时空高效的低深度量子态制备及其应用

Kaiwen Gui*, [1,2,3], Alexander M. Dalzell*, [4], Alessandro Achille[5], Martin Suchara[1], 和 Frederic T. Chong[3]

[1]亚马逊云科技，美国华盛顿州[2]芝加哥大学普利兹克分子工程学院，美国伊利诺伊州[3]芝加哥大学计算机科学系，美国伊利诺伊州[4]AWS量子计算中心，美国加利福尼亚州帕萨迪纳[5]AWS人工智能实验室，美国加利福尼亚州帕萨迪纳

arXiv:2303.02131v3 [quant-ph] 9 Feb 2024

我们提出了一种制备任意量子态的新型确定性方法。将我们的协议编译为受控非门和任意单量子比特门后，可在深度 $O(\log(N))$ 内制备一个 N 维量子态，并实现 $O(N)$ 的时空分配（这一指标考虑了某些辅助量子比特通常无需在整个量子电路运行期间保持激活这一事实）；这两项指标均为最优。编译为{H,S,T,CNOT}门集后，我们证明，与以往方法相比，该方法渐近地需要更少的量子资源。具体而言，该方法以最优深度 $O(\log(N) + \log(1/\epsilon))$ 和 $O(N \log(\log(N)/\epsilon))$ 的时空分配，将任意量子态制备至误差 ϵ 以内，分别优于 $O(\log(N)\log(\log(N)/\epsilon))$ 和 $O(N\log(N/\epsilon))$。我们说明，该协议较低的时空分配如何支持快速制备多个互不相交的量子态，同时仅需常数因子的辅助量子比特开销——通过高效复用 $O(N)$ 个辅助量子比特，可在深度 $O(w + \log(N))$ 内制备由 w N 维量子态构成的乘积态，而不是 $O(w\log(N))$，从而实现每个量子态的有效常数深度。我们重点介绍了这项能力可能发挥作用的若干应用，包括量子机器学习、哈密顿量模拟和求解线性方程组。我们提供了该协议的量子电路描述、详细伪代码，以及使用 Braket 的量子门级实现示例。

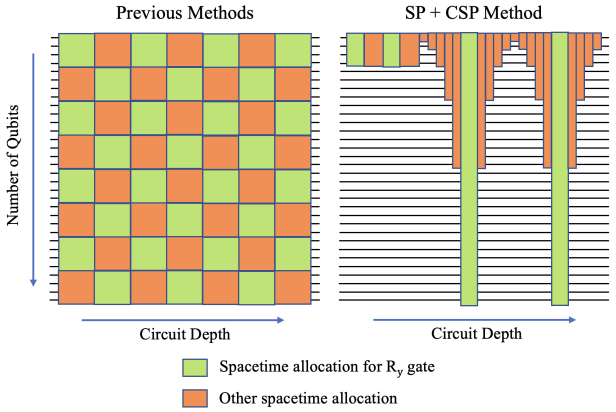


图1：我们提出的态制备方法（SP+CSP）与先前方法的时空分配示意图。绿色区域表示为任意单量子比特旋转分配的时空。橙色区域表示分配给活跃量子比特的时空（即不处于 $|0\rangle$ 中），这些量子比特不进行单量子比特旋转；它们可能经历受控非门或 Toffoli 门等操作，也可能在与其他量子比特纠缠时处于空闲状态。我们的方法中的时空分配渐近地小于量子比特数与深度的乘积。

1 引言

量子态制备（QSP）是许多量子算法中的关键子程序，这些算法声称在量子机器学习等应用中比其经典对应算法更快 [1–8]，模拟量子系统 [9–12]，求解线性方程组 [13–15]，合成酉算子操作 [16–18]。

态制备问题是创建如下形式的任意量子态：

$$|\psi\rangle = \frac{1}{\|\mathbf{x}\|}\sum_{i=0}^{2^n-1} x_i\,|i\rangle \tag{1}$$

其中 n 表示量子比特的数量，$\|\cdot\|$ 表示标准欧几里得范数，而 $\mathbf{x}$ 是一个 2^n维向量，其分量为 $x_i \in \mathbb{C}$。在许多应用中，令 $x_i \in \mathbb{R}^+ \cup \{0\}$即可满足需求，因此为简便起见，以下均作此假设；额外的相位信息可以轻松纳入，且开销极小（见附录 C）。我们还将 $N = 2^n$ 记为编码的参数总数，它也等于向量空间的维数。

*这两位作者贡献相同；kgui@uchicago.edu, dalzel@amazon.com

已于 2024年2月2日 接收，点击标题进行验证。根据 CC-BY 4.0 协议发布。

门集	{U(2), 受控非门}		{H门, S 门, T门,受控非门}	
	深度	时空分配	深度	时空分配
Sun 等人 [17]	$O(\log(N)\log(\log(N)))$	$\Theta(N)$	$O(\log(N)\log(\log(N))\log(N/\epsilon))$	$O(N$ 对数$(N/\epsilon))$
Sun 等人 [17]	$\Theta(\log(N))$	$O(N$ 对数$(N))$	$O(\log(N)\log(N/\epsilon))$	$O(N\log(N)\log(N/\epsilon))$
Zhang 等人 [19]	$\Theta(\log(N))$	$O(N$ 对数$(N))$	$O(\log(N)\log(\log(N)/\epsilon))$	$O(N\log(N)\log(\log(N)/\epsilon))$
Yuan 等人 [18]	$\Theta(\log(N))$	$\Theta(N)$	$\Omega(\log(N)\log(\log(N)/\epsilon))^{\dagger}$	$\Omega(N\log(\log(N)/\epsilon))^{\dagger}$
Clader 等人 [20]	$O(\log^2(N))$	$O(N\log^2(N))$	$O(\log^2(N)+\log(1/\epsilon))$	$O(N(\log^2(N)+\log(1/\epsilon)))$
本工作（SP）	$\mathbf{\Theta(\log(N))}$	$\mathbf{O(N}$ **对数**$\mathbf{(N))}$	$\mathbf{\Theta(\log(N)+\log(1/\epsilon))}$	$\mathbf{O(N}$ **对数**$\mathbf{(N/\epsilon))}$
本工作（SP+CSP）	$\mathbf{\Theta(\log(N))}$	$\mathbf{\Theta(N)}$	$\mathbf{\Theta(\log(N)+\log(1/\epsilon))}$	$\mathbf{O(N\log(\log(N)/\epsilon))}$

