第二届 CCF “司南杯”量子计算编程挑战赛高校组决赛报告

——具有一维链式拓扑结构的量子芯片上的通用计算的探究

摘要 量子技术已经进入有噪声的中尺度量子（NISQ）时代，而通用量子计算机的实现已成为各国政府及科技巨头之间科技竞争的重要领域。目前可用的量子芯片由于受限于可用量子比特数量和质量以及量子操作的类型和精度等因素而难以实现通用量子计算。量子芯片的拓扑结构对其上可实现的量子门的约束已经成为限制芯片上量子操作类型和精度的主要因素，进而成为实现通用量子计算的主要限制之一。本文提出在具有一维链式拓扑结构的量子芯片上构造任意酉操作和制备任意量子态的方案。本文基于最近邻受控非门和单比特旋转门构造了一种通用变分量子线路，并通过训练变分线路来高精度地实现任意酉操作的构造和任意量子态的制备。具体地，该算法应用于三比特 W 态、四比特 GHZ 态、随机三比特量子态以及三比特量子傅里叶变换电路的制备精度均达到了机器精度。

1 引言与背景介绍

量子计算作为量子物理和计算机深度融合的技术，正成长为一个具有广泛应用前景的新兴前沿领域。量子计算有望显著提升经典计算机的计算能力，实现一系列重要实际问题的高效求解，对推动新一轮科技革命和产业变革的高质量发展、提高我国信息产业国际竞争力、保障国家安全具有重要意义。世界各国政府及科技巨头均投入巨大的资源进行量子技术的发展。2022 年底欧盟发布了量子旗舰计划 (Quantum Flagship)，提出了四大技术支柱：量子计算、量子模拟、量子通信、量子传感和计量等技术的 2030 年路线图 [1]。同年美国 Q-NEXT 国家量子信息科学研究中心发布《量子互连路线图》(A Roadmap for Quantum Interconnects）报告，阐述了量子互连在量子计算、通信和传感技术中的作用，并概述了未来 10-15 年开发量子信息技术所需的研究和科学发现及其应用前景 [2]。2021 年 3 月我国发布的《中华人民共和国国民经济和社会发展第十四个五年规划和 2035 年远景目标纲要》指出要瞄准人工智能、量子信息等前沿领域，实施一批具有前瞻性、战略性的国家重大科技项目，加快布局量子计算、量子通信等前沿技术 [3]。通用量子计算机的研制成为量子科技发展的重中之重，已成为各国和各个科技巨头之间量子竞争的必争之地。现有的量子计算机实现方案包括超导线路 [4]、离子阱 [5]、光量子线路 [6]、硅自旋 [7]、拓扑量子计算 [8] 等技术路线，其中超导量子线路的发展最为成熟，芯片上比特数量已经达到 50-100 比特，步入中等尺度量子芯片范围。在有噪声的中尺度量子计算（NISQ）时代，实现通用量子计算的所面临的挑战包括量子比特数量的扩展、量子比特质量的提升、以及高质量通用量子操作的实现等。由于通用量子操作的实现受到量子芯片上量子比特排布的拓扑结构的限制，更具有挑战并亟待解决。

现有的有关拓扑受限的量子芯片上的量子操作的实现方案包括谷歌团队提出的线路结构搜索算法 [9]、Solovay-Kitaev 算法 [10] 等。线路结构搜索是一种基于深度学习的方法，自适应性、可扩展性强，鲁棒性较好，但是对于数据的依赖性高，需要大量的计算资源，且会受限于神经网络的性能。而 Solovay-Kitaev 算法可靠性强，硬件依赖性低，其虽然可以在任何误差界内以任意精度进行逼近，但是受到误差界参数的限制，在某些情况下，该算法可能无法达到所需要的精度。本文基于量子变分线路提出了一种拓扑受限芯片上的量子门实现和量子态制备的通用方案。具体地，本文考虑一维链式拓扑结构的量子芯片情况，即在仅允许单比特量子操作和最近邻比特之间的受控非门的情况下进行的任意量子门实现和量子态制备。本方案在用于三比特 W 态、四比特 GHZ 态、随机三比特量子态以及三比特量子傅里叶变换电路的制备上，均达到了机器精度，且很容易拓展到具有其它拓扑结构芯片上的量子门实现和量子态制备上。

2 主要结果介绍

本文利用拓扑受限量子芯片所允许的量子操作设计出一种通用的量子操作实现和量子态制备的方案。本方案基于芯片所支持的量子门构造了一种量子变分线路，并有潜力通过增加线路层数提高拟合的精度。具体地，本文考虑一维链式拓扑结构的量子芯片，即在仅允许单比特量子操作和最近邻比特之间的受控非门的情况下进行的任意量子门实现和量子态制备。本方案在用于三比特 W 态、四比特 GHZ 态、随机三比特量子态以及三比特量子傅里叶变换电路的制备上，均达到了机器精度。

2.1 具有一维链式拓扑结构的量子芯片上任意量子操作实现和量子态制备的方案

具有一维链式拓扑结构的量子芯片的拓扑结构如图1所示，其所支持的量子操作包括单比特量子门（如 H 门, 泡利 X,Y,Z 门, 旋转门 $R_x(\theta)$,$R_y(\theta)$,$R_z(\theta)$(Eqs.(1)-(5)) ）以及芯片上最近邻比特之间的 CNOT 门。受此限制，在该芯片上实现通用的量子门和任意量子态制备具有一定挑战。

