软件学报 ISSN 1000-9825, CODEN RUXUEW
2025,36(8):3431−3443 [doi: 10.13328/j.cnki.jos.007343] [CSTR: 32375.14.jos.007343]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

Fast-USYN: 从酉矩阵到高质量量子电路的快速合成*

谭思危, 卢丽强, 郎聪亮, 陈明帅, 尹建伟

(浙江大学 计算机科学与技术学院, 浙江 杭州 310027)

通信作者: 卢丽强, E-mail: liqianglu@zju. edu. cn; 尹建伟, E-mail: zjuyjw@zju.edu.cn

摘　要: 当前的量子程序一般由量子电路表示, 由多个量子门组成. 如果程序包含了被直接表示为酉矩阵的门, 需要将这些量子门转化为基本门所构成的量子电路. 该步骤被称为量子电路合成. 然而, 当前的合成方法可能会生成包含数千个门的量子电路. 这些量子电路的质量较低, 在部署到真实含噪声的量子硬件时非常容易输出错误的结果. 此外, 在保证门数量较小的情况下, 当量子比特数量增至 8 时, 量子电路合成需要数周甚至数月的时间. 在这项工作中, 提出一种量子电路合成方法, 实现从酉矩阵到高质量量子电路的快速合成. 首先介绍一种迭代方法, 通过插入电路模块来逼近目标酉矩阵. 在迭代中, 提出一种具有奖励机制的前瞻策略减少冗余量子门. 在量子电路合成的加速过程中, 为了减少候选电路模块的空间, 提出一种剪枝方法, 首先描述每个候选电路模块的闭包来刻画电路的表示空间, 然后基于模块的表示空间重叠率进行剪枝, 以此构建一个小而高质量的候选集合. 此外, 为了减少搜索最优门参数的开销, 将选定的候选与目标酉矩阵打包成统一电路, 然后通过计算其在基态上的期望来快速获得近似距离. 实验证明, 与当前的最优的量子电路合成方法 QuCT 和 QFAST 相比, 该方法在 5–8 量子比特量子电路合成中实现了减少门数量为原有方法的 37.04%–62.50%, 同时实现 3.7–20.6 倍的加速.

关键词: 量子计算; 量子软件; 量子程序; 编译; 程序合成

中图法分类号: TP311

中文引用格式: 谭思危, 卢丽强, 郎聪亮, 陈明帅, 尹建伟. Fast-USYN: 从酉矩阵到高质量量子电路的快速合成 . 软件学报, 2025, 36(8): 3431–3443. http://www.jos.org.cn/1000-9825/7343.htm

英文引用格式: Tan SW, Lu LQ, Lang CL, Chen MS, Yin JW. Fast-USYN: Fast Synthesis from Unitary Matrices to High-quality Quantum Circuits. Ruan Jian Xue Bao/Journal of Software, 2025, 36(8): 3431–3443 (in Chinese). http://www.jos.org.cn/1000-9825/7343.htm

Fast-USYN: Fast Synthesis from Unitary Matrices to High-quality Quantum Circuits

TAN Si-Wei, LU Li-Qiang, LANG Cong-Liang, CHEN Ming-Shuai, YIN Jian-Wei

(College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China)

Abstract: Current quantum programs are generally represented by quantum circuits, including various quantum gates. If the program contains gates that are directly represented as unitary matrices, these gates need to be transformed into quantum circuits composed of basic gates. This step is called quantum circuit synthesis. However, current synthesis methods may generate circuits with thousands of gates. The quality of these quantum circuits is low and they are very likely to output incorrect results when deployed to real noisy quantum hardware. When the number of qubits is increased to 8 while ensuring a small number of gates, the quantum circuit synthesis takes weeks or even months. This study proposes a quantum circuit synthesis method, realizing the fast synthesis from unitary matrices to high-quality quantum circuits. Firstly, an iterative method is introduced to approximate the target unitary matrix by inserting circuit modules. During the iteration, a look-ahead strategy with a reward mechanism is proposed to reduce redundant quantum gates. In the acceleration process of quantum circuit synthesis, the study proposes a pruning method to reduce the space of candidate circuit modules. The method first

* 基金项目: 国家重点研发计划 (2023YFF0905200); 中央高校基本科研业务费专项资金 (226-2024-00051, 226-2024-00140); 浙江尖兵项目 (2023C01036)
本文由“形式化方法与应用”专题特约编辑陈明帅研究员、田聪教授、熊英飞副教授推荐.
收稿时间: 2024-08-21; 修改时间: 2024-10-14; 采用时间: 2024-11-26; jos 在线出版时间: 2024-12-10
CNKI 网络首发时间: 2025-04-21

describes the closure of each candidate circuit module to characterize the representation space of the circuit, and then prunes based on the overlap rate of the representation spaces of the modules, thus constructing a small and high-quality candidate set. Furthermore, to reduce the overhead of searching for optimal gate parameters, this study packs the selected candidates with the target unitary into a uniform circuit so that we can quickly obtain the approximation distance by calculating its expectation on the ground state. Experiments show that, compared with the current optimal quantum circuit synthesis methods QuCT and QFAST, this study reduces the number of gates to 37.0%–62.5%, and achieve a 3.7–20.6 times acceleration in the 5 to 8 qubit quantum circuit synthesis.

Key words: quantum computing; quantum software; quantum circuit; compiler; program synthesis

量子程序的部署包含从高级量子语言到可执行量子指令的多个优化阶段. 该过程从语言描述[1,2]开始, 经历了比特布局[3–5]、脉冲合成[6]等环节. 当前量子程序通过量子电路描述量子比特 (量子寄存器) 的操作. 量子电路由一系列量子门组成, 这些量子门按照一定顺序操作特定的量子比特. 然而量子电路合成具有最高的延迟, 可能引入数千个门. 在传统的编程语言领域, 程序合成 (program synthesis)[7]是一种用于自动生成满足特定功能需求的程序的重要技术. 例如, 胡星等人利用深度学习生成程序. 同时也存在基于演绎推理的 SQL 语句[8,9]以及数据结构[10]的自动合成方法. 这些程序合成方法能帮助编程者大幅降低编程难度. 同样的, 量子电路合成也是辅助量子编程的重要工具[11]. 量子电路合成阶段的输入是一个目标酉矩阵, 合成旨在搜索找到一个由基本门组成的量子电路, 该量子电路在数学上和目标酉矩阵相等.

在当前的中等规模含噪量子时代中, 门容易受到噪声的影响. 因此, 目前的合成过程都需要尽可能地减少量子电路中门的数量[12]. 例如在部署中量子比特映射阶段, 通过多因素成本优化方法降低量子比特映射中实现映射变化的交换门数量[4]. 谢磊等人[13]对这一系列的优化方法进行了系统性的描述. 在合成电路的过程中, 由于量子门的各种组合表示的空间随着比特数和电路深度指数增长, 且空间中门的数量分布尤其不均匀, 低效的合成可能会导致指数级别增长的门, 因此在量子电路合成中减少量子电路的门数量也变得更加重要[14].

量子电路合成已经被广泛应用于当前的面向量子语言的编译器中, 如 Qiskit[15]和 Cirq[16]. 这些编译器中的电路合成过程采用了基于数学模板的策略[17,18], 可以在 100 s 以内的时间找到酉矩阵的量子电路表示. 然而, 这些方法可能会生成过多的量子门, 在当前含噪量子时代, 这会导致量子程序在执行时产生大量噪声, 从而导致执行失败. 具体来说, 对于超过 5 量子比特的酉矩阵, Qiskit 和 Cirq 生成的量子电路包含了数千个门. 例如, Qiskit 默认使用的是按列分解方法 (CCD)[17], 其将酉矩阵分解成多个小酉矩阵, CCD 在合成 6 量子比特的量子傅里叶电路时会产生超过 9000 个门, 然而最优的量子傅里叶电路只需要 81 个门.