表 1：我们所得结果与此前最先进的低深度 QSP 方法的比较。 $N=2^n$ 表示基组/参数的总数， ϵ 表示误差精度参数。请注意，Sun 等人 [17] 提出了 QSP 协议的其他变体，这些变体具有更大的电路深度和更少的辅助量子比特，其时空分配均为 $\Theta(N)$ 以及 $O(N\log(N/\epsilon))$。Ref.† 一行中的匕首符号表示，根据我们的计算得出的下界，因为该论文未分析 Clifford+ T 成本。请注意，Clader 等人 [20]优化了 T 深度。我们还指出，我们的 SP+CSP 协议允许部分辅助寄存器为脏态，与 [16]中的做法类似。参见使用 {H门、S 门、T门、受控非门} 门集时空分配的相应理论下界，见第2.3节。

Sun 等人的近期进展 [17] 提出了一种最优构造，可制备 $|\psi\rangle$ ，其电路深度为 $\Theta(2^n/n)$[1]，使用任意单量子比特门和双量子比特 CNOT 门（以下简称 {U(2),CNOT} 门集），且不需要辅助量子比特。另一方面，若有辅助量子比特可用，则可以大幅降低电路深度。当态制备过程必须快速完成或需要连续重复多次时，采用低深度协议更为理想。很多时候，制备 $|\psi\rangle$之后执行的量子算法（例如进行机器学习推理）的深度为 $\mathrm{poly}(n)$，比无辅助量子比特的量子态制备实现快指数倍，因此低深度状态制备对于使总体运行时间保持合理至关重要。

为此，近期的确定性方法[17, 19, 21] 已经在 $\Theta(n)$ 量子电路深度方面实现了最优结果，所使用的门集为 {U(2)、受控非门} 。然而，这种指数级深度缩减伴随着空间开销的指数级增加。换言之，需要指数数量的辅助量子比特。近期工作 [18] 能够以仅 $\Theta(n)$ 深度和仅$\Theta(2^n/n)$ 辅助量子比特实现这一目标，这已经是最优的。

在考量所需量子资源的总体权衡关系时，通常会考虑电路深度、电路规模和量子比特数等指标。假设物理架构允许并行执行量子门，则电路深度可作为计算总体运行时间的代理指标，而量子比特数则代表必须为计算分配的总体空间。现在，我们提出另一项指标——**时空分配**——即每个量子比特必须保持活跃的总时间（即，不处于 $|0\rangle$ 态），并对所有量子比特求和。时空分配以下界受电路规模约束，上界受量子比特数与电路深度之积约束，但它具有独特的操作含义。在这样一种模型中，如果希望使用固定数量的辅助量子比特执行许多不同的、对辅助量子比特需求密集的任务（例如快速制备许多相互独立的 n量子比特态），处于 $|0\rangle$ 态的新辅助量子比特的可用性就会成为算法瓶颈。假设辅助量子比特一旦恢复为 $|0\rangle$ 态，就可以在不同任务之间重新分配，那么完成这批任务所需的总运行时间将由这些任务的时空分配决定，而不是由其深度或规模决定。在深度或规模方面达到最优的态制备算法，不一定也能在时空分配方面达到最优。我们将证明，当编译到 {U(2),受控非门} 门集时，我们的态制备协议在深度、规模及时空分配方面同时达到最优（相差常数因子）。这一成就此前已由参考文献 [18], 中的态制备方法实现；该方法具有 $\Theta(n)$ 深度和 $\Theta(2^n/n)$量子比特数，以及$\Theta(2^n)$规模及时空分配。

然而，在实践中，可能无法以精确精度执行任意单量子比特门，因此 {U(2)，受控非门} 门集可能并不适用。在这种情况下，我们无法期望精确制备量子态 $|\psi\rangle$ ；相反，给定误差参数 ϵ,~，我们希望制备量子态 $|\psi\rangle$ ，使其满足

$$\big\||\psi\rangle-|\tilde{\psi}\rangle\big\|\le\epsilon\,. \tag{2}$$

例如，大多数基于纠错码的容错量子计算方案都属于这种情况：其中，逻辑单量子比特门是通过从离散门集中选取的一系列逻辑量子门来近似实现的（例如，参见 [22]）。一种常见的离散门集选择是{H门、S门、T门、受控非门}

[1]在全文中，我们使用 $O(f(n,\ 1/\epsilon))$ 记号表示：代价至多为 $f(n,\ 1/\epsilon)$ 的常数倍，其中 $n,\ 1/\epsilon\to\infty$；使用 $\Omega(f(n,\ 1/\epsilon))$ 表示下界；当上下界在常数因子意义下匹配时，使用 $\Theta(f(n,\ 1/\epsilon))$ 表示。

（定义见下一节）；我们引用协议在编译到该门集后进行近似量子态制备时的资源开销标度。我们使用**近似时空分配**和**精确时空分配**这两个术语，分别记为 $\mathrm{SA}_{\mathrm{approx}}$ 和 $\mathrm{SA}_{\mathrm{exact}}$，以区分两种模型中的开销。类似地，我们将两种模型中的电路深度分别记为 $\mathrm{D}_{\mathrm{approx}}$ 和 $\mathrm{D}_{\mathrm{exact}}$。