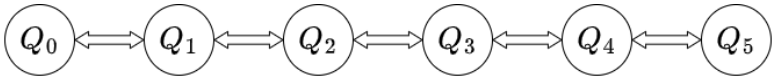


图 1: 一维链式拓扑结构示意图

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \tag{1}$$

$$\sigma_X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \sigma_Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \sigma_Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2}$$

$$R_X(\theta) = \begin{bmatrix} \cos(\theta/2) & -1i \times \sin(\theta/2) \\ -1i \times \sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \tag{3}$$

$$R_Y(\theta) = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \tag{4}$$

$$R_Z(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix} \tag{5}$$

本文基于一维链式拓扑结构的量子芯片所支持的量子门设计一种量子变分线路方案，可高精度地实现任意量子门和任意量子态的制备。本方案的线路结构如图2所示，该参数化量子线路 U' 采用分层式结构进行设计，包含 1 个固定的初始化层，和 m 个结构相同的含参量子线路层 L（m 的值由问题规模和问题类型选定）。具体地，制备 3，4 比特量子态以及实现 3 比特酉算子，所选定的参数值将在3.3节中给出）

每个含参量子线路层仅由量子芯片所支持的单比特门 R 以及芯片结构上的最近邻比特之间的 CNOT 门构成。单比特门 R 的线路结构如图3所示，为单比特任意旋转操作 $U4$ 的有效近似 (我们将在3.1节中证明这种近似的有效性)。每个 R 门包括三个含参量子门 $(R_Z(\alpha), R_Y(\theta), R_Z(\beta))$, 其中 $(\alpha\ \theta, \beta)$ 为相应的比特旋转参数。

利用量子机器学习的方案，对线路中的参数集进行强化学习训练，实现对任意量子门和量子态的制备。具体地，在任意酉算子的制备问题中，所训练的含参线路的训练结果即为对目标酉算子 U 的有效近似 U'；在任意量子态的制备问题中，将训练结果线路作用于处于基态 $|0\rangle^{\otimes n}$ 的 n 比特量子寄存器上，即可获得对目标量子态的有效近似 $|\psi'\rangle$。

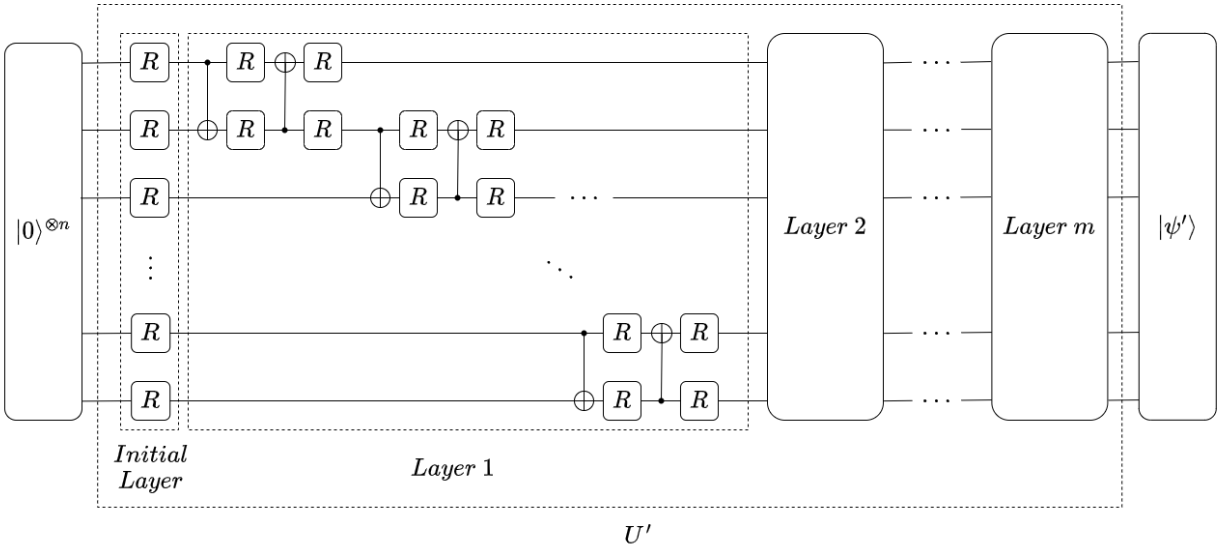


图 2: 算法线路结构示意图

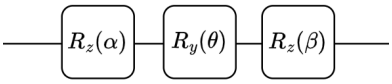


图 3: 单比特旋转操作组 R 的实现

2.2 具有一维链式拓扑结构的量子芯片上任意量子操作实现和量子态制备

2.2.1 制备 4 比特 GHZ 态、3 比特 W 态以及 3 比特傅里叶门

通过对图2中量子线路的强化学习训练，本文所提方案能够在机器精度级别制备任意量子态。具体地，本文进行 4 比特 GHZ 态 Eq.(6) 和 3 比特 W 态 Eq.(7) 的制备，以及 3 比特量子傅里叶门 Eq.(8) 的制备，其中 $\omega = e^{2i/8}$。

$$\mathrm{GHZ}_4 = \frac{1}{\sqrt{2}}\left(|0\rangle^{\otimes 4} + |1\rangle^{\otimes 4}\right) \tag{6}$$

$$\mathrm{W}_3 = \frac{1}{\sqrt{3}}\left(|001\rangle + |010\rangle + |100\rangle\right) \tag{7}$$

$$\mathrm{QFT}_3 = \frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega \end{bmatrix} \tag{8}$$