近期, 国际上提出了一系列旨在优化量子电路数量的量子电路合成方法, 例如 2021 年提出的 QFAST[3]和 2023 年提出的 QuCT[19]. 这些方法都采用迭代搜索来逼近目标酉矩阵, 每次迭代的时候, 它们都会从候选空间中选择插入一个候选门或者一个候选的电路模块. 然而, 它们依赖于重复的搜索, 导致了较高时间复杂度. 例如, QFAST[3]需要超过 6 个月的时间来合成 6 比特酉矩阵的量子电路. Kang 等人[11]和 Paradis 等人[20]则提出了利用模块化的量子电路来组成合成电路的思想, 但是由于这些方法使用的电路模块是人工决定了, 该方法只适用于特定酉矩阵. 此外, 这些方法仍然无法实现最少的门数量. 例如, QuCT[19]合成 6 量子比特的量子傅里叶酉电路需要 283 个门, 相对于最优结果增加了 3.5 倍的门数量. Kang 等人的方法则在合成 6 量子比特的随机酉矩阵的时候产生了超过 11000 个门, 甚至比基于数学模板的策略[17,18]的方法更低效.

根据本研究的观察发现, 这些方法存在的高合成延迟或是低质量缺陷本质上来源于 3 个限制.

(1) 这些方法在搜索过程中容易陷入局部最优解. 当前的方法采用了贪心算法[3]或者临近搜索[20]来寻找量子电路, 这导致了大量冗余的门的产生. 在实现上, 这些方法面临搜索时间和结果质量的权衡. 如果搜索时间越少, 则找到高质量的量子电路的机会就越小. 但是如果想找到高质量的量子电路, 一般又需要非常巨大的搜索的时间. 总体来说. 为了在搜索中平衡效率, 这方法都采用了贪心策略. 在每次迭代中插入门时, 它们只会选择使量子电路更快逼近酉矩阵的门插入量子电路中, 而不考虑这可能会导致局部最优. 从而导致比最优结果更多的门.

(2) 这些方法在搜索空间中是庞大且重叠的. 当前方法的搜索的候选空间包括量子门在不同比特和位置上的量子模块组合. 先前的研究, 如 QFAST[3]和 QSEARCH[21]会穷尽所有可能的候选电路模块, 一一测试其是否适合被

插入合成电路中. 对于大于 6 比特的量子电路合成任务, 每次迭代都要搜索上百个候选. QuCT[19]采用了数据驱动模型对空间进行剪枝. 然而, 本研究发现, 超过 90% 的候选电路模块在它们能表示的酉矩阵空间中存在高度重叠, 导致冗余搜索, 从而产生较高的合成延迟. Kang 等人[11]则采用了人工选择的候选集合, 同样会产生表示能力的重叠. 总体而言, 目前的方法缺乏对这些候选电路模块的表示能力的表征和剪枝的方法.

(3) 这些方法在参数优化中承受高计算成本. 基于搜索的合成方法在每次迭代中, 需要通过一个参数优化过程为每个门找到参数, 以确定量子电路和酉矩阵之间的距离. 该优化过程占据了合成大部分的时间, 这是因为它依赖于迭代地计算电路的酉矩阵, 然后计算电路与目标酉矩阵之间的距离, 最后通过梯度更新量子门的参数. 例如, 对于 8 量子比特的量子电路合成, QFAST 的每次迭代中需要超过 10 h 进行参数优化. 总的来说, 参数优化占用了 QFAST 搜索中超过 99% 的时间.

本研究提出了 Fast-USYN, 它既实现了较少的门数量, 又具有较低的合成延迟. Fast-USYN 采用了一种基于前瞻的搜索策略, 通过插入电路模块来逼近目标酉矩阵. 相较于之前工作的改进如图 1 所示. 为了避免局部最优解, Fast-USYN 采用了一种前瞻策略, 每次迭代中会搜索一定深度内的最优电路. 本文还采用了一种奖励估计机制来评估不同合成阶段中选择不同候选电路带来的增益. 其次, 为了剪枝搜索空间, 我们为每个候选电路模块构建了一个闭包以表征其表示空间. 然后基于此特征构建了一个高质量的候选集合. 另外, 本文通过消除参数优化中的电路酉矩阵计算来加速搜索. 将电路和目标酉矩阵的距离计算构造为希尔伯特-施密特测试电路的期望计算来比较它们的相似度. 本研究证明了可以通过在基态上最大化该电路期望来优化门参数. 相较于先前的方法, 这个构造方式能够为量子电路合成的参数优化带来指数级别的复杂度提升. 总体而言, 本文的主要贡献总结如下.

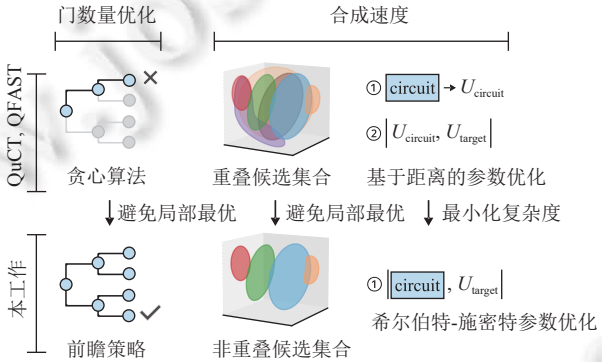


图 1 Fast-USYN 与之前的合成方法相比的改进

1) 提出 Fast-USYN, 该方法通过前瞻策略避免合成中的局部最优解, 实现了从酉矩阵到电路的门数量的优化.

2) 提出一种基于闭包的候选剪枝方法. 这是一种对各种参数化的电路模块的表示空间进行表征的方法, 利用该方法, Fast-USYN 在合成中实现搜索空间减少 98% 以上.

3) 提出一种快速参数优化方法, 通过将问题转化为希尔伯特-施密特测试电路的模拟, 实现了超过 8 倍的端到端加速.

实验证明, 与最先进的方法 QuCT 相比, Fast-USYN 在 5–8 量子比特量子电路合成中实现了减少门数量为原有方法的 37.04%–62.50% 和 3.7–20.6 倍的加速. Fast-USYN 被集成在 https://github.com/JanusQ/JanusQ 项目中.

1 背景知识

1.1 量子电路

量子电路是一种广泛使用的量子程序设计模型. 电路由顺序排列的量子门组成, 这些门在一组量子比特上操作. 基本门集合是指可以被特定量子器件执行的门, 由硬件实现决定. 例如, 谷歌的悬铃木量子计算机的基本门集合包括 $\sqrt{X}$、Y、$\sqrt{W}$ 和 FSim 门. 本文采用单量子比特旋转 U 门和 CRZ 门作为基本门集合, 介绍了 Fast-USYN

方法. *CRZ* 型门有 1 个参数, *U* 门有 3 个参数. Fast-USYN 也可以应用于其他基本的门集合.

在数学上, 量子门可以被表示为一个酉矩阵. 酉矩阵的数值由其操作类型 (如 *U* 门和 *CRZ* 门) 和参数 (如 *CRZ* 门的参数可以是 0.25π). 一个量子电路可以被表示为其中的量子门的酉的张量积 ($\otimes$) 和乘法. 例如, 图 2 展示了一个 *U* 门和 *CRZ* 门的电路, 其中 $U(\varphi,\theta,\omega)$ 和 $CRZ(\varphi)$ 分别是生成 *U* 门和 *CRZ* 门的酉矩阵的函数.

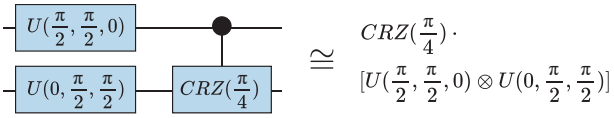


图 2 量子电路和其对应的酉矩阵的例子

1.2 量子电路合成

当前量子程序由量子电路表示. 量子电路合成 (quantum circuit synthesis), 也指酉矩阵合成 (unitary synthesis) 和酉矩阵分解 (unitary decomposition). 该任务以一个目标酉矩阵为输入, 搜索与该酉矩阵相等的量子电路. 该量子电路只能由基本的门组成. 早期的方法, 如 QSD[18]和 CCD[17], 遵循 cosin-sine 分解[22]等数学公式将一个酉矩阵递归分解为一系列较小的酉矩阵, 最后合成量子门. 最近的方法, 如 QuCT[19]和 QFAST[3], 采用基于搜索的方法, 迭代地搜索适合的门后电路模块, 将其插入到电路的末端, 直到量子电路和目标酉矩阵之间的距离缩小到阈值.