关键的是，在 {H, S, T, CNOT} 门集中，对于常数目标误差 ϵ，参考文献 [18] 的方法无法达到 $\Theta(n)$ 深度。尽管他们没有明确报告该门集下的深度，但从其量子电路中可以看出这一点：该电路包含 $O(2^n)$ 个任意单量子比特门，分布在 $O(n)$ 个不同的层中；要在引入常数总误差的同时转换到离散门集，每个单量子比特门都必须近似分解为深度至少为 $\Omega(\log(n))$ 的序列，因此总深度至少为 $\Omega(n\log(n))$。即使在离散门集中，当 ϵ 取常数时，我们的态制备方法也能达到 $\Theta(n)$ 深度。

从现实硬件考量的角度来看，优化时空分配的价值也显而易见。例如，在没有纠错的基于量子门的近期设备中，减少时空分配（即使电路规模保持不变）意味着缩短量子比特处于空闲状态的时间。通过在 $|0\rangle$ 中释放并重新初始化辅助量子比特，可以尽量缩短这类空闲时间，并将噪声的影响降至最低。此外，在容错架构中，逻辑量子比特空闲也会带来高昂代价，因为即使不执行任何量子门，也需要持续进行纠错。每一轮纠错都需要一组奇偶校验量子电路和测量，随后还要进行复杂的经典解码计算；所有这些都会增加计算的总体能量消耗。

我们构造的另一个特点是无垃圾。这与一些先前的工作（例如，[23, 24]）形成对比；这些工作执行的是一种较宽松的量子态制备任务：目标不是制备式 n（1）中的 $|\psi\rangle$ 所示的量子比特态，而是制备一个(1)态$(n+a)$量子比特态 $|\Psi\rangle$，其定义如下：

$$|\Psi\rangle = \frac{1}{\|\mathbf{x}\|}\sum_{i=0}^{2^n-1}\left(x_i\,|i\rangle\otimes|\mathrm{garbage}_i\rangle\right). \qquad (3)$$

通常，a 会随 n至少线性增长；这些 a量子比特会与 n 数据量子比特保持纠缠。

允许存在垃圾会使量子态制备任务更容易，而且在某些应用中，垃圾是可以容忍的（因为制备出的量子态仅用作控制量子比特）。然而，只要 a 辅助量子比特仍与数据纠缠，它们就不能作为新的 $|0\rangle$ 辅助量子比特用于其他任务，并且仍会占用量子态制备协议的时空分配。此外，必须强调，这在本质上是一个完全不同的任务。迹掉垃圾后，该 n量子比特态是一个混合态，缺少方程(1)中纯态所具有的相干性。当人们希望直接操纵制备出的量子态（例如，在量子线性系统求解器和量子机器学习应用中），而非仅将其用作控制比特时，这一点尤为重要。

最后，我们指出态制备协议的一个有趣特点：实现对数深度所需的$O(N)$ 辅助量子比特中，有固定比例可以是脏的，也就是说，它们可以从任意量子态开始，并且在量子电路结束时会恢复到起始态。脏量子比特很有吸引力，因为无需耗费资源，以容错方式制备高保真度的 $|0\rangle$ 态，就能使用它们。此外，脏量子比特也可以是来自另一项计算、正经历长时间空闲的量子比特，前提是在空闲期结束之前将它们恢复到原始态。在使用{H门，S 门，T门，受控非门} 门集时，这种情况尤其值得注意，因为我们的态制备电路包含一层：其中一些量子比特处于空闲状态，而另一些则执行单量子比特门序列，以近似实现单量子比特旋转——处于空闲状态的量子比特可用作其他态制备任务的脏辅助量子比特。参考文献 [16], 此前探讨过将T门数量与脏辅助量子比特数量之间的权衡关系。我们认为，通过进一步创新，或许可以将所需干净量子比特的数量进一步降低，使其渐近地优于 $O(N)$。

本工作的贡献如下：

1. 我们提出了一种新颖的、确定性的、无垃圾的量子态制备方法，将其称为 SP+CSP（第 3 节），并证明在 {U(2)，受控非门} 门集中，该方法同时实现了最优深度和最优时空分配。我们还证明，在 {H门，S 门，T门，受控非门} 门集中，其深度和时空分配在渐近意义上均优于以往方法（见表1）。一个匹配的下界表明，该门集中的深度标度是最优的，而关于时空分配的最佳下界仍留下了通过对数因子改进的可能性。

2. 我们展示了本协议的最优时空分配如何使其能够比其他方法更快地制备某个量子态的多个副本（第4节）。

3. 我们讨论了最优时空分配量子态制备协议的若干应用（第5节），包括

   (a) 量子机器学习
   (b) 哈密顿量模拟
   (c) 使用 HHL 风格求解线性方程组 e
   量子算法

4. 我们提供了 SP 的量子电路级实现+CSP 方法（第3 和附录 A），并附有详细的伪代码。

5. 我们使用 Braket 提供了两个门级实现示例（第 6节）。

我们在表 图1 中总结了我们的结果与其他最先进结果的比较，并在图 图1 中以图示呈现。

为实现这些结果，我们借鉴了 Clader 等人 [20], 的工作。他们提出了一种量子信号处理方法，在假设 FANOUT-CNOT 操作 —— 即最多 $O(2^n)$ 个共享同一控制量子比特的 CNOT 门的乘积——能够在一个时间步内完成的情况下，实现深度 $O(n)$ 和量子比特数 $O(2^n)$。我们的协议（称为 SP 协议）无需 FANOUT-CNOT，代价仅是在辅助量子比特数量和深度上增加常数因子。其核心思路是使用仅由 CNOT 门构成的树状数据复制电路，将控制比特复制到许多辅助量子比特中，从而可以并行控制多个操作（见附录 A.1）。然而，仅有这一观察还不足以实现 $O(n)$ 的总体深度；我们还需要一种精巧的方法，在控制比特复制层与受控操作层之间交替进行（见附录 A.4）。与 Clader 等人的电路类似，该电路的特点是仅有常数层涉及单量子比特旋转（将其近似分解为有限门集时会产生深度 $O(\log(n/\epsilon))$），因此总体深度可按 $O(n+\log(1/\epsilon))$ 的规模增长。相比之下，以前的方法（例如 [17, 19]）会将这些单量子比特旋转分配到至少 n 层中，使总深度至少达到 $\Omega(n\log(n/\epsilon))$。