制备 4 比特 GHZ 态 GHZ_4 和 3 比特 W 态 W_3，以及制备 3 比特傅里叶门 QFT_3 所使用的变分量子线路结构参数 m, n、所用 CNOT 门个数及制备精度（由 Eq.(13) 定义）如表1所示，本方案制备量子态精度可达到机器精度（QTensor 内部数据以单精度浮点数保存，最多支持 7 个有效数字）。

	量子线路层数 m	量子比特个数 n	CNOT 门个数	制备精度 $\text{loss}_{\text{state}}/\text{loss}_{\text{gate}}$
GHZ_4	4	4	24	-6.62312
W_3	3	3	12	-6.78433
QFT_3 门	6	3	24	-5.65342

表 1: 本方案制备 4 比特 GHZ_4 态、3 比特 W_3 态及 3 比特量子傅里叶门 QFT_3 所使用的变分量子线路结构参数 m, n、所用 CNOT 门个数及可达到的制备精度

2.2.2 制备任意 3、4 比特量子态和 3 比特量子门

制备任意 3、4 比特量子态和任意 3 比特量子门所使用的变分量子线路结构参数 m, n、所用 CNOT 门分数及制备精度如表2所示，本方案制备精度可达到机器精度。本方案制备任意量子态、量子门的精度的定义为随机生成的大量量子态、量子门的精度的平均值，如 Eqs.(9,10) 所示。

$$\overline{\text{loss}}_{\text{state}} = \frac{1}{N_{\text{state}}} \sum_{i=1}^{N_{\text{state}}} \text{loss}_{\text{state}}^{i} \tag{9}$$

$$\overline{\text{loss}}_{\text{gate}} = \frac{1}{N_{\text{gate}}} \sum_{i=1}^{N_{\text{gate}}} \text{loss}_{\text{gate}}^{i} \tag{10}$$

此处 $N_{\text{state}}, N_{\text{gate}}$ 为随机制备量子态、量子门的个数。

	量子线路层数 m	量子比特个数 n	CNOT 门个数	制备精度 $\overline{\text{loss}}_{\text{state}}/\overline{\text{loss}}_{\text{gate}}$	$N_{\text{state}}/N_{\text{gate}}$
3 比特量子态	3	3	12	-6.70546	20
4 比特量子态	4	4	24	-6.57032	20
3 比特量子门	6	3	24	-5.84868	20

表 2: 本方案制备任意 3、4 比特量子态、任意 3 比特量子门所使用的变分量子线路结构参数 m, n、所用 CNOT 门个数及可达到的制备精度

3 主要方法与原理

本章主要介绍具有一维链式拓扑结构的量子芯片上任意量子态制备和量子门实现的设计方案和实用性，即相应的量子变分线路设计及其通用性和量子变分线路的训练方式及相应结果分析。

3.1 变分量子线路设计

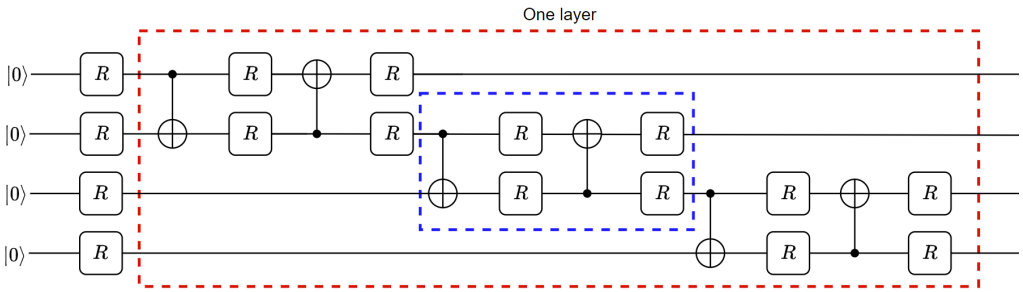


图 4: 由单体门构建的变分量子线路，其中红色框内部分为变分线路的单层线路，蓝色框内部分为本文所设计的核心线路模块。

由单体门和控制非门组成的门集的通用性已经被证明 [11]，且任意多位上的所有酉运算都可以表示为这些门的组合。本文设计的变分量子线路单层模块如图4所示，在不考虑全局相位的情况下，当线路层数足够多的时候，该线路理论上可以模拟任意一个多体酉门。该线路所采用的结构可等效为文献 [12] 中所提出的一种自动可微量子线路 (ADQC) 结构，如图5所示，其可被用于高效地制备任意多量子比特态。但该方案中线路构造的最小不可分单元为任意两比特量子门，无法直接应用于本文所考虑的受芯片拓扑结构以及可支持的量子操作的类型受限的情况。