矩阵平方距离 (matrix squared distance, *MSD*) 在之前的量子电路合成工作[3,19]中被广泛用于比较目标酉矩阵和电路酉矩阵的距离. 假设目标酉矩阵表示为 U_{target}, 量子电路酉矩阵表示为 U_{circuit}, *MSD* 的计算公式为:

$$MSD = 1 - \frac{\left|\text{tr}\left(U_{\text{target}}^{\dagger} U_{\text{circuit}}\right)\right|}{D} \tag{1}$$

其中, tr 和 † 分别为矩阵的迹 (trace) 和共轭转置 (conjugate transpose) 运算. 具体来说, 迹是矩阵的对角矩阵中元素值的和, 共轭转置是对矩阵中的元素取共轭复数后进行转置. 对于酉矩阵来说, 其共轭转置也等于矩阵的逆. *D* 是矩阵的维数. 如果两个矩阵相等, 则距离为 0. *MSD* 对矩阵的全局相位不敏感, 而全局相位对程序的计算无影响, 因此可以更好地分辨酉矩阵的相似度, 是一个常用的表示矩阵距离的度量单位.

2 Fast-USYN 工作流

Fast-USYN 采用了迭代搜索的方式合成量子电路. 图 3 展示了其搜索流程. 在每次迭代中, Fast-USYN 会执行步骤 (a) 到 (b), 主要采用了前瞻搜索以减少陷入局部最优的概率. Fast-USYN 从一个初始的空电路开始执行, 1 次迭代包含 3 个步骤 ((a)–(c)), 步骤 (a) 包含 3 个子步骤 ((a-1)–(a-3)).

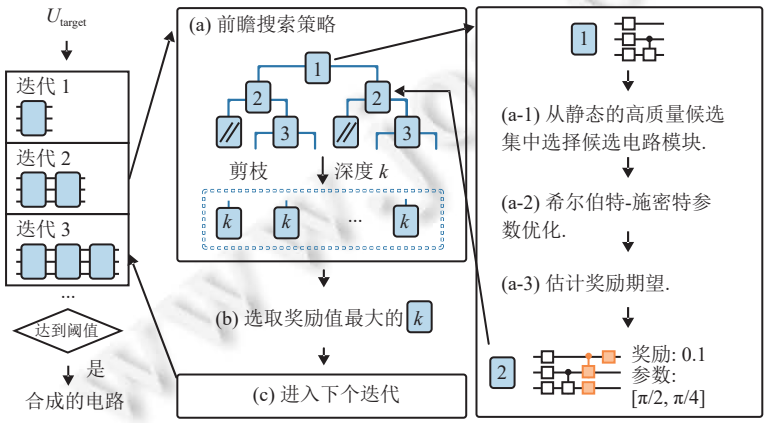


图 3 Fast-USYN 的搜索流程

步骤 (a). 从当前电路开始执行指定深度的树搜索 (对应图 3 中的 k). 在树搜索的每个节点 (例如图 3 中的节点 1, 2, 3) 中, 一个候选电路模块会被插入到电路中. 插入操作包括以下 3 个子步骤.

步骤 (a-1). 从预先生成的候选集合中选择一个候选电路模块, 将其插入电路, 候选集的构建在第 4 节中介绍.

步骤 (a-2). 执行基于希尔伯特-施密特电路的参数优化以找到当前电路的参数, 具体优化方法参见第 5 节.

步骤 (a-3). 计算奖励, 以衡量插入的候选电路模块随距离递减的贡献.

步骤 (b). 从搜索树的叶子节点中选择奖励最大的节点.

步骤 (c). 使用选中节点的电路进行下一次迭代. 检查电路与目标酉矩阵的距离是否小于阈值. 如果距离在阈值内, 则终止搜索并输出电路.

流程的总体算法如算法 1 所示.

算法 1. 量子电路合成.

输入: 酉矩阵 U, 停止阈值 l, 前瞻搜索深度 k;

输出: 合成量子电路 *circuit*.

1. *circuit* ⇐ 空量子电路
2. **FOR** $\left|U_{\text{circuit}}, U_{\text{target}}\right| > l$ **DO**
3. **FOR** *depth*=1, *depth*++, *depth*≤k **DO**
4. **FOR** $c \in$ 高质量候选集 **DO**
5. *circuit*′ ⇐ *circuit*′ + c
6. 基于希尔伯特-斯密特参数优化 *circuit*′ 量子门参数
7. *circuit*′.*reward* ⇐ *estimate*(*circuit*′, U_{target})
8. **END FOR**
9. **END FOR**
10. *circuit* ⇐ *circuit*+ 具有最高 *reward* 的 *circuit*′
11. **END FOR**

● 候选集合. 在 Fast-USYN 中, 候选电路被定义为由一系列参数化的基本门组成的子电路. 这样的子电路被称为量子电路模块. 候选电路模块的最大、最小门数量和量子比特数是可配置的, 由期望的搜索粒度决定. Fast-USYN 对候选集合的元素的定义与之前的工作存在差异, 如 QFAST[3]和 QSEARCH[21]采用具有较少量子比特酉矩阵门或单个量子门作为候选集合的元素. 使用单个量子门会导致较低的近似效率, 从而产生更多的迭代次数. 而使用小的酉矩阵门则会带来额外分解这些酉矩阵的开销.

● 奖励估计. 本文提出一种奖励估计技术, 用于指导步骤 (a) 中的树搜索剪枝和步骤 (b) 中的候选电路模块选择. 对于一个候选电路模块, 其被插入产生的奖励被表示为:

$$reward = \frac{MSD_{\text{before}} - MSD_{\text{after}}}{2^{G_{\text{circuit}}} G_{\text{candidate}}} \tag{2}$$

其中, MSD_{before} 和 MSD_{after} 分别是插入候选电路模块前后的距离. G_{circuit} 和 $G_{\text{candidate}}$ 是电路和候选的门数. 惩罚项 $2^{G_{\text{circuit}}} G_{\text{candidate}}$ 旨在表征在合成的不同阶段中电路逼近酉矩阵的边际难度, 即平衡在分解不同阶段, 插入不同大小候选量子电路模块带来的距离减少预期. 具体来说: 1) 距离的优化空间随当前电路中的量子门数量增加而指数递减. 换句话说, 随着电路深度的增加, 插入一个候选电路模块能带来的逼近效益是指数衰减的, 这是因为通过实验观察发现, 合成量子电路和酉矩阵之间的距离随着深度增长逐渐收敛, 需要加入惩罚值 $2^{G_{\text{circuit}}}$ 平衡不同深度电路下的奖励情况; 2) 对于有更多门的候选电路模块, 它能有更大的概率逼近目标酉矩阵, 但也会导致了更多的门, 因此需要增加惩罚项 $G_{\text{candidate}}$ 来要求搜索尽量选择较小的候选电路模块. 总的来说, 惩罚保证了搜索不会总是选择具有更多门的电路来快速完成合成.

● 剪枝策略. Fast-USYN 采用两种剪枝策略减少步骤 (a) 中的前瞻搜索的次数 (对应图 3 中表示为//的节点). 第 1 种策略是当候选电路模块的奖励小于阈值 α 时, 从该候选电路模块开始的进一步搜索将被取消. 第 2 种策略是当需要探索的节点数量超过阈值时, Fast-USYN 只探索具有 top-l 奖励值的候选节点, 其中 l 由 CPU 的核心数量决定, 以确保较高的 CPU 利用率.