SP 协议（以及先前工作中设计的协议，例如 [17] 和 [19]）未能实现最优的时空分配，因为它从根本上要求 $O(2^n)$ 个辅助量子比特与$O(n)$ 个数据量子比特纠缠，且这种状态需持续整个 $O(n)$ 电路深度中一个固定比例的时间，因此导致 $O(n2^n)$ 的时空分配（与图 1 左侧所示情况类似）。为避免这一问题，我们采用了另一种思路：先用 SP 方法制备大约一半的量子比特，再从这些量子比特向其余量子比特执行受控态制备（CSP）。SP 和 CSP 都只需要 $O(1)$ 层单量子比特旋转，因此近似编译的深度仍为 $O(n+\log(1/\epsilon))$。此外，在 CSP 过程中，只有极短时间需要全部 $O(2^n)$ 个辅助量子比特；仅经过 $O(1)$ 深度后，大部分辅助量子比特就可以释放（如图 1 右侧所示）。最终，我们证明可以实现 $O(2^n)$ 的时空分配。我们还在附录 A.7 中给出了详细的量子电路实现，展示了时空分配需求的差异。

2 背景

本节将介绍所需的量子门，并进一步说明时空分配度量，以及相关下界和相关先前工作的概述。

2.1 量子门

在本工作中，我们将使用以下量子门：X门、S门、T门、Hadamard（H）门、R_y 门、CNOT门、SWAP门、受控-R_y 门、Fredkin 门和Toffoli门。

Gate	酉算子
X	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Hadamard（H）	$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$
Y旋转 ($R_y(\theta)$)	$\begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$
CNOT	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
S	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
T	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$

表2：基本量子门。注意存在一些冗余，例如 $S = T^2$ 和 $X = HS^2H$。 R_y 门可以通过一系列H、S和 T 量子门以任意精度近似（参见例如 [22]）

表2 展示了 {{U(2),CNOT} 门集和 {H,S,T,CNOT} 门集所需的基本量子门。我们还在图2、3、4和5中汇总了所有其他可构建的所需量子门。

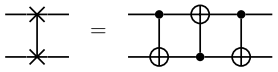


图2：使用受控非门进行 SWAP 门分解

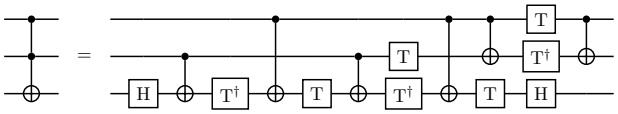


图3：使用 H门、T门和受控非门进行 Toffoli（受控受控非门）门分解 [25]

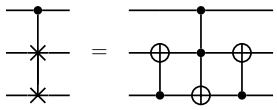


图4：使用受控非门、Toffoli 门分解 Fredkin（受控交换门）[26]

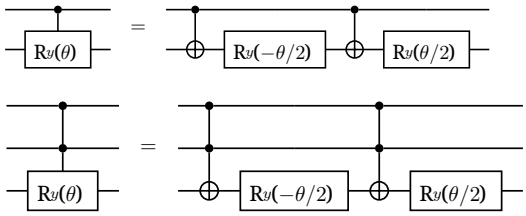


图5：受控旋转门（CR$_y$ 和CCR$_y$）使用 R$_y$、受控非门、Toffoli 的分解

2.2 时空分配——量子处理单元“核心小时”

在经典计算中，“核心小时” [27]是指用于运行某项计算任务的中央处理器数量与该作业持续时间（以小时计）的乘积。现将经典“核心小时”的量子对应概念定义为量子时空分配成本，等价地可通过以下任一种方式定义：

- 每个逻辑量子比特处于活动状态（即不处于 $|0\rangle$ 态）的持续时间（深度）之和
- 每一层中活动量子比特数量之和

或者定量地表示为：

$$\mathrm{SA} := \sum_{i=0}^{q-1} d_i = \sum_{t=0}^{d-1} q_t \tag{4}$$

其中， d 是总深度， q 是量子比特总数， d_i 是 i^{th} 量子比特的活跃时间（深度）， q_t 是第 t层中的活动量子比特数量。

如果所有量子比特在整个计算过程中都处于活动状态，那么时空分配就是电路深度与量子比特总数的乘积。然而，如果大多数辅助量子比特只需短暂使用，并且可以在计算结束前重置为 $|0\rangle$ 态，那么时空分配可能会显著小于 [28]。在本工作中，我们将证明，在早期释放辅助量子比特可以为量子态制备带来**渐近的**时空分配优势。

2.3 下界

已有研究表明，在仅使用任意单量子比特门和 CNOT 门时，量子态制备在量子电路深度、规模、量子比特数及其相应权衡关系方面存在下界 [17, 29]

由于量子电路中的每个量子门至少作用于一个量子比特，因此每一层中每个量子门至少需要一个处于活动状态的量子比特。由此，量子门数量为时空分配提供了一个下界。

引理2.1。 如果一个量子电路的电路规模为C，则其时空分配至少为C。

因此，如果我们希望对时空分配给出下界，只需对电路规模给出界即可。任意量子态由 $O(2^n)$ 个实参数描述。每个单量子比特门最多可以引入 $O(1)$ 个实参数，因此至少需要$\Omega(2^n)$个单量子比特门。此外，同一量子比特上任何不相邻的单量子比特门之间若没有受控非门，则可以将它们合并为一个量子门，因此受控非门的数量必须与单量子比特门的数量处于同一数量级。这一推理得出了电路规模的下界 [29]。

引理2.2 (第 II 节，见 [29])。 由 *{*U(2),受控非门*}* 中的量子门构成、能够制备任意 n 量子比特量子态的量子电路，必须至少包含 $\Omega(2^n)$ 个受控非门量子门，以及至少包含 $\Omega(2^n)$ 个 U(2)量子门。