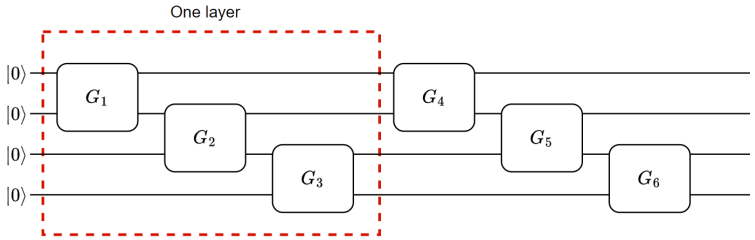
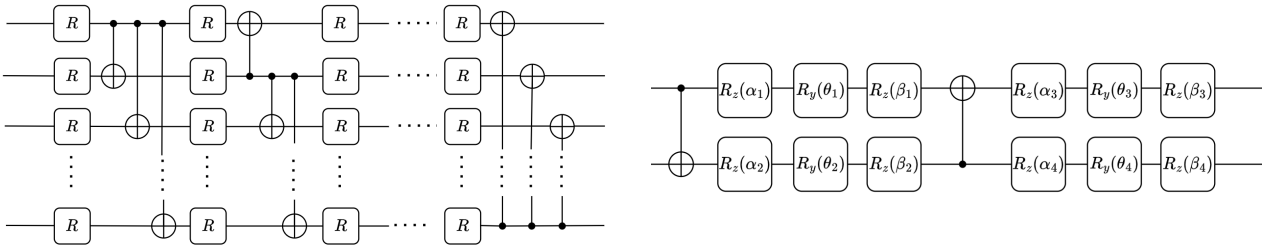


图 5: 由二体门构建的变分量子电路

文献 [13] 提出了将一个二体酉门分解为多个单体门和控制非门进而拓展到制备任意酉门的方案，如图6(a) 所示，但该方案中所给出的多比特通用电路使用了非最近邻比特的控制非门。基于上述工作，本方案设计的用于拓扑受限量子门类型受限的量子芯片上实现任意量子门和量子态的变分线路中所使用的二体门如图6(b) 所示。在该线路中，基于 Eq.(11) 所示的任意单比特门的分解公式，R 门被替换为图3所示的三个单比特旋转门。

$$U4 = e^{i\delta} \cdot \begin{bmatrix} e^{i\frac{\alpha}{2}} & 0 \\ 0 & e^{-i\frac{\alpha}{2}} \end{bmatrix} \cdot \begin{bmatrix} cos\frac{\theta}{2} & sin\frac{\theta}{2} \\ -sin\frac{\theta}{2} & cos\frac{\theta}{2} \end{bmatrix} \cdot \begin{bmatrix} e^{i\frac{\beta}{2}} & 0 \\ 0 & e^{-i\frac{\beta}{2}} \end{bmatrix} \tag{11}$$



(a) (b)

图 6: (a) 利用单体门和受控非门模拟任意二体门;(b) 由基本门构建的线路基本模块

3.2 量子机器学习

机器学习算法在图像识别、语音识别、自然语言处理等领域表现极佳且得到广泛应用。量子机器学习算法的应用也受到广泛关注且被应用于量子近似优化 [14]，量子本征值求解 [15]，量子化学模拟 [16]，量子金融 [17] 等领域。本文将应用量子机器学习思想对量子变分线路进行训练，完成任意量子态制备和量子门实现。具体地，变分量子算法 (Variational Quantum Algorithm, VQA)[18] 用一个经典的优化器来训练一个含参量子线路。经典的机器学习算法中 model 通常指的是经典计算机中的神经网络，而在变分量子算法中 model 指的是量子线路。通过取最小化变分量子线路中成本函数 (cost function) 来达到寻找最优的参数的目的。

本文通过变分量子算法来实现任意量子态和量子门的制备。本方案所使用的量子变分线路如图2所示，使用的均为在芯片拓扑结构限制下所允许的量子门。

本方案在制备任意量子态的情况下，设定的损失函数如 Eqs.(12) 所示，其中 $|\psi\rangle_{\text{output}}$ 为变分线路当前制备的量子态，下标 2 代表采用 2 范数（欧氏距离）来衡量制备态和目标态之间的相似度。本方案在制

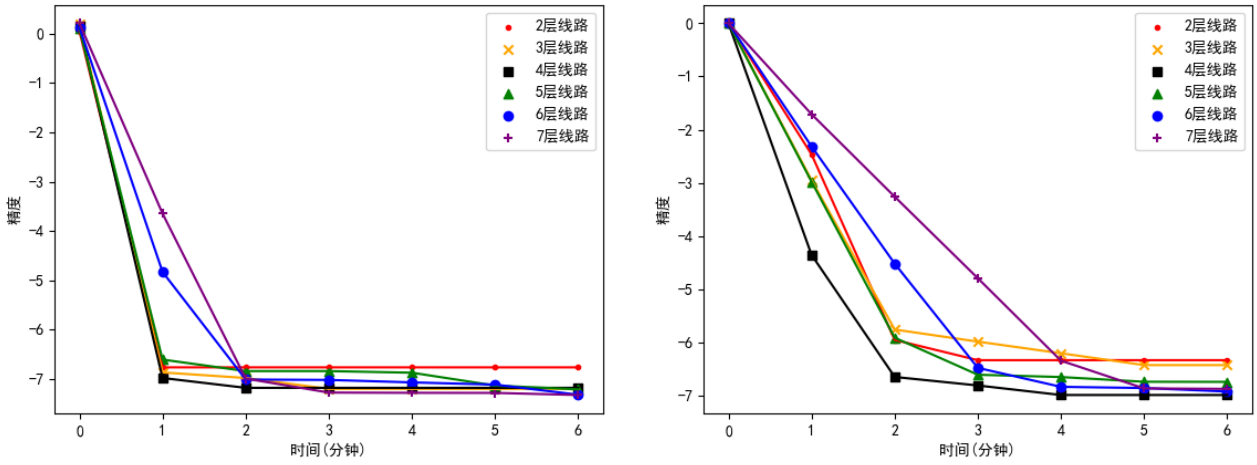
备任意量子门的情况下，设定的损失函数如 Eqs.(13) 所示，其中 U_{vqa} 为变分线路当前实现的量子门，下标 F 代表采用 F 范数来衡量制得的酉门与目标酉门之间的相似度。两式中的 $e^{i\theta}$ 项是为消除酉操作全局相位而设定。本方案中量子变分线路使用自适应学习率的 Adam 优化器进行参数优化，收敛速度更快。