3 基于闭包的高质量候选集合构建

候选量子电路模块作为参数化电路, 可以通过修改门的参数表示不同的酉矩阵, 本研究观察到候选量子电路模块可以表示的酉矩阵空间中表现出高度重叠. 换句话说, 许多候选电路模块在搜索逼近中是可以替换的. 之前的工作如 QuCT[19]和 QFAST[3]对整个候选空间进行穷举搜索, 导致高冗余计算. 此外, 这些候选电路模块的表示空间虽然重叠, 但是其包含的量子门数量不同. 因此选择更小的电路模块能够提高最终电路的质量, 而如采用冗余且低质量的候选集合, 则可能会选择包含较多量子门的电路模块. 为了避免这些情况, 需要表征参数化电路的表示能力, 然后构建高质量、无重叠的候选集合.

具体来说, Fast-USYN 旨在通过刻画候选量子电路模块在酉矩阵空间中表示能力, 发现电路模块表示空间的重叠性, 修剪候选空间, 最后构建一个非重叠候选集. 因为候选空间被修剪, 新的候选集可以实现更快的合成. 同时, 因为低质量且能被替代的候选电路模块不会被包括在候选集中, Fast-USYN 的合成也能产生更少的门. 图 4 展示了候选集构建的 4 步工作流程. 第 1 步对应图 4(a), 配置电路模块的最大和最小量子比特数和门数量. 第 2 步对应图 4(b), 根据量子比特数和门数量范围配置枚举候选电路模块. 第 3 步对应图 4(c), 对每个电路模块进行参数采样, 并计算这些参数下的电路酉矩阵, 形成电路模块在酉矩阵空间的点云. 每个酉矩阵可以表示为图 4(c) 中的酉矩阵空间中的一个点. 第 4 步对应图 4(d), 基于采样得到的酉矩阵计算每个电路模块的闭包, 该闭包包含了所有电路模块采样得到的酉矩阵, 代表该电路模块可以表示的酉矩阵空间. 第 5 步对应图 4(e), 根据电路模块的闭包构造高质量的候选集合. 选择的目标是在保证酉矩阵空间的高覆盖率的同时, 减少所选候选电路模块的闭包的重叠区域, 并最小后候选模块总的量子门数量.

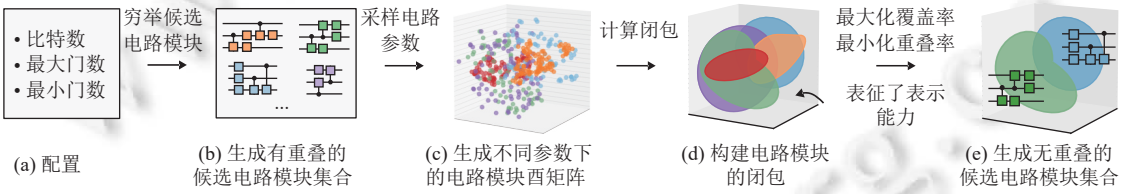


图 4 构建表示空间无重叠的候选集合的过程

基于该目标, 选择非重叠候选集 $NOCS$ 的元素过程可以被建模为一个约束组合问题的求解:

$$\begin{cases} \min\limits_{NOCS} \sum\limits_{c_i,c_j\in NOCS} OverlapArea(c_i,c_j) + \sum\limits_{c\in NOCS} NumGate(c) \\ \text{s.t.} \quad \sum\limits_{c\in NOCS} Area(c) \geqslant \gamma \end{cases} \tag{3}$$

其中, $OverlapArea(c_i,c_j)$ 表示候选电路模块 c_i, c_j 的闭包的重叠区域. $NumGate(c)$ 和 $Area(c)$ 分别是候选电路模块 c 的量子门数和闭包面积. 最小化重叠区域的目的是减少候选电路模块的数量, 最小化量子门的数量的目的是找到高质量的候选. 本研究添加了约束 $\sum\limits_{c\in NOCS} Area(c) \geqslant \gamma$ 以确保高覆盖率. 值得注意的是, 覆盖率是没有必要为 100% 的, 因为候选电路模块的组合仍然可以表示整个酉矩阵空间, 这在实验中得到了评估. 公式 (3) 可以通过 SMT 求解器或者遗传算法进行求解.

多比特酉矩阵对应的合成候选电路模块具有更高度重叠度, 这也意味着剪枝的优化空间会更大. 表 1 给出了剪枝前后的候选集合的大小, 以及每个候选电路模块的平均门数. 在剪枝过程中, 枚举候选电路模块时的门数量在 10–50. 覆盖阈值 γ 被设置为 80%. 剪枝后, 对于 4 量子比特到 6 量子比特的电路合成, 候选数量减少了 98% 以

上. 此外, 剪枝也使候选电路模块的门数量更少, 从而容易产生更高质量的电路, 如对于 4 量子比特的合成, 每个候选电路模块的平均门数从 18.4 减少到 14.2, 减少了 23.1%.

表 1 剪枝前后的候选集对比

量子比特	剪枝前		剪枝后			
	候选数	门数	候选数	门数	覆盖率 (%)	剪枝率 (%)
4	1042	18.4	19	14.2	84.3	98.18
5	1452	21.0	27	18.0	87.5	98.14
6	1963	27.2	35	22.9	82.8	98.22

● 收敛性保证. 基于 Solovay-Kitaev 定理[23]可知, *U* 门和 *CRZ* 门可以实现通用量子计算 (universal quantum computing), 即可以使用该门集合逼近任意酉矩阵. 因此, 只需要证明 Fast-USYN 的候选电路集合和 *U* 门和 *CRZ* 门的表示空间等价, 就可以保证 Fast-USYN 的收敛性. 以下为等价性的推理:

1) Fast-USYN 所用的候选集合由任意角度单比特旋转门 *U* 和 *CRZ* 门组成. 当这两种门的参数为 0 时, 它们的酉矩阵都为单位矩阵, 即等价于不操作门.

2) 对于每一个比特或两比特组合, 只需要存在一个候选组合包含其对应的 *U* 门或 *CRZ* 门, 则这个候选集合就可以通过设置其他门参数为 0 表示这个 *U* 门或 *CRZ* 门.

3) 步骤 2) 中的要求是易实现的. 因为 Fast-USYN 在候选集构造的约束组合问题中, 要求候选量子电路的覆盖率大于一个较高的阈值. 同时初始候选电路构造包含所有的量子门组合.

综上所述, Fast-USYN 的候选电路集合是可首选的.

4 希尔伯特-斯密特参数优化

本研究所使用的候选量子电路由参数化的门组成, 所以需要为每个候选电路搜索距离目标矩阵最近的门参数. 目前参数搜索通常采用的是随机梯度下降法 (stochastic gradient descent). 其参数求解过程包括: 1) 初始化一个随机参数; 2) 计算量子电路在当前参数下的酉矩阵; 3) 计算电路与目标酉之间的 *MSD*; 4) 根据 *MSD* 计算参数的梯度, 更新参数, 然后重复第 2)、3) 和 4) 步. 在 Fast-USYN 中, 本研究通过将第 2) 和 3) 步打包到希尔伯特-施密特测试电路中, 以加快距离计算和参数更新. 打包的希尔伯特-施密特测试电路在基态上的期望和 *MSD* 线性相关, 而期望计算相较于计算 *MSD* 具有较低的复杂度.

图 5 展示了用于比较合成电路和目标酉矩阵相似度的希尔伯特-施密特测试电路, 它由个量子比特组成. 电路和目标酉矩阵的共轭 (以 $U^{\dagger}$ 表示) 应用于电路中间的每个量子比特. 从数学上讲, 电路得到全 0 输出的概率等于:

$$P(00\ldots0)=\frac{1}{D^2}\left|\mathrm{tr}\left(U_{\mathrm{circuit}}^{\dagger}U_{\mathrm{target}}\right)\right|^2 \tag{4}$$

其中, D 为酉矩阵 U_{target} 和 U_{circuit} 的维度. 具体来说, 公式 (4) 成立是因为图 5 中电路可以分成 3 个阶段.