推论2.2.1。 由 *{*U(2),受控非门*}* 中的量子门构成、能够制备任意 n 量子比特量子态的量子电路，必须具有至少 $\Omega(2^n)$ 的时空分配。

在{H门，S 门，T门，受控非门} 门集中，也可以得到类似的下界。

引理 2.3。 由 *{*H门,S 门, T门, 受控非门*}* 中的量子门构成、能够以误差 ϵ 制备任意 n 量子比特量子态的量子电路，其规模必须至少为$\Omega\left(n+2^n\log(1/\epsilon)\log(\log(1/\epsilon))\right)$。

证明。 我们沿用[30]定理 9 中的类似推理。根据 [30], 中的引理 1，C 电路规模 C 必须满足 $C\log(\ \geq)\ \Omega(2^n\log(1/\epsilon))$。令$U = C\log(C)$，并对两边取对数，得到 $\log(U) = \log(C\log(C)) \geq \log(C)$，因此 $C = U/\log(C) \geq U/\log(U)$。于是，$U \geq \Omega(2^n\log(1/\epsilon))$ 意味着 $C \geq \Omega(2^n\log(1/\epsilon)/(n+\log(\log(1/\epsilon))))$。□

请注意，参考文献 [16] 指出，T 门的数量可能远小于这个下界。有趣的是，在许多容错量子计算方法中，T 门的成本显著高于其他量子门。

推论 2.3.1。由 *{*H门、S 门、T门、受控非门*}* 中的量子门组成的量子电路，若要将任意 n 量子比特态制备至误差 ϵ，则必须至少具有$\Omega\left(n+2^n\log(1/\epsilon)\log(\log(1/\epsilon))\right)$ 的时空分配。

尚不清楚引理2.3和推论2.3.1中的下界是否紧。

在两种门集中，也可以证明深度的下界。

引理 2.4 （ [17]的定理 3）。一个由以下门集中的量子门组成的量子电路{U（2）、受控非门}，若要制备任意的 n量子比特态，并使用 q 个辅助量子比特，则其深度至少为 $\Omega(\max(n, \frac{2^n}{q+n}))$。

引理 2.5。一个由门集 {H门、S 门、T门、受控非门} 中的量子门组成的量子电路，若要制备任意的 n量子比特态，并使用 O（2^n）个辅助量子比特，则其深度至少为 $\Omega(n+\log(1/\epsilon))$。

证明。首先，对于任意常数 $\epsilon < 1$，我们考察对 n 的依赖关系。按照 [17, 在深度 D 时的定理 3]，所采用的类似论证，位于 n 数据量子比特“光锥”中的量子门数量上界为 $n2^D$。其余量子门可以忽略。由于门集中包含的量子门数量为常数，唯一输出态的总数上界为 $e^{O(n2^D)}$。与此同时，参考文献 [31] 的公式（4.85）断言：为了以 $\epsilon = O(1)$ 精度覆盖全部 n 量子比特态，所需的量子电路数量至少为 $e^{\Omega(2^n)}$。综合起来，这些结果意味着$D = \Omega(n)$。另一方面，一旦设定 $n_{anc} = O(2^n)$，我们可以直接根据 [30]，的定理 9，将对 ϵ 的依赖关系下界为 $D = \Omega(\log(1/\epsilon))$。当 n 足够大且 ϵ 足够小时，这两个下界彼此独立地成立，因此下界之和也必须成立（相差一个可吸收到大 Ω 记号中的常数因子），从而得到总体所述的下界 $\Omega(n+\log(1/\epsilon))$。 □

SP+第图3 中所示的 CSP 电路构造将给出一个与引理2.5的下界相匹配的上界。

2.4 先前工作的上界

2.4.1 {U(2)，受控非门} 门集

据我们所知，最先进的量子态制备 [17, 18] 方法可达到以下深度，其中 a 是可用辅助量子比特的数量：

$$\mathrm{D}_{\text{exact}} = \begin{cases} \Theta(\frac{2^n}{n}), & \text{if } a = 0 \text{ (need } n \text{ data qubits)} \\ \Theta(\frac{2^n}{n+a}), & \text{if } a = O(2^n/n) \\ \Theta(n), & \text{if } a = \Omega(\frac{2^n}{n}) \end{cases}$$

此处，$\mathrm{D}_{\text{exact}}$ 表示使用{U(2), 受控非门}门集的精确量子态制备电路的深度，a 表示辅助量子比特的数量，而 n 表示待制备任意量子态的量子比特数。

我们可以通过将深度与所需的量子比特数量相乘，直接计算时空分配上界；我们发现，下界 $\Omega(2^n)$ 对任意数量的辅助量子比特$a < O(2^n/n)$ 都能达到，特别是：

$$\mathrm{SA}_{\text{exact}} = \begin{cases} \Theta(2^n), & \text{if } \mathrm{D}_{\text{exact}} = \Theta(\frac{2^n}{n}) \\ \Theta(2^n), & \text{if } \mathrm{D}_{\text{exact}} = \Theta(n) \end{cases}$$

有趣的是，Sun 等人 [17] 还从图论角度证明，对于使用任意单量子比特门和双量子比特门的电路，其深度下界为$\Omega(n)$，且与辅助量子比特的数量无关。在第3节中，我们将展示，我们的 SP+CSP 态制备协议也能实现 $\Theta(n)$ 深度和$\Theta(2^n)$ 时空分配。这提供了一种不同于参考文献 [18], 中构造的方案，且同样是最优的。

2.4.2 {H、S、T、CNOT} 门集

采用{H、S、T、CNOT}门集编译为量子门后，先前的方法达到了以下深度，其中 a 为可用辅助量子比特的数量： [17, 19]

$$\mathrm{D}_{\text{approx}} = \begin{cases} O(\frac{2^n \log(2^n/\epsilon)}{n}), & \text{if } a = 0 \\ O(n\log(n)\log(2^n/\epsilon)), & \text{if } a = \Theta(\frac{2^n}{n\log(n)}) \\ O(n\log(n/\epsilon)), & \text{if } a = \Omega(2^n) \end{cases}$$

