qft_10	10	200	81	58	18	16
sym9_148	10	9408	4066	2853	860	754
sqn_258	10	10223	4522	3726	777	676
9symml_195	11	34881	12574	19861	9627	8642
z4_268	11	3073	1449	864	600	526
wim_266	11	427	232	185	142	127
sym9_193	11	34881	12574	24749	11113	9749
rd84_253	12	13658	5212	8321	6072	4671
cycle10_2_110	12	6050	2236	4642	806	672
sqrt8_260	12	1314	826	954	802	658
ising_model_13	13	633	66	514	174	157
radd_250	13	3213	1448	1877	828	716
qft_13	13	403	215	297	116	104
adr4_197	13	3439	1662	1754	513	456
clip_206	14	14772	6011	9232	3610	3122

表 1 基准量子电路上不同算法优化 SWAP 门数比较 (续)

电路模型	比特数	初始门数	附加门数			
			IBM_QX	HQAA	ACO	LQCS-ACO
pm1_249	14	1 776	1 224	927	553	487
sym6_316	14	270	97	167	72	63
rd84_142	15	343	220	180	95	82
misex1_241	15	4 813	1 927	2 204	1 388	1 198
square_root_7	15	7 630	2 453	4 003	1 123	942
urf6	15	171 840	74 315	41 537	7 533	6 523
co14_215	15	17 936	6 864	1 078	554	488
ising_model_16	16	786	320	458	184	124
qft_16	16	512	336	333	110	96

图 4 展示了 LQCS-ACO 相对于传统的 t|ket>和 Qiskit 编译方法的开销对比. 当 LQCS-ACO 应用于量子电路时, 在 t|ket>编译器中, 针对 4–10 量子比特的情况, SWAP 门插入数量平均减少 26.4%, CNOT 门数量平均减少 14.4%; 对于 12–22 量子比特的情况, SWAP 门插入数量平均减少 43.4%, CNOT 门数量平均减少 15.6%. 与此同时, LQCS-ACO 对比 2QAN 的 SWAP 门插入数量平均减少 3.2%, CNOT 门数量平均减少 5.9%, 对比 HQAA 的 SWAP 门插入数量平均减少 5.2%, CNOT 门数量平均减少 9.0%.

在 Qiskit 编译器中, 4–10 量子比特的 SWAP 门插入数量平均减少 26.9%, 同时 CNOT 门数量平均减少 13.7%; 12–22 量子比特的情况, SWAP 门插入数量减少 62.6%, CNOT 门数量减少 17.4%. 与此同时, LQCS-ACO 对比 2QAN 的 SWAP 门插入数量平均减少 8.5%, CNOT 门数量平均减少 10.6%, 对比 HQAA 的 SWAP 门插入数量平均减少 9.5%, CNOT 门数量平均减少 10.0%.



(a) LQCS-ACO、t|ket>、2QAN、HQAA和ACO之间的CNOT门编译开销对比

(b) LQCS-ACO、Qiskit、2QAN、HQAA和ACO之间的CNOT门编译开销对比

(c) LQCS-ACO、t|ket>、2QAN、HQAA和ACO之间的SWAP门编译开销对比

(d) LQCS-ACO、Qiskit、2QAN、HQAA和ACO之间的SWAP门编译开销对比

图 4 LQCS-ACO 的优化结果

由图 4 的实验结果对比可以看出, 在经过了 LQCS-ACO 算法的优化后, 编译开销有了一定的减少. 具体而言, LQCS-ACO 的优势在于以下几点.

(1) 调度灵活性: LQCS-ACO 结合了蚁群算法的全局搜索能力和局部调度优化策略, 能够根据量子计算机硬件的约束条件灵活地调整调度策略, 从而在不同的量子硬件平台上都能实现较优的调度方案.

(2) 自适应性强: 蚁群算法通过模拟自然界蚂蚁觅食的行为, 能够在搜索空间中进行自适应调整, 保证了量子电路调度的全局优化. LQCS-ACO 设计了适应于量子电路的适应度函数与信息素更新策略, 不断优化调度方案, 最终获得较优解.

(3) 抗局部最优: 蚁群算法的自然特点使得 LQCS-ACO 能有效避免陷入局部最优解, 从而提高全局最优解的概率, 确保调度结果的高质量.

LQCS-ACO 与传统蚁群算法 ACO 在 t|ket>和 Qiskit 编译器里的编译方法的开销对比. 实验结果显示, LQCS-ACO 对比 ACO 有更低的编译开销. 具体而言, 在 t|ket>编译器中, LQCS-ACO 对 CNOT 的优化比 ACO 提高 34.1%, 对 SWAP 的优化比 ACO 提高 5.7%; 在 Qiskit 编译器中, LQCS-ACO 对 CNOT 的优化比 ACO 提高 33.6%, 对 SWAP 的优化比 ACO 提高 8.9%.