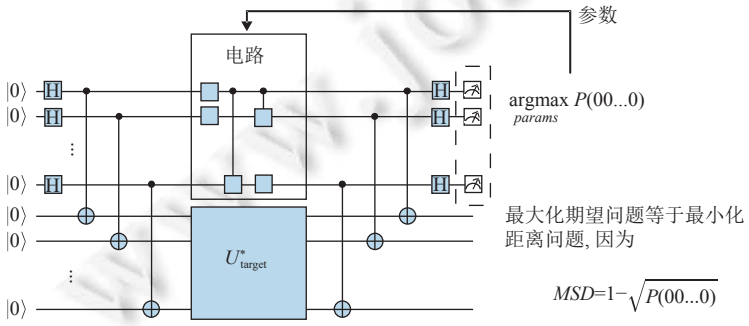


图 5 使用希尔伯特-施密特测试电路进行参数优化

假设阶段电路和 U_{target} 分别操作比特组 A 和 B. 在第 1 步中, 比特通过 H 门和 CNOT 门形成一个纠缠态:

$$|\varphi_{AB}\rangle = \frac{1}{\sqrt{D}}\sum_i (|i\rangle_A \otimes |i\rangle_B),$$

其中, $i \in \{0,1\}^{\otimes N}$, N 是比特组 A 和 B 的比特数. 在第 2 步中, A 和 B 分别被酉矩阵 U_{target} 和 U_{circuit} 操作, 操作后的量子态为:

$$(U^*_{\text{target}} \otimes U_{\text{circuit}})|\varphi_{AB}\rangle = \frac{1}{\sqrt{D}}\sum_i \left[\left(U^*_{\text{target}}|i\rangle_A\right)\otimes(U_{\text{circuit}}|i\rangle_B)\right].$$

第 3 步, 施加和第 1 步相反的 H 门和 CNOT 门, 然后测量, 该操作等价计算在第 1 步的 $|\varphi_{AB}\rangle$ 方向期望:

$$\langle\varphi_{AB}|U^*_{\text{target}} \otimes U_{\text{circuit}}|\varphi_{AB}\rangle = \langle\varphi_{AB}|U^{\dagger}_{\text{target}}U_{\text{circuit}} \otimes I|\varphi_{AB}\rangle = \frac{1}{D}\left|\text{tr}\left(U^{\dagger}_{\text{taregt}}U_{\text{circuit}}\right)\right|,$$

其中, $\langle\varphi_{AB}|U^*_{\text{target}} \otimes U_{\text{circuit}}|\varphi_{AB}\rangle = \langle\varphi_{AB}|U^{\dagger}_{\text{target}}U_{\text{circuit}} \otimes I|\varphi_{AB}\rangle$ 成立是因为量子态 φ_{AB} 是由 H 门和 CNOT 门形成的最大纠缠态, 因此符合 Choi-Jamiołkowski 同构性, 即 $\langle\varphi_{AB}|U^*_{\text{target}} \otimes U_{\text{circuit}}|\varphi_{AB}\rangle = \langle\varphi_{AB}|(U^*_{\text{target}})^{\text{T}} \otimes U_{\text{circuit}}|\varphi_{AB}\rangle = \langle\varphi_{AB}|U^{\dagger}_{\text{target}}U_{\text{circuit}} \otimes I|\varphi_{AB}\rangle$ (具体推导见文献 [24] 的第 2 节). 根据公式 (1) 中 MSD 的定义, 公式 (4) 和期望和概率之间的关系, 可以推出最小化 MSD 等于最大化测量概率 $P(00\ldots0)$:

$$\underset{params}{\text{argmin}}MSD = \underset{params}{\text{argmin}}1 - \sqrt{P(00\ldots0)} = \underset{params}{\text{argmax}}\ \sqrt{P(00\ldots0)} \tag{5}$$

本研究的方法的优势在于其不需要获得电路的酉矩阵, 然后通过矩阵-矩阵乘法计算得到距离, 而是通过复杂度较低的电路模拟来计算距离. 通过基于单振幅模拟器[25], 计算单个基态的概率 $P(00\ldots0)$ 的复杂度为 $O(4^{N-1})$. 而基于 MSD 的方法计算电路和目标酉矩阵的距离, 时间复杂度为 $O(4^N)$. 表 2 展示了本研究方法的加速比, 在经验上与复杂性分析中的 4 倍加速比 $\left(\frac{4^N}{4^{N-1}}\right)$ 相匹配. 对于 4 比特到 6 比特电路的合成, Fast-USYN 在梯度下降中实现了 6.2–8.6 倍的加速. 将 6 量子比特合成的时间中一次迭代的时间从 6.29 s 降低到 0.75 s. 此外, 因为希尔伯特-施密特电路的优化只依赖于单个 $P(00\ldots0)$, 这使得其梯度计算更准确. 也提供了更多将目标酉矩阵逼近到最小距离的机会. 例如表 2 所示, 4 量子比特合成的最小距离从 4.1×10^{-4} 缩小到了 1.1×10^{-6}.

表 2 参数优化方法的性能对比

量子比特	基于MSD的方法		Fast-USYN	
	每次迭代的时间 (s)	最小距离	每次迭代的时间 (s)	最小距离
4	2.31	4.1×10^{-4}	0.37	1.1×10^{-6}
5	4.12	2.9×10^{-3}	0.48	1.2×10^{-4}
6	6.29	5.1×10^{-3}	0.75	4.6×10^{-3}

5 实 验

5.1 实验设置

● Fast-USYN 实现. 本研究基于 Python 3.9.13 和 NumPy 1.23.1 实现了 Fast-USYN. 采用 Jax 0.4.12 和 Pennylane 0.33.0 包进行参数优化, 采用 Scipy 1.11.3 计算候选集的闭包. 在 Fast-USYN 的默认配置中, 覆盖阈值被设置为 0.8. 前瞻搜索的深度设置为 3, 最大候选电路模块设置为 20. 在候选电路集合构造中, 对于每个电路模块, 参数采样数设置 5000. 本研究还在实验中对这些参数选择进行了评估.

● 测试数据集. Fast-USYN 测试的数据集包括 550 个随机酉矩阵, 每个酉矩阵包含 5–8 个量子比特. 每个量子比特的数据集包含了 100 个随机酉矩阵和 10 个量子算法的酉矩阵. 随机酉矩阵由 scipy.stats.unitary_group.rvs 函数生成的, rvs 函数在数学上被证明具有生成任意酉矩阵的能力. 而实验评估所用的算法酉矩阵被列在表 3 中.

● 基准电路合成方法. 本研究将 Fast-USYN 与 QSD[18]、CCD[17]、QFAST[3]和 QuCT[19]这 4 个量子电路合成方法进行了比较. CCD 是 Qiskit[15]所用的默认合成方法. QuCT 是当前最先进的基于搜索的合成方法, 旨在通过搜索

减少合成量子电路中量子门的数量. 为了公平的测试, 本研究对被比较的方法的参数都进行了微调, 以保证其最优性能. 当目标酉矩阵与电路之间的距离在 0.01 以内时, 合成结束并输出结果. 对于执行了超过 7 周的方法, 本研究会终止程序并根据距离收敛曲线估计可能需要的时间.

表 3　实验中所用的量子算法

缩写	算法	缩写	算法
QFT	量子傅里叶算法	HS	哈密顿模拟
ISING	伊辛模型	QNN	量子神经网络
QSVM	量子支持向量机	VQC	变分量子分类器
GHZ	格林伯格-霍恩-齐林格态	—	—

● 指标. 所有实验均在具有 2 个 AMD EPYC 64 核 CPU 和 2TB DDR4 内存的服务器上进行. 为了做一个公平的比较, 所有程序都只使用 20 个核心. 为了比较合成电路的质量, 研究记录了门的数量和电路的深度. 为了比较计算成本, 本研究将合成时间记录为从启动程序启动到输出结果的时间.