我们还可以将量子电路深度乘以所涉及的量子比特数量，来计算时空分配成本：

$$\mathrm{SA}_{\text{approx}} = \begin{cases} O(2^n \log(2^n/\epsilon)), & \text{if } a = 0 \\ O(2^n \log(2^n/\epsilon)), & \text{if } a = \Theta(\frac{2^n}{n\log(n)}) \\ O(n2^n \log(n/\epsilon)), & \text{if } a = \Omega(2^n) \end{cases}$$

我们还注意到，Yuan 等人 [18] 提出的方法至少需要 $\Omega(n\log(n/\epsilon))$ 的深度和 $\Omega(2^n\log(n/\epsilon))$ 的时空分配。论文中没有明确说明这一点，但可以从以下事实推导出来：单量子比特旋转出现在$O(n)$个不同的层中，而在离散门集中，每个旋转至少需要 $\Omega(\log(n/\epsilon))$ 的深度。

此外，Clader 等人 [20] 提出了一种深度为 $O(\log(2^n/\epsilon))$ 且时空分配为 $O(2^n\log(2^n/\epsilon))$的方法，但该方法假设可以在单位时间内执行 FANOUT-CNOT。将具有 2^n 个目标的 FANOUT-CNOT 分解为双量子比特 CNOT 门，会使协议深度产生 $O(n)$ 倍的开销，并对时空分配造成类似的开销（见表1）。

在本文中，我们将证明，我们的方法在 [20] 门集{H门、S 门、T门、受控非门}中实现了参考文献所述的深度标度（即无需 FANOUT-CNOT），同时实现了更优的时空分配O（$2^n \log(n/\epsilon)$）。

图3 SP+CSP态制备

本节将详细介绍我们的 SP+CSP 方法，该方法实现了 $\Theta(N)$ 时空

分配，同时使用 {U(2)、受控非门} 门集，保持 $\Theta(\log(N))$ 深度。

回想一下，我们希望制备式 n-量子比特 $|\psi\rangle$（(1)）中已知系数为 $x_i/\|\mathbf{x}\|$ 的计算基态。我们提出采用以下方法：首先对部分数据量子比特执行态制备步骤（**SP**），然后执行受控态制备步骤（**CSP**）。之所以采用这种方法而非直接 SP，是因为如果设置得当，SP+CSP 协议能够兼具 SP 和 CSP 步骤的优点，同时避免它们的缺点。本节后面的第3.3节将详细说明其复杂度方面的优缺点。

SP+CSP 协议的基本思路是将 $N = 2^n$ 基态划分为 $M = 2^m$个互不重叠的集合，每个集合的大小为 $\frac{N}{M}$。前 m 位相同（以二进制表示）的计算基态被放入同一集合。将这M个集合记为 J_i，其中 $i = 0$、……、$M-1$。对于每个 i，令 $y_i = \sqrt{\sum_{j\in J_i}|x_j|^2}$。

我们首先制备 m 个量子比特的量子态（$m < n$）：

$$U_{\mathrm{SP}}|0^m\rangle = |\phi\rangle = \frac{1}{\|\mathbf{y}\|}\sum_{i=0}^{2^m-1} y_i|i\rangle \tag{5}$$

现在总量子态为 $|\phi\rangle \otimes |0^{n-m}\rangle$。

接下来，我们执行受控态制备操作 U_{CSP}，其定义为

$$U_{\mathrm{CSP}}(|i\rangle \otimes |0^{n-m}\rangle) = y_i^{-1}\sum_{j\in J_i} x_j|j\rangle \tag{6}$$

对于被控制的特定 $|i\rangle$ 量子态，有

$$U_{\mathrm{CSP}}(|\phi\rangle \otimes |0^{n-m}\rangle) = |\psi\rangle = \frac{1}{\|\mathbf{x}\|}\sum_{j=0}^{2^n-1} x_j|j\rangle \tag{7}$$

电路图如图6所示。的实现细节 U_{SP} 和 U_{CSP} 如图7和图8所示。

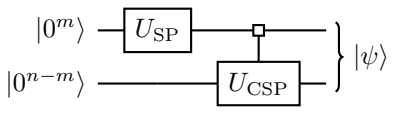


图6：SP+CSP 协议的量子电路。该协议通过首先制备一个 m 个量子比特的量子态 $|\phi\rangle$，然后对最终的 $n-m$ 个量子比特执行受控态制备，从而实现任意 n 个量子比特量子态 $|\psi\rangle$ 的态制备。方形控制位 ◘ 表示：对于前 m 个量子比特的每种取值，都会在最终的 $n-m$ 个量子比特上执行不同的操作。

现在我们将介绍如何执行 U_{SP} andU_{CSP} 电路。

3.1 SP 电路结构

执行 U_{SP} 时，我们给出一种深度为 $O(m)$、使用 $O(2^m)$ 个辅助量子比特的方法。因此，时空分配至多为 $O(m2^m)$。如果要求使用离散门集，且近似误差为 ϵ，则该方法可实现深度为 $O(m+\log(1/\epsilon))$、时空分配为 $O(m2^m + 2^m\log(m/\epsilon))$。

SP 协议的构想沿袭先前文献，首先定义 $M-1$ 角度，这些角度均可根据振幅列表高效地通过经典计算得出$\{y_i\}_{i=0}^{M-1}$。为方便起见，并与先前文献保持一致 [8], 我们采用二索引角度定义来定义旋转角：

$$\theta_{s,p} = 2\cos^{-1}\left(\frac{\sqrt{\sum_{l=0}^{2^{m-s-1}-1}|y_{p\cdot 2^{m-s}+l}|^2}}{\sqrt{\sum_{l=0}^{2^{m-s}-1}|y_{p\cdot 2^{m-s}+l}|^2}}\right), \tag{8}$$