$$\mathrm{loss}_{\mathrm{state}} = \log_{10}\left\| e^{i\theta}\left|\psi\right\rangle_{\mathrm{output}} - \left|\psi\right\rangle_{\mathrm{target}} \right\|_2 \tag{12}$$

$$\mathrm{loss}_{\mathrm{gate}} = \log_{10}\left\| e^{i\theta}U_{\mathrm{vqa}} - U_{\mathrm{target}} \right\|_{\mathrm{F}} \tag{13}$$

3.3 量子变分线路的参数分析

收敛速度和执行所需得运行时间是执行量子变分算法所必须考虑的因素。本文实验考察具有不同迭代线路层的量子变分线路用于制备 3、4 比特量子态的损失函数的收敛情况如图7所示，用于制备三比特酉算子的损失函数随训练时间的趋势如图8所示。本文中用损失函数的收殓值来衡量算法的精度精度。从图中可看出在制备精度达到机器精度的情况下，制备 3、4 量子比特态所需的最少线路层数分别为 3 层和 4 层，而制备任意 3 比特量子门所需的最少线路层数为 6 层。在此种变分线路结构下线路收敛所需的时间均低于 10 分钟。



(a) (b)

图 7: (a) 不同层数线路在拟合一个三比特量子态时的精度-时间曲线;(b) 不同层数线路在拟合一个四比特量子态时的精度-时间曲线。精度用损失函数的收敛值衡量

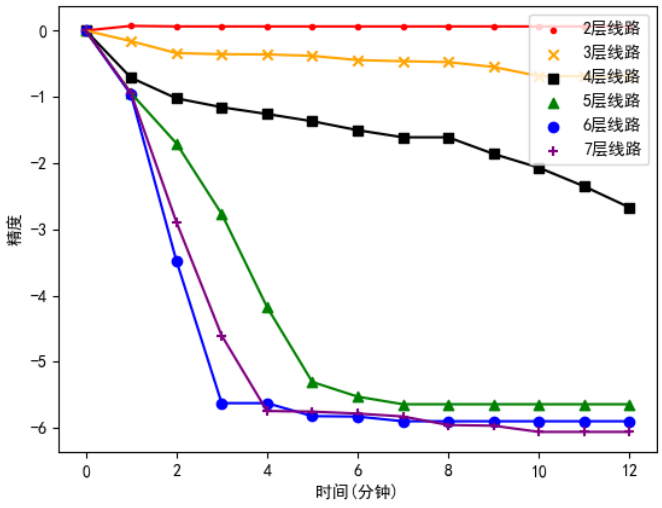
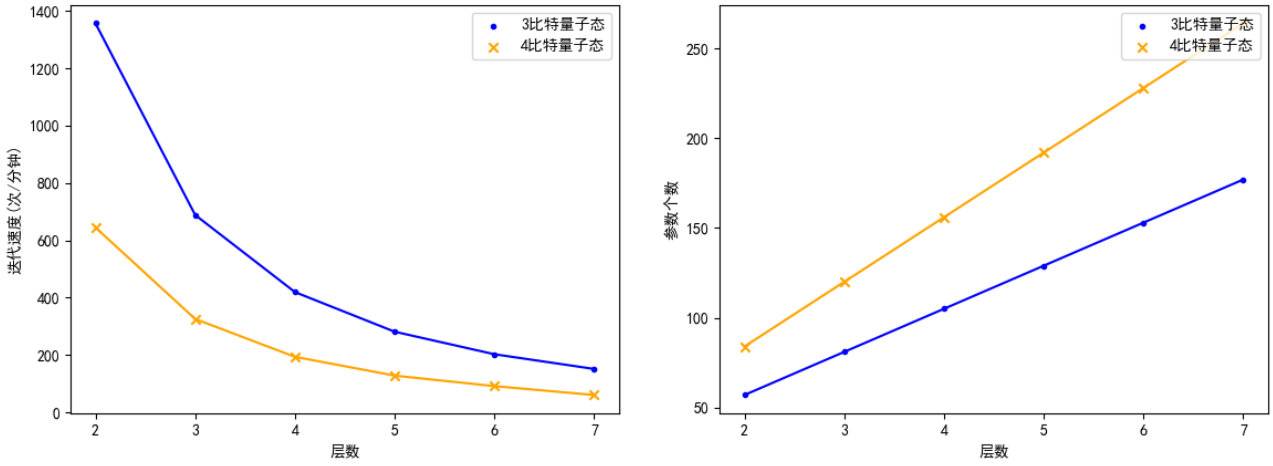