4 总结与展望

随着量子计算技术的飞速发展, 量子电路调度作为实现高效量子计算的重要环节, 逐渐成为学术界与工业界的研究热点. 本文以基于层结构的量子电路调度算法为核心, 针对量子电路调度的核心挑战, 提出了层级量子电路调度-蚁群融合算法 LQCS-ACO. 该算法从不同角度优化了量子电路调度的性能, 显著减少了 SWAP 门数量和电路深度, 提升了并行性和硬件适配性, 为量子计算的高效执行提供了有效的解决方案.

尽管本文提出的 LQCS-ACO 算法在实验中展现出显著的性能提升, 但仍存在一些不足和局限性. 首先, 对于大规模复杂量子电路, 算法的计算复杂度会随量子比特数目和量子门数量的增加而急剧上升, 这导致了计算资源消耗过大和计算时间显著延长的问题. 其次, 虽然本文的算法在多种典型硬件拓扑和量子线路场景下展现了良好的适应性, 但在特殊硬件架构或非典型量子算法场景下, 其适应性和通用性仍有待进一步提升. 此外, 部分算法的参数调优过程依赖于人工经验, 在实际应用中可能需要针对不同的量子电路场景进行重复的调整与验证, 缺乏普适性和自动化能力.

未来的研究应进一步深入量子电路的结构特性, 研究复杂量子电路的拓扑结构、量子门之间的依赖关系以及量子比特的耦合特性, 设计更高效的调度算法. 可以探索新的图论模型或代数方法, 以更精准地描述量子线路的依赖关系, 并在此基础上开发更高效的优先级分配与调度策略. 此外, 通过引入动态更新机制和在线学习算法, 使得调度策略能够根据电路实时结构和运行状态进行灵活调整, 从而确保在不同规模和复杂度的量子电路中均能保持高效性.

参考文献

1 Chand S, Singh HK, Ray T, *et al*. Rollout based heuristics for the quantum circuit compilation problem. Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC). Wellington: IEEE, 2019. 974–981.

2 Karuppasamy K, Puram V, Johnson S, *et al*. A comprehensive review of quantum circuit optimization: Current trends and future directions. Quantum Reports, 2025, 7(1): 2. [doi: 10.3390/quantum7010002]

3 Preskill J. Quantum computing in the NISQ era and beyond. Quantum, 2018, 2: 79. [doi: 10.22331/q-2018-08-06-79]

4 Campagne-Ibarcq P, Eickbusch A, Touzard S, *et al*. Quantum error correction of a qubit encoded in grid states of an oscillator. Nature, 2020, 584(7821): 368–372. [doi: 10.1038/s41586-020-2603-3]

5 Johnson MW, Amin MHS, Gildert S, *et al*. Quantum annealing with manufactured spins. Nature, 2011, 473(7346): 194–198. [doi: 10.1038/nature10012]

6 Devitt SJ. Performing quantum computing experiments in the cloud. Physical Review A, 2016, 94(3): 032329. [doi: 10.1103/PhysRevA.94.032329]

7 Barends R, Shabani A, Lamata L, *et al*. Digitized adiabatic quantum computing with a superconducting circuit. Nature, 2016, 534(7606): 222–226. [doi: 10.1038/nature17658]

8 Versluis R, Poletto S, Khammassi N, *et al*. Scalable quantum circuit and control for a superconducting surface code. Physical Review Applied, 2017, 8(3): 034021. [doi: 10.1103/PhysRevApplied.8.034021]

9 Sete EA, Zeng WJ, Rigetti CT. A functional architecture for scalable quantum computing. Proceedings of the 2016 IEEE International Conference on Rebooting Computing (ICRC). San Diego: IEEE, 2016. 1–6.