其中 $s = 0$、……、$m-1$ 以及 $p = 0$、……、2^s-1，总计有 $1+2+4+\ldots+2^{m-1} = M-1$ 个角度。直接计算每个角度可能需要查询向量 y 中多达 $O(2^m)$ 个条目，因此经典计算的总运行时间为 $O(2^{2m})$。然而，我们可以将量 $S_{s,p} = \sum_{l=0}^{2^{m-s}}|y_{p\cdot 2^{m-s}+l}|^2$ 存储在具有 m 层 [3] 的二叉树数据结构中，从而重复利用部分计算。该树有 2^{m-1} 个叶节点；我们将量 $S_{m-1,p}$ 存储在叶节点中，分别对应 $p = 0$、……、$2^{m-1}-1$。随后按照以下规则递归构建该树：父节点存储其子节点之和。由于 $S_{s,p} = S_{s+1,2p} + S_{s+1,2p+1}$，我们可以验证，层级为 s 的 p^{th} 节点将存储值 $S_{s,p}$。因此，只需利用该节点及其两个子节点所存储的值，即可确定角度 $\theta_{s,p}$。构建整棵树的总工作量为 $O(2^m)$。此外，如果向量 y 中的某个条目被修改，或者条目以在线方式到达，沿着从叶节点到根节点的一条路径更新二叉树数据结构，经典计算只需 $O(m)$ 时间。虽然我们并未利用这一性质，但当 y 为稀疏向量且仅有 d 个非零条目时，树结构也将是稀疏的，且最多有 dm 个非零条目。

这种二索引角度定义在标记存储计算所得角度的辅助寄存器时，也会带来更多便利，相关角度如公式(9) 所示，以及相应的数据量子比特。具体而言，s 对应辅助寄存器的索引 A_s（例如，算法 图5中引用的寄存器），同时也对应正在处理的数据量子比特的索引。与此同时，p 对应特定辅助寄存器 A_s中量子比特的索引。

Araujo 等人 [24] 提出了一种低深度策略，用于实现 U_{SP} 使用 $O(M)$ 个辅助量子比特。其总体思路是通过预先制备状态来对多个角度进行“预旋转”——

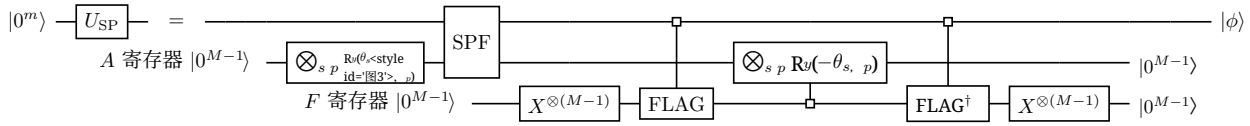


图7：实现 U_{SP} 的量子电路，该电路从量子态 $|0^m\rangle$ 出发，制备任意的 m 量子比特态，并借助 $O(2^m)$ 个辅助量子比特；这些辅助量子比特始终处于 $|0\rangle$态。令 $M=2^m$，并说明受控旋转门表示 $M-1$ 个受控旋转同时进行，且旋转角度各不相同，分别为 $\theta_{s,\ p}$，其中 $s=0,\ \ldots\ldots,\ m-1$，以及 $p=0,\ \ldots\ldots,\ 2^s-1$。请注意，附录中给出的 SPF 和 FLAG 的实现还需要 $O(M)$ 个未显示的额外辅助量子比特。

制备状态

$$|\theta_{s,p}\rangle = \mathrm{R}_y(\theta_{s,p})\,|0\rangle = \cos\left(\frac{\theta_{s,p}}{2}\right)|0\rangle + \sin\left(\frac{\theta_{s,p}}{2}\right)|1\rangle \tag{9}$$

对每个 $s=0,\ \ldots\ldots,\ m-1,\ \ p=0,\ \ldots\ldots,\ 2^s-1$，并高效地将其中一部分量子态注入数据量子比特。注入哪些量子态由数据量子比特自身控制，这使数据量子比特与一个大型垃圾寄存器纠缠在一起。Clader 等人 [20] 展示了如何利用标志机制对垃圾进行反计算。他们证明，注入步骤（例如 SPF 和 FLAG）的总 T深度为$O(m)$，但并未研究 Clifford 深度。若坚持只使用双量子比特 Clifford 门，朴素计算会表明 Clifford 深度至少为 $\Omega(m^2)$，这是由于存在 FANOUT-CNOT 门。本工作通过利用位复制机制，给出了参考文献 [20] 所述方法的改进版本，从而确保 Clifford 深度也为 $O(m)$。

完成 U_{SP}例程的高级量子电路如图图7所示。该量子电路描述了制备任意 m量子比特态 $|\phi\rangle$ 的量子门序列，并借助 $2M-2=2^{m+1}-2$个辅助量子比特；这些辅助量子比特初始为[2] 态，最终也回到 $|0\rangle$。请注意，附录A中讨论的 SPF 电路和 FLAG 电路实现，需要额外的 $O(M)$ 个辅助量子比特来实现上述提到的位复制机制，图中未显示这些辅助量子比特。

现在我们遵循参考文献 [20]，定义图图7中 SPF 和 FLAG 量子电路的作用。首先，定义乘积态

$$\begin{aligned} |\Theta_s\rangle &= \bigotimes_{p=0}^{2^s-1} |\theta_{s,p}\rangle \qquad \text{for } s=0,1,\ldots,m-1 \\ |\Theta\rangle &= \bigotimes_{s=0}^{m-1} |\Theta_s\rangle\ . \end{aligned} \tag{10}$$

接下来，对于每个 $j=0,\ \ldots,\ M-1,\ s=0,\ \ldots,\ m-1, p=0,\ \ldots,\ 2^s-1$，定义

$$f_{(s,p)|j} = \begin{cases} 1 & \text{if } p = j \bmod 2^s \\ 0 & \text{otherwise} \end{cases}\ . \tag{11}$$