5.2　实验结果

● 合成时间的评估. 表 4 展示了 Fast-USYN 与 QSD[18]、CCD[17]、QFAST[3]和 QuCT[19]相比的合成门数量以及合成时间. 与 QuCT 相比, Fast-USYN 在 5 量子比特和 6 量子比特的量子电路合成上实现了 14.5 倍和 20.6 倍的加速. 与 QFAST 相比, Fast-USYN 实现了 304.5 倍和 3 414.19 倍的加速. 与 QuCT 和 QFAST 相比, Fast-USYN 通过对候选空间的剪枝和对量子门参数选择方法进行优化, 实现端到端的加速. 在候选空间方面, 进行 6 量子比特量子电路合成时, QFAST 每次迭代中需要搜索 62 个候选电路模块, 耗时约 4.8 h, 而 Fast-USYN 只搜索 35 个候选. 此外, QFAST 需要首先将酉矩阵分解为更小的酉矩阵, 从而需要额外迭代用于进一步分解, 而 Fast-USYN 直接合成基本门而无需进一步分解.

表 4　4 种电路合成方法与 Fast-USYN 的比较

方法	5 量子比特			6 量子比特			7 量子比特			8 量子比特		
	门数	深度	时间	门数	深度	时间	门数	深度	时间	门数	深度	时间
QSD[18]	1 247.5	465.5	1.5 s	4 761.2	3 652.8	3.6 s	3.8×10^4	3.6×10^4	20.5 h	7.8×10^4	3.1×10^4	111.5 s
CCD[17]	1 376.5	528.1	2.1 s	5 696.1	4 920.3	7.3 s	2.8×10^4	1.8×10^4	33.5 s	9.3×10^4	3.6×10^4	253.5 s
QFAST[3]	887.5	294.1	60.9 h	3 597.1	473.3	6.3 w	—	—	> 6 m	—	—	> 1 y
QuCT[19]	788.1	227.7	2.9 h	3 360.2	432.5	6.4 h	1.5×10^4	671.7	9.4 h	3.1×10^4	1 030.6	17.3 h
Fast-USYN	468.1	182.1	0.2 h	1 255.4	213.2	0.31 h	7 135.2	319.8	2.2 h	1.8×10^4	452.2	4.6 h
提升 (倍)	1.6	1.3	14.5	2.7	2.0	20.6	2.1	2.1	4.1	1.7	2.3	3.7

注: (1) “w”表示星期, “m”表示月, “y”表示年; (2) 提升表示了Fast-USYN相比于QuCT[19]的改进程度

对于 7 量子比特和 8 量子比特的合成, QFAST 需要几个月或几年的时间, 与之相比, Fast-USYN 实现了数百倍加速. 尽管 Fast-USYN 在 7 量子比特和 8 量子比特量子电路合成产生的门数仍然呈指数级增长, 这是由于酉矩阵的参数量在数学上就是指数级别的. 但与 QuCT 相比, Fast-USYN 仍然将合成时间从 9.4 h、17.3 h 减少到 2.2 h、4.6 h. QSD 和 CCD 算法比 Fast-USYN 算法耗时少, 但 Fast-USYN 提供了较大的优化机会, 例如, 将 8 量子比特合成的门数从 7.8×10^4 减少到 1.8×10^4. 此外, 由于噪声在当前硬件上对于量子程序影响更为重要, Fast-USYN 最高 4.6 h 的合成时间在量子电路合成时是可接受的.

● 合成电路的质量. 图 6 比较了 Fast-USYN 与 QuCT 合成结果和最优的量子程序实现的编译质量. 其中最优实现来自当前已知的手动设计的量子程序的文章. 从表 4 中可知, 与 QSD[18]、CCD[17]相比, Fast-USYN 分别最多减少了 76.6% 和 80.4% 的量子门数量, 从而产生更高质量的电路. 例如, 对于 8 量子比特随机生成酉矩阵, 与 QSD 相比, Fast-USYN 将门的平均数量从 7.8×10^4 减少到 1.8×10^4. 在当前的数据集上, 与 QuCT 相比, Fast-USYN 减少了 37.5%–63.0% 量子门数量, 并将 8 量子比特合成的门从 3.1×10^4 到 1.8×10^4, 减少了 1.3×10^4 个门. QuCT 采

用贪心搜索策略和重叠的候选集, 容易陷入局部最优. 本研究还在图 6 中比较 Fast-USYN 在合成 5 比特量子算法的酉矩阵时的性能. 由于这些算法已知最优实现, 因此更容易比较 Fast-USYN 合成结果和理论最优值的距离. 与 QuCT 相比, Fast-USYN 将平均门数量从 119 减少到 58, 减少了 50.6% 的门数量. QuCT 在 VQC 算法合成时实现了大量冗余门, 产生了 374 个量子门, 而 Fast-USYN 只需要 123 个量子门, 减少了 67.1% 的门数量. Fast-USYN 可以找到接近最优结果的门数, 平均只增加了 56.7% 的门数, 而 QuCT 需要增加 221.7% 的门数. 这是因为 Fast-USYN 采用了高质量的候选集, 通常可以在不到 3 次迭代的时间内逼近酉矩阵.

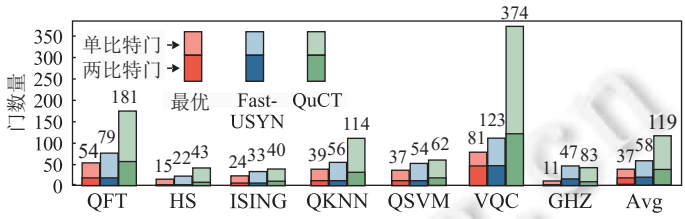
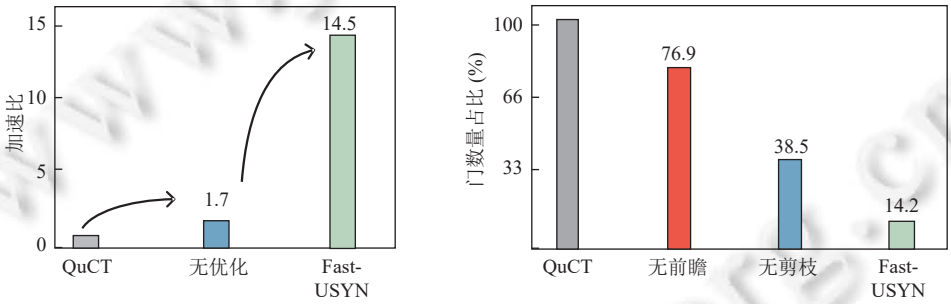


图 6 Fast-USYN 与 QuCT 在合成 7 个量子电路 (5 量子比特) 结果比较

●消融实验. 图 7 (a)、(b) 展示了本研究提出的每种技术对量子电路合成的贡献. 该实验在 5 比特酉矩阵上进行. 相比于 QuCT[19], Fast-USYN 所实现的加速主要归功于基于希尔伯特-施密特的参数优化, 其带来的加速比达到了 8.5 倍. 在不进行参数优化的情况下, Fast-USYN 的加速比为 1.7 倍, 这得益于候选电路模块剪枝带来的迭代次数减少. 具体来说, Fast-USYN 和 QuCT 分别搜索了 35 和 30 个 6 量子比特. 与 QuCT 相比, Fast-USYN 将 23.0 次迭代减少到 13.4 次迭代, 减少了 42% 的迭代次数. 图 7(b) 展示了消融操作后, 以 QuCT 方法为基准 (即 100%) 的门数量对比, 门数减少归功于前瞻策略和剪枝, 分别带来 2.0 倍和 1.3 倍的门数量优化. 前瞻策略避免搜索陷入局部最优, 剪枝策略去除重叠候选集, 总的减少了门数量.