图 8: 不同层数线路在拟合一个三比特酉算子时的精度-时间曲线，精度用损失函数的收敛值衡量

本文所设计的量子变分线路用于制备任意量子态和量子门达到参数收敛所需的时间以及制备的精度与所使用变分线路的层数 m 和量子比特数量 n 密切相关。本节通过实验来探究本文所提方案的表现。当本方案使用 m 层线路进行 n 量子比特的拟合问题时，线路中的参数个数 param_num 可由如下公式得出

$$\text{param\_num} = 3n + 12m(n-1) \tag{14}$$

使用具有不同层数的变分线路制备 3、4 比特量子态时线路的参数更新速率（次/分）及相应的变分线路中参数个数与变分线路层数之间关系如图9所示。由此图可见，变分线路中参数个数与变分线路中参数更新次数呈反关联关系。具有不同层数的变分线路中单个参数更新时间（用线路参数更新一次所用时间除以线路参数总数）与变分线路层数之间关系如图10所示。当变分线路层数较低时，变分线路中单个参数更新时间随层数增加呈正比例增加趋势，当变分线路层数增到 7 层，变分线路中单个参数更新时间随层数增加速率陡增。因此，变分线路层数的选择要权衡线路参数更新时间以及精度两方面因素。以上所有数据均来自本地计算机运行结果, 其配置为 15 vCPU AMD EPYC 7543 32-Core Processor 30GB ubuntu18.04。



(a) (b)

图 9: (a) 本方案用于制备 3、4 比特量子态的变分线路的参数更新速率（次/分钟）与变分线路层数之间关系;(b) 本方案用于制备 3、4 比特量子态的变分线路的参数个数与变分线路层数之间关系

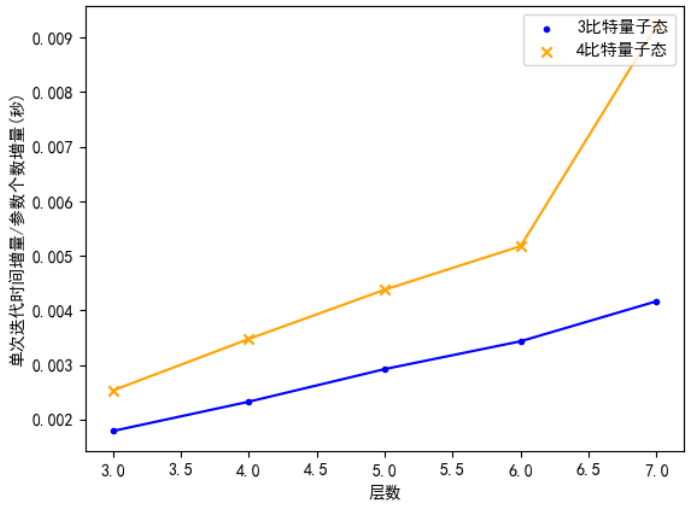


图 10: 本方案用于制备 3、4 比特量子态的变分线路单次迭代时间延长幅度 (秒) 与变分线路层数之间关系

4 结论和展望

本文提出在具有一维链式拓扑结构的量子芯片上的任意酉操作构造和任意量子态制备方案。本文基于最近邻受控非门和单比特旋转门构造了一种通用变分量子线路，并通过训练变分线路来高精度地实现任意酉操作的构造和任意量子态的制备。具体地，该算法应用于三比特 W 态、四比特 GHZ 态、随机三比特量子态以及三比特量子傅里叶变换电路的制备的精度均达到了机器精度。本方案中的所需量子线路的层数需要按照所需实现的门/比特的维度以及所需制备的门/比特的精度进行调整。本文所提方案具有通用性，可以拓展到具有其它拓扑结构的量子芯片上（包括高维拓扑结构的芯片）的通用量子计算的设计，为实现量子芯片通用量子计算提供解决方案。

参考文献

[1] The quantum flagship. Accessed: 2023-05-26.

[2] David D. Awschalom, Hannes Bernien, Rex Brown, Aashish Clerk, Eric Chitambar, Alan Dibos, Jennifer Dionne, Mark Eriksson, Bill Fefferman, Greg David Fuchs, Jay Gambetta, Elizabeth Goldschmidt, Supratik Guha, F. Joseph Heremans, Kent David Irwin, Ania Bleszynski Jayich, Liang Jiang, Jonathan Karsch, Mark Kasevich, Shimon Kolkowitz, Paul G. Kwiat, Thaddeus Ladd, Jay Lowell, Dmitri Maslov, Nadya Mason, Anne Y. Matsuura, Robert McDermott, Rod van Meter, Aaron Miller, Jason Orcutt, Mark Saffman, Monika Schleier-Smith, Manish Kumar Singh, Phil Smith, Martin Suchara, Farzam Toudeh-Fallah, Matt Turlington, Benjamin Woods, and Tian Zhong. A roadmap for quantum interconnects. 7 2022.

[3] 新华社. 中华人民共和国国民经济和社会发展第十四个五年规划和 2035 年远景目标纲要, 2021.

[4] Qiujiang Guo, Cheng Cheng, Zhi-Hao Sun, Zedong Song, Hekang Li, Zhen Wang, Wuxin Ren, Hui Dong, Da Zheng, Yirong Zhang, et al. Observation of energy-resolved many-body localization. *Nature Physics*, 17:234–239, 2021.

[5] Jiehang Zhang, Giuseppe Pagano, Paul W Hess, Antonis Kyprianidis, Peter Becker, Harvey B Kaplan, Alexey V Gorshkov, Zhexuan Gong, and Christopher Monroe. Observation of a many-body dynamical phase transition with a 53-qubit quantum simulator. *Nature*, 551:601–604, 2017.

[6] Han-Sen Zhong, Hui Wang, Yu-Hui Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Da Wu, Xingdong Ding, Yi Hu, et al. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, 2020.