10 李晖, 卢凯, 韩子傲, 等. NISQ 设备的量子电路调度策略优

化研究. 计算机工程与应用, 2024, 60(22): 105–113. [doi: 10.3778/j.issn.1002-8331.2401-0224]
11 Bhoumik D, Majumdar R, Saha A, *et al*. Distributed scheduling of quantum circuits with noise and time optimization. arXiv:2309.06005, 2023.
12 Romero-Alvarez J, Alvarado-Valiente J, Casco-Seco J, *et al*. Quantum circuit scheduler for QPUs usage optimization. arXiv:2404.01055, 2024.
13 Guerreschi GG. Scheduler of quantum circuits based on dynamical pattern improvement and its application to hardware design. arXiv:1912.00035, 2019.
14 Bahreini T, Mohammadzadeh N. An MINLP model for scheduling and placement of quantum circuits with a heuristic solution approach. ACM Journal on Emerging Technologies in Computing Systems (JETC), 2015, 12(3): 1–20.
15 Venturelli D, Do M, Rieffel E, *et al*. Compiling quantum circuits to realistic hardware architectures using temporal planners. Quantum Science and Technology, 2018, 3(2): 025004. [doi: 10.1088/2058-9565/aaa331]
16 Whitney M, Isailovic N, Patel Y, *et al*. Automated generation of layout and control for quantum circuits. Proceedings of the 4th International Conference on Computing Frontiers. Ischia: ACM, 2007. 83–94.
17 Booth K, Do M, Beck J, *et al*. Comparing and integrating constraint programming and temporal planning for quantum circuit compilation. Proceedings of the 28th International Conference on Automated Planning and Scheduling. Delft: Delft University of Technology, 2018. 366–374.
18 Gong LH, Pei JJ, Zhang TF, *et al*. Quantum convolutional neural network based on variational quantum circuits. Optics Communications, 2024, 550: 129993. [doi: 10.1016/j.optcom.2023.129993]
19 Ruiz FJR, Laakkonen T, Bausch J, *et al*. Quantum circuit optimization with alphatensor. Nature Machine Intelligence, 2025, 7(3): 374–385. [doi: 10.1038/s42256-025-01001-1]
20 Li ZK, Peng JJ, Mei YX, *et al*. Quarl: A learning-based quantum circuit optimizer. Proceedings of the ACM on Programming Languages, 2024, 8(OOPSLA1): 555–582. [doi: 10.1145/3649831]
21 Li YZ, Liu W, Xu GS, *et al*. Quantum circuit mapping based on discrete particle swarm optimization and deep reinforcement learning. Swarm and Evolutionary Computation, 2025, 95: 101923. [doi: 10.1016/j.swevo.2025.101923]
22 Itoko T, Imamichi T. Scheduling of operations in quantum compiler. Proceedings of the 2020 IEEE International Conference on Quantum Computing and Engineering (QCE). Denver: IEEE, 2020. 337–344.
23 Bhoumik D, Majumdar R, Sur-Kolay S. Resource-aware scheduling of multiple quantum circuits on a hardware device. arXiv:2407.08930, 2024.
24 Kurowski K, Pecyna T, Slysz M, *et al*. Application of quantum approximate optimization algorithm to job shop scheduling problem. European Journal of Operational Research, 2023, 310(2): 518–528. [doi: 10.1016/j.ejor.2023.03.013]
25 Metodi TS, Thaker DD, Cross AW, *et al*. Scheduling physical operations in a quantum information processor. Proceedings of the 2006 Quantum Information and Computation IV. Orlando: SPIE, 2006, 6244. 210–221.
26 Kan SW, Du ZF, Palma M, *et al*. Scalable circuit cutting and scheduling in a resource-constrained and distributed quantum system. arXiv:2405.04514, 2024.
27 Gokhale P, Baker JM, Duckering C, *et al*. Asymptotic improvements to quantum circuits via qutrits. Proceedings of the 46th International Symposium on Computer Architecture. Phoenix: ACM, 2019. 554–566.
28 Dorigo M, Birattari M, Stutzle T. Ant colony optimization. IEEE Computational Intelligence Magazine, 2006, 1(4): 28–39. [doi: 10.1109/MCI.2006.329691]
29 Rokbani N, Kumar R, Abraham A, *et al*. Bi-heuristic ant colony optimization-based approaches for traveling salesman problem. Soft Computing, 2021, 25(5): 3775–3794. [doi: 10.1007/s00500-020-05406-5]
30 Nayar N, Gautam S, Singh P, *et al*. Ant colony optimization: A review of literature and application in feature selection. Proceedings of the 2020 Conference on Inventive Computation and Information Technologies. Singapore: Springer, 2021. 285–297.
31 Wang Y, Han ZP. Ant colony optimization for traveling salesman problem based on parameters optimization. Applied Soft Computing, 2021, 107: 107439. [doi: 10.1016/j.asoc.2021.107439]
32 Xie XW, Tang ZH, Cai JJ. The multi-objective inspection path-planning in radioactive environment based on an improved ant colony optimization algorithm. Progress in Nuclear Energy, 2022, 144: 104076. [doi: 10.1016/j.pnucene.2021.104076]
33 Zhou XB, Ma HJ, Gu JG, *et al*. Parameter adaptation-based ant colony optimization with dynamic hybrid mechanism. Engineering Applications of Artificial Intelligence, 2022, 114: 105139. [doi: 10.1016/j.engappai.2022.105139]
34 Lao LL, Browne DE. 2QAN: A quantum compiler for 2-local qubit hamiltonian simulation algorithms. Proceedings of the 49th Annual International Symposium on Computer Architecture. New York: ACM, 2022. 351–365.

(校对责编: 张重毅)