(a) 希尔伯特-施密特参数搜索的消融实验 (b) 前瞻策略和候选集合剪枝的消融实验

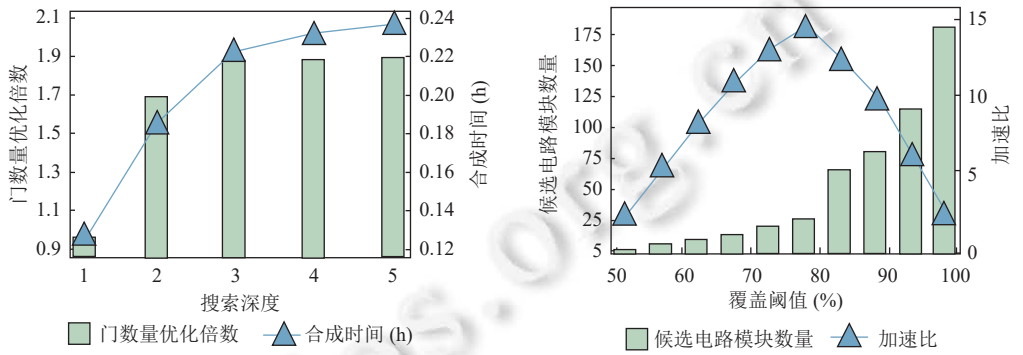
图 7 量子电路 (5 量子比特) 合成的消融实验

● 参数设置的评估. 图 8(a) 展示了在前瞻搜索时设置不同深度时的门数量优化效果. 该实验在 5 比特酉矩阵上进行. 当深度达到 3 时, 门数量优化收敛. 进一步增加深度会导致额外的计算开销, 使合成时间从 0.22 h 增加到 0.24 h. 此外, 本研究还观察到增加深度的开销在深度达到 3 后缓慢增长, 这是因为本研究采用了各种剪枝技术, 以确保搜索在少量的候选中进行. 图 8(b) 评估不同覆盖率阈值下的候选集的候选数和加速比. 在 1 452 个电路模块中使用 174 个作为候选集合可以确保接近 100% 的覆盖率. 此外, 80% 是一个最佳加速点. 这是因为 100% 的覆盖率会不必要地增加了参数优化的开销, 同时不会带来更大的门数量减少.

6 相关工作

在中等规模含噪量子硬件时代, 我们需要通过减少量子的门的数量以减轻噪声的影响[14]. 先前的量子电路合成通过数学分解方程[17]进行, 能发现单量子比特和双量子比特量子电路合成的最优结果. 它们在更大规模的量子电路合成中会产生大量的门. 对于 4–5 量子比特的酉矩阵, 基于搜索的方法的合成程序包含相对较少的门[3]. 但由

于计算成本高, 这些方法无法扩展到更大数量的量子比特. QuCT[19]是最近的一种量子电路合成方法, 它采用数据驱动的方法来加速搜索, 相比于最优量子程序, 它合成的程序仍然会包含 3 倍以上的门. Bocharov 等人[26]使用重复迭代电路进行近似. 一些方法针对特定类型的酉矩阵设计了合成方法, 如 Clifford 酉矩阵[27]和稀疏酉矩阵[28]. Peham 等人[29]提供了一种检验电路等效性的范例, 但该方法不能比较参数化电路的等价性, 无法用于酉矩阵合成中. 与之前的方法相比, Fast-USYN 通过重新设计合成的关键阶段提供了显著的加速和门数量优化. Kang 等人[11]提出了利用模块化的量子电路组成合成量子程序, 但是该方法难以合成任意酉矩阵. Paradis 等人[20]则提出了基于模拟退火的方法合成电路, 但是该方法难以应用于超过 4 量子比特的随机酉矩阵的合成. Li 等人[30]提出了一种基于强化学习的门优化算法, 可以帮助合成电路降低量子门的数量.



(a) 不同前瞻深度对合成时间和门数量的影响 (b) 不同覆盖率阈值对候选集合规模和合成时间的影响

图 8 5 比特量子电路合成中参数设置的影响

除了量子电路合成之外, 同样存在一系列旨在最小化门数量、提高量子程序保真度、以及降低量子程序的编写难度的方法. 例如, 窦星磊等人[31]提出了面向超导量子计算机的程序映射技术, 通过社区发现算法实现辅助量子比特的划分, 并利用动态规划提高量子程序的并行度. 李晖等人[4]则提出了多目标的量子比特映射和路由方法. Das 等人[32]提出采用分段优化和构造模拟相似电路的方式指导门插入, 以降低量子比特空闲产生的退相位噪声. 谢磊等人[33]提出了通过将编译转化为图问题来优化门电路实现中的 Z 通道串扰. Bichsel 等人[34]提出自动的量子比特回收机制, 使编程者可以在量子程序编写时无需手动编写比特的去纠缠和回收. Xu 等人[35]提出了量子编译器的合成, 允许指定量子电路的基础门, 然后自动合成门转换所需的公式. Zhou 等人[36]提出了通过霍尔逻辑对量子程序进行验证的方法.

7 结 论

本文提出 Fast-USYN, 在从酉矩阵到量子电路的合成中实现了最小的门数和延迟. Fast-USYN 采用前瞻策略指导的迭代搜索方法. 通过识别表示空间中高度重叠的候选电路模块, 在搜索中对搜索空间进行剪枝. 通过避免电路的等价酉矩阵的计算, 加快了电路参数的优化. 与目前最先进的方法[19]相比, Fast-USYN 实现了 3.7–20.6 倍的加速的同时, 减少门数量为原有方法的 37.04%–62.50%.

References:

[1] Yuan C, Carbin M. Tower: Data structures in quantum superposition. Proc. of the ACM on Programming Languages, 2022, 6(OOPSLA2): 134. [doi: 10.1145/3563297]

[2] Chen ZY. Design and application of quantum programming language QPanda [Ph.D. Thesis]. Hefei: University of Science and Technology of China, 2021 (in Chinese with English abstract). [doi: 10.27517/d.cnki.gzkju.2021.002170]

[3] Younis E, Sen K, Yelick K, Iancu C. QFAST: Conflating search and numerical optimization for scalable quantum circuit synthesis. In: Proc. of the 2021 IEEE Int'l Conf. on Quantum Computing and Engineering. Broomfield: IEEE, 2021. 232–243. [doi: 10.1109/QCE52317.2021.00041]

[4] Li H, Han ZA, Lu K, Liu SJ, Ju MM. Comprehensive SWAP optimization strategy for improving initial qubit mapping. Computer Engineering and Applications, 2024, 60(14): 66–73 (in Chinese with English abstract). [doi: 10.3778/j.issn.1002-8331.2310-0211]

[5] Wu TA, Jiang YJ, Fang SY. A robust quantum layout synthesis algorithm with a qubit mapping checker. In: Proc. of the 2022 IEEE/ACM Int'l Conf. on Computer Aided Design. San Diego: IEEE, 2022. 1–9.

[6] Christensen M, Tzimpragos G, Kringen H, Volk J, Sherwood T, Hardekopf B. PyLSE: A pulse-transfer level language for superconductor electronics. In: Proc. of the 43rd ACM SIGPLAN Int'l Conf. on Programming Language Design and Implementation. San Diego: ACM, 2022. 671–686. [doi: 10.1145/3519939.3523438]

[7] Gulwani S, Polozov O, Singh R. Program synthesis. Foundations and Trends® in Programming Languages, 2017, 4(1–2): 1–119. [doi: 10.1561/2500000010]

[8] Wang CL, Cheung A, Bodik R. Synthesizing highly expressive SQL queries from input-output examples. In: Proc. of the 38th ACM SIGPLAN Conf. on Programming Language Design and Implementation. Barcelona: ACM, 2017. 452–466. [doi: 10.1145/3062341.3062365]

[9] Zhang J, Li Y, Peng X, Zhao WY. Inductive SQL synthesis with positive and negative tuples. Ruan Jian Xue Bao/Journal of Software, 2023, 34(9): 4132–4152 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6646.htm [doi: 10.13328/j.cnki.jos.006646]

[10] Feser JK, Chaudhuri S, Dillig I. Synthesizing data structure transformations from input-output examples. ACM SIGPLAN Notices, 2015, 50(6): 229–239. [doi: 10.1145/2813885.2737977]

[11] Kang CG, Oh H. Modular component-based quantum circuit synthesis. Proc. of the ACM on Programming Languages, 2023, 7(OOPSLA1): 87. [doi: 10.1145/3586039]

[12] Zhang X. Research and implementation of quantum programming and circuit optimization [MS. Thesis]. Chongqing: Chongqing University, 2019 (in Chinese with English abstract). [doi: 10.27670/d.cnki.gcqdu.2019.002616]