[7] Jun Yoneda, Kenta Takeda, Tomohiro Otsuka, Takashi Nakajima, Matthieu R Delbecq, Giles Allison, Takashi Honda, Tetsuo Kodera, Shunri Oda, Yuto Hoshi, et al. A quantum-dot spin qubit with coherence limited by charge noise and fidelity higher than 99.9%. *Nature nanotechnology*, 13(2):102–106, 2018.

[8] Chetan Nayak, Steven H Simon, Ady Stern, Michael Freedman, and Sankar Das Sarma. Non-abelian anyons and topological quantum computation. *Reviews of modern physics*, 80(3):1083–1159, 2008.

[9] Yuxuan Du, Tao Huang, Shan You, Min-Hsiu Hsieh, and Dacheng Tao. Quantum circuit architecture search for variational quantum algorithms. *npj Quantum Information*, 8(1):1–11, 2022.

[10] Michael A. Nielson Christopher M. Dawson. The solovay-kitaev algorithm. *Quantum Information and Computation*, 0(0):000–000, 2005.

[11] Barenco, Bennett, Cleve, DiVincenzo, Margolus, Shor, Sleator, Smolin, and Weinfurter. Elementary gates for quantum computation. *Physical review. A, Atomic, molecular, and optical physics*, 52 5:3457–3467, 1995.

[12] Peng-Fei Zhou, Rui Hong, and Shi-Ju Ran. Automatically differentiable quantum circuit for many-qubit state preparation. *Physical Review A*, 104(4):042601, 2021.

[13] Paulo Regis Menezes Sousa and Rubens Viana Ramos. Universal quantum circuit for n-qubit quantum gate: a programmable quantum gate. *Quantum Inf. Comput.*, 7:228–242, 2006.

[14] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Physical Review X*, 10(2):021067, 2020.

[15] Jin-Min Liang, Shu-Qian Shen, Ming Li, and Shao-Ming Fei. Quantum algorithms for the generalized eigenvalue problem. *arXiv preprint arXiv:2112.02554*, 2021.

[16] Yuan Su, Dominic W Berry, Nathan Wiebe, Nicholas Rubin, and Ryan Babbush. Fault-tolerant quantum simulations of chemistry in first quantization. *PRX Quantum*, 2(4):040332, 2021.

[17] Dylan Herman, Cody Googin, Xiaoyuan Liu, Alexey Galda, Ilya Safro, Yue Sun, Marco Pistoia, and Yuri Alexeev. A survey of quantum computing for finance. *arXiv preprint arXiv:2201.02773*, 2022.

[18] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.