[13] Xie L, Zhai JD. Survey on quantum computing system software. Ruan Jian Xue Bao/Journal of Software, 2024, 35(1): 1–18 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6908.htm [doi: 10.13328/j.cnki.jos.006908]

[14] Jang E, Choi S, Ro WW. Quixote: Improving fidelity of quantum program by independent execution of controlled gates. In: Proc. of the 60th ACM/IEEE Design Automation Conf. San Francisco: IEEE, 2023. 1–6. [doi: 10.1109/DAC56929.2023.10247757]

[15] Aleksandrowicz G, Alexander T, Barkoutsos P, *et al*. Qiskit: An open-source framework for quantum computing. 2019. https://zenodo.org/records/2562111

[16] Omole V, Tyagi A, Carey C, Hanus A, Hancock A, Garcia A, Shedenhelm J, Cowen J. Cirq: A Python framework for creating, editing, and invoking quantum circuits. 2020. https://sdmay20-08.sd.ece.iastate.edu/docs/Design-Document-v2.pdf

[17] Iten R, Colbeck R, Kukuljan I, Home J, Christandl M. Quantum circuits for isometries. Physical Review A, 2016, 93(3): 032318. [doi: 10.1103/PhysRevA.93.032318]

[18] Shende VV, Bullock SS, Markov IL. Synthesis of quantum logic circuits. In: Proc. of the 2005 Asia and South Pacific Design Automation Conf. Shanghai: IEEE, 2005. 272–275. [doi: 10.1109/ASPDAC.2005.1466172]

[19] Tan SW, Lang CL, Xiang L, Wang SD, Jia XH, Tan ZQ. QuCT: A framework for analyzing quantum circuit by extracting contextual and topological features. In: Proc. of the 56th IEEE/ACM Int'l Symp. on Microarchitecture. Toronto: IEEE, 2023. 494–508.

[20] Paradis A, Dekoninck J, Bichsel B, Vechev M. Synthetiq: Fast and versatile quantum circuit synthesis. Proc. of the ACM on Programming Languages, 2024, 8(OOPSLA1): 96. [doi: 10.1145/3649813]

[21] Davis MG, Smith E, Tudor A, Sen K, Siddiqi I, Iancu C. Towards optimal topology aware quantum circuit synthesis. In: Proc. of the 2020 IEEE Int'l Conf. on Quantum Computing and Engineering. Denver: IEEE, 2020. 223–234. [doi: 10.1109/QCE49297.2020.00036]

[22] Sutton BD. Computing the complete CS decomposition. Numerical Algorithms, 2009, 50(1): 33–65. [doi: 10.1007/s11075-008-9215-6]

[23] Dawson CM, Nielsen MA. The Solovay-Kitaev algorithm. Quantum Information & Computation, 2006, 6(1): 81–95.

[24] Jiang M. Channel-state duality. Physical Review A, 2013, 87(2): 022310. [doi: 10.1103/PhysRevA.87.022310]

[25] Schutski R, Lykov D, Oseledets I. Adaptive algorithm for quantum circuit simulation. Physical Review A, 2020, 101(4): 042335. [doi: 10.1103/PhysRevA.101.042335]

[26] Bocharov A, Roetteler M, Svore KM. Efficient synthesis of universal repeat-until-success quantum circuits. Physical Review Letters, 2015, 114(8): 080502. [doi: 10.1103/PhysRevLett.114.080502]

[27] Soeken M, Miller DM, Drechsler R. Quantum circuits employing roots of the pauli matrices. Physical Review A, 2013, 88(4): 042322. [doi: 10.1103/PhysRevA.88.042322]

[28] Malvetti E, Iten R, Colbeck R. Quantum circuits for sparse isometries. Quantum, 2021, 5: 412. [doi: 10.22331/q-2021-03-15-412]

[29] Peham T, Burgholzer L, Wille R. Equivalence checking paradigms in quantum circuit design: A case study. In: Proc. of the 59th ACM/IEEE Design Automation Conf. San Francisco: ACM, 2022. 517–522. [doi: 10.1145/3489517.3530480]

[30] Li ZK, Peng JJ, Mei YX, Lin SN, Wu Y, Padon O, Jia ZH. Quarl: A learning-based quantum circuit optimizer. Proc. of the ACM on

Programming Languages, 2024, 8(OOPSLA1): 114. [doi: 10.1145/3649831]

[31] Dou XL, Liu L, Chen YT. An investigation into quantum program mapping on superconducting quantum computers. Journal of Computer Research and Development, 2021, 58(9): 1856–1874 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2021.20210314]

[32] Das P, Tannu S, Dangwal S, Qureshi M. ADAPT: Mitigating idling errors in qubits via adaptive dynamical decoupling. In: Proc. of the 54th Annual IEEE/ACM Int'l Symp. on Microarchitecture. ACM, 2021. 950–962. [doi: 10.1145/3466752.3480059]

[33] Xie L, Zhai JD, Zhang ZX, Allcock J, Zhang SY, Zheng YC. Suppressing ZZ crosstalk of quantum computers through pulse and scheduling co-optimization. In: Proc. of the 27th ACM Int'l Conf. on Architectural Support for Programming Languages and Operating Systems. Lausanne: ACM, 2022. 499–513. [doi: 10.1145/3503222.3507761]

[34] Bichsel B, Baader M, Gehr T, Vechev M. Silq: A high-level quantum language with safe uncomputation and intuitive semantics. In: Proc. of the 41st ACM SIGPLAN Conf. on Programming Language Design and Implementation. London: ACM, 2020. 286–300. [doi: 10.1145/3385412.3386007]

[35] Xu A, Molavi A, Pick L, Tannu S, Albarghouthi A. Synthesizing quantum-circuit optimizers. Proc. of the ACM on Programming Languages, 2023, 7(PLDI): 140. [doi: 10.1145/3591254]

[36] Zhou L, Yu NK, Ying MS. An applied quantum Hoare logic. In: Proc. of the 40th ACM SIGPLAN Conf. on Programming Language Design and Implementation. Phoenix: ACM, 2019. 1149–1162. [doi: 10.1145/3314221.3314584]

附中文参考文献:

[2] 陈昭昀. 量子程序语言 QPanda 的设计及应用研究 [博士学位论文]. 合肥: 中国科学技术大学, 2021. [doi: 10.27517/d.cnki.gzkju.2021.002170]

[4] 李晖, 韩子傲, 卢凯, 刘述娟, 鞠明媚. 改进量子位初始映射的综合 SWAP 优化策略. 计算机工程与应用, 2024, 60(14): 66–73. [doi: 10.3778/j.issn.1002-8331.2310-0211]

[9] 张健, 李弋, 彭鑫, 赵文耘. 正反例归纳合成 SQL 查询程序. 软件学报, 2023, 34(9): 4132–4152. http://www.jos.org.cn/1000-9825/6646.htm [doi: 10.13328/j.cnki.jos.006646]

[12] 张鑫. 量子编程与线路优化的研究与实现 [硕士学位论文]. 重庆: 重庆大学, 2019. [doi: 10.27670/d.cnki.gcqdu.2019.002616]

[13] 谢磊, 翟季冬. 量子计算系统软件研究综述. 软件学报, 2024, 35(1): 1–18. http://www.jos.org.cn/1000-9825/6908.htm [doi: 10.13328/j.cnki.jos.006908]

[31] 窦星磊, 刘磊, 陈岳涛. 面向超导量子计算机的程序映射技术研究. 计算机研究与发展, 2021, 58(9): 1856–1874. [doi: 10.7544/issn1000-1239.2021.20210314]

谭思危(1997－), 男, 博士, CCF 专业会员, 主要研究领域为量子计算, 计算机体系结构.

陈明帅(1990－), 男, 博士, 研究员, 博士生导师, CCF 高级会员, 主要研究领域为形式化方法, 程序验证与合成.

卢丽强(1994－), 男, 博士, 研究员, 博士生导师, CCF 专业会员, 主要研究领域为量子计算, 人工智能芯片.

尹建伟(1974－), 男, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为软件工程, 量子计算.

郎聪亮(2000－), 男, 硕士生, 主要研究领域为量子计算.