A 附录: 实现代码

Listing 1: **answer.py**

```python
#导入必须的库和函数
import numpy as np
import pyqpanda as pq
from pyvqnet.qnn.quantumlayer import QuantumLayer
from pyvqnet.optim import adam
from pyvqnet.tensor import QTensor
from pyvqnet.nn.module import Module
from pyvqnet.utils import set_random_seed

# 设置量子计算层的参数初始化随机数种子
set_random_seed(256)

g_param_num_u4 = 3      # 构建U4门所需的参数个数（即Ry,Rz,Ry三个门的旋转角度）
g_param_num_double_gate = 4 * g_param_num_u4    # 构建二体门所需的参数个数（需要 CNOT门*2 + U4门*4）
g_layer_num = 0         # 基本块层数
g_qstate = None         # 目标量子态
g_umatrix = None        # 目标酉矩阵


def build_u4(qubit, params):
    """使用RZ,RY,RZ门搭建U4门"""
    circuit = pq.QCircuit()
```

```
    circuit.insert(pq.RZ(qubit, params[0]))
    circuit.insert(pq.RY(qubit, params[1]))
    circuit.insert(pq.RZ(qubit, params[2]))
    return circuit


def build_double_gate(qlist, params):
    """使用单体门和受控非门搭建二体门"""
    global g_param_num_u4

    circuit = pq.QCircuit()
    now = 0     # 当前已使用的参数个数
    circuit.insert(pq.CNOT(qlist[0], qlist[1]))
    circuit.insert(build_u4(qlist[0], params[now: now + g_param_num_u4]))
    now += g_param_num_u4
    circuit.insert(build_u4(qlist[1], params[now: now + g_param_num_u4]))
    now += g_param_num_u4

    circuit.insert(pq.CNOT(qlist[1], qlist[0]))
    circuit.insert(build_u4(qlist[0], params[now: now + g_param_num_u4]))
    now += g_param_num_u4
    circuit.insert(build_u4(qlist[1], params[now: now + g_param_num_u4]))
    return circuit


def build_layer(qlist, params, qubit_num):
    """在所有相邻的两个量子比特间叠加一层二体门，视为一层基本块"""
    global g_param_num_double_gate

    circuit = pq.QCircuit()
    for i in  range(qubit_num - 1):
        double_gate = build_double_gate(
                    [qlist[i], qlist[i + 1]],
                    params[i * g_param_num_double_gate: (i + 1) * g_param_num_double_gate])
        circuit.insert(double_gate)
    return circuit


def build_cir(qlist, params, layer_num, qubit_num):
    """构建通用量子线路"""
    global g_param_num_u4
    global g_param_num_double_gate
    param_num_layer = (qubit_num - 1) * g_param_num_double_gate     # 在所有相邻的两个量子比特间叠加一层二体门所需要的参数个数

    circuit = pq.QCircuit()
    now = 0     # 当前已使用的参数个数
    # 首先是初始化层，即对所有量子比特添加一层U4门
    for i in  range(qubit_num):
        circuit.insert(build_u4(qlist[i], params[0 + now: g_param_num_u4 + now]))
        now += g_param_num_u4

    # 接着是layer_num层基本块
    for i in  range(layer_num):
```

```
        layer = build_layer(
            qlist,
            params[0 + now: param_num_layer + now],
            qubit_num)
        circuit.insert(layer)
        now += param_num_layer
    return circuit


def qvc_circuit_qstate( input, params, qlist, clist, machine):
    """目标是量子态时的变分量子线路与损失函数定义"""
    global g_qstate
    global g_layer_num
    qubit_num =  len(qlist)
    circuit = build_cir(qlist, params, g_layer_num, qubit_num)
    prog = pq.QProg()
    prog.insert(circuit)

    machine.directly_run(prog)
    qstate = np.array(machine.get_qstate())
    loss = np.linalg.norm(g_qstate.reshape(-1) - (np.exp(1j*params[-1]))*qstate) # 排除全局相位后计算制备态与目标态之间的欧式距离
    return np.log10(loss)   # 返回欧式距离的对数形式作为损失函数，避免梯度消失


def qvc_circuit_Umatrix( input, params, qlist, clist, machine):
    """目标是酉矩阵时的变分量子线路与损失函数定义"""
    global g_umatrix
    global g_layer_num
    qubit_num =  len(qlist)
    circuit = build_cir(qlist, params, g_layer_num, qubit_num)
    prog = pq.QProg()
    prog.insert(circuit)

    machine.directly_run(prog)
    mat = np.array(pq.get_matrix(prog))
    loss = np.linalg.norm(g_umatrix.flatten() - np.exp(1j*params[-1])*mat)  # 排除全局相位后计算拟合的酉矩阵与目标酉矩阵之间的欧式距离
    return np.log10(loss)   # 返回欧式距离的对数形式作为损失函数，避免梯度消失


class Model(Module):
    """使用VQNet的优化算法进行模型训练"""
    def __init__(self, question, param_num, qubit_num):
         super(Model, self).__init__()
        global g_layer_num
        if question == "question1":
            self.pqc = QuantumLayer(qvc_circuit_qstate, param_num, "cpu", qubit_num)
            self.qubit_num = qubit_num
        elif question == "question2":
            self.pqc = QuantumLayer(qvc_circuit_Umatrix, param_num, "cpu", qubit_num)
            self.qubit_num = qubit_num

        self.best_loss = 0          # 记录训练过程中的最佳loss
```

```
        self.best_params = None     # 记录对应的最佳参数
        self.optimizer = adam.Adam(self.parameters(), lr=0.01)  # 使用自适应学习率的Adam优化器优化参数


    def forward(self):
        """定义前向传递逻辑"""
         input = QTensor([[None]]) # 必须要加才能过编译
        return self.pqc( input)

    def get_circuit(self, qlist):
        """返回参数训练后的最终电路"""
        return build_cir(qlist, self.best_params, g_layer_num, self.qubit_num)


def question1(quantum_state_vector, qlist):
    global g_param_num_u4
    global g_param_num_double_gate
    global g_qstate
    global g_layer_num
    g_qstate = quantum_state_vector
    qubit_num =  len(qlist)
    g_layer_num = 3 if qubit_num == 3 else 4    # 制备3bits量子态需要三层基本块，4bits量子态需要四层基本块
    # 总参数量除了量子线路内部待训练的旋转门参数以外还包括一个自适应的全局相位参数，用以排除全局相位的影响
    param_num = qubit_num * g_param_num_u4 + (qubit_num - 1) * g_param_num_double_gate * g_layer_num + 1

    model = Model("question1", param_num, qubit_num)
    model.train()
    epoch = 500     # 训练500次
    print("Start training")
    loss_arr = []

    for i in  range(epoch):
        model.optimizer.zero_grad()
        loss = model.forward()
        loss.backward()
        model.optimizer._step()
        loss_arr.append(loss.to_numpy()[0])
        # 记录最佳的loss以及对应的参数
        if loss_arr[-1] < model.best_loss:
            model.best_loss = loss_arr[-1]
            model.best_params = model.pqc.m_para.to_numpy()[:-1]

        if i % 20 == 19:
            print(f"epoch {i} loss: {loss.to_numpy()[0]}")

    return model.get_circuit(qlist)


def question2(unitary_matrix, qlist):
    global g_param_num_u4
    global g_param_num_double_gate
    global g_umatrix
    global g_layer_num
    g_umatrix = unitary_matrix
```

```
    qubit_num =  len(qlist)
    g_layer_num = 6     # 制备QFT(3)酉矩阵需要6层基本块
    # 总参数量除了量子线路内部待训练的旋转门参数以外还包括一个自适应的全局相位参数，用以排除全局相位的影响
    param_num = qubit_num * g_param_num_u4 + (qubit_num - 1) * g_param_num_double_gate * g_layer_num + 1

    model = Model("question2", param_num, qubit_num)
    model.train()
    epoch = 600     # 训练600次
    print("Start training")
    loss_arr = []

    for i in  range(epoch):
        model.optimizer.zero_grad()
        loss = model.forward()
        loss.backward()
        model.optimizer._step()
        loss_arr.append(loss.to_numpy()[0])
        # 记录最佳的loss以及对应的参数
        if loss_arr[-1] < model.best_loss:
            model.best_loss = loss_arr[-1]
            model.best_params = model.pqc.m_para.to_numpy()[:-1]

        if i % 20 == 19:
            print(f"epoch {i} loss: {loss.to_numpy()[0]}")

    return model.get_circuit(qlist)
```