[2]请注意，图 A 7 中的 寄存器中的辅助量子比特只有在输入 $|0\rangle$ 为 U_{SP} 时 $|0^m\rangle$，才能保证最终处于该状态。对于其他输入态，该寄存器最终可能与数据量子比特纠缠。关于为何这一特性不会破坏我们的协议在输入并非总是B 时的应用最优性，请参见附录 $|0^m\rangle$。

因此，如果固定 j 的值，则存在 m 对 $(s,\ p)$ 满足 $f_{(s,p)|j}=1$：事实上，对于 s 的每个取值，恰好存在一个 $p\in[0,2^s-1]$ 满足 $f_{(s,p)|j}=1$。满足 $f_{(s,p)|j}=1$ 的 $(s,\ p)$ 取值对应于角度态$|\theta_{s,\ p}\rangle$；对于数据量子比特的设定 $|j\rangle$，这些角度态由 SPF 电路注入数据中。

该 U_{SP} 过程首先将 $|\Theta\rangle$ 制备到一个辅助“角度”寄存器（图 A 中的寄存器图7）中，该寄存器的大小为 $M-1$，并使用 $M-1$ 并行的Ry 门。随后施加 SPF 电路，将部分角度态注入数据，并产生如下作用

$$\mathrm{SPF}\left(|0^m\rangle\,|\Theta\rangle\right) = \frac{1}{\|\mathbf{y}\|}\sum_{j=0}^{M-1} y_j\,|j\rangle\,|g_j\rangle\ . \tag{12}$$

其中

$$|g_j\rangle = \bigotimes_{s=0}^{m-1}\bigotimes_{p=0}^{2^s-1} |\theta_{s,p}(1-f_{(s,p)|j})\rangle\ . \tag{13}$$

这里，记号 $|\theta(1-f)\rangle$ 表示当 $f=0$ 时，量子态为 $|\theta\rangle$；当 $f=1$时，量子态为 $|0\rangle$。换言之，对于满足 $f_{(s,p)|j}=1$ 的寄存器，其所持有的$|\theta_{s,p}\rangle$ 会被替换为 $|0\rangle$。我们看到，SPF 的输出具有与目标态 $|\phi\rangle$相同的振幅，但数据仍与一个（M–1）量子比特的垃圾寄存器纠缠在一起。

FLAG 操作将位 $f_{(s,p)|j}$ 计算到一个新的辅助量子比特“标志”寄存器[3]（寄存器 F，见图 7），对每一对 (s,p)执行该操作，即：

$$\mathrm{FLAG}(|j\rangle\,|1^{M-1}\rangle) = |j\rangle\bigotimes_{s=0}^{m-1}\bigotimes_{p=0}^{2^s-1}|1-f_{(s,p)|j}\rangle \tag{14}$$

同时保持角度量子比特上的垃圾态不变。因此，在执行 FLAG 操作后，对处于量子态$|\theta_{s,p}(1-f_{(s,p)|j})\rangle$ 的角度量子比特施加由标志位（$1-f_{(s,p)|j}$）控制的 controlled-R$y(-\theta_{s,p})$ 量子门，即可将所有角度量子比特重置回 $|0\rangle$。对每一对 (s,p) 并行执行此操作。最后，可对 FLAG 操作施加伴随操作（FLAG†），以反计算标志位，并使标志寄存器恢复到 $|0^{M-1}\rangle$。这一过程的输出是量子态 $|\phi\rangle$，不含任何垃圾。

[3]SPF 量子电路之后需要采用标志机制，这正是 SPF 量子电路名称中字母 F 的由来（SP 代表“态制备”），LOADF 量子电路（将在下一节介绍）的名称也由此而来。

参考文献 [20] 中介绍的 SPF 和 FLAG 构造仅针对T门深度和 T门计数进行了优化，并未明确研究 Clifford 计数和深度。此外，该构造假设具有任意数量目标的 FANOUT-CNOT 门可在一个时间步内完成。在附录中，我们详细介绍了 SPF（第 A.4节）和 FLAG（第 A.5节）子程序的构造，并说明这两个子程序如何使用 $O(m)$ 深度和 $O(m2^m)$ 时空分配，在使用 {U(2)、CNOT} 门集时实现，并以近乎最优的代价使用{H门、S 门、T门、CNOT} 门集。

我们在下方的算法1中记录了 SP 电路实现的伪代码。

算法 1 SP 电路（见图7）

过程 SP（m, D, A, F）
▷ D：大小为 m 的数据寄存器
▷ A：大小为 $M-1$ 的角度寄存器
▷ F：大小为 $M-1$ 的标志寄存器
  对于 s，**遍历** range(m)，**并对于** p，**遍历** range(2^s)，**执行**
    经典地计算 $\theta_{s,\,p}$ 从 $\mathbf{y}$ ▷ 公式（8）
  循环结束
  对于 s，**遍历** range(m)，**并对于** p，**遍历** range(2^s)，**执行**
    将 $\mathrm{R}_y(\theta_{s,\,p},\ A_{s,\,p})$ 作用于新的“角度”辅助量子比特
  结束循环
  SPF$(D,\ A,\ m)$ 子程序 ▷ $O(m)$
  对于 s**in** range(m) **和** p **in** range(2^s)，**执行**
    将 X门$(F_{s,\,p})$ 作用于新的“标志”辅助量子比特
  结束循环
  FLAG$(D,\ F,\ m)$ 子程序 ▷ $O(m)$
  对于 s，**范围为** range(m) **且** p，**范围为** range(2^s)，**执行**
    $\mathrm{CR}_y(-\theta_{s,\,p},\ F_{s,\,p},\ A_{s,\,p})$
  结束循环
  FLAG$^\dagger(D,\ F,\ m)$ 子程序 ▷ $O(m)$
  对 s，**遍历** range(m)，**并对** p，**遍历** range(2^s)，**执行**
    对 X$(F_{s,\,p})$ 执行操作，将“flag”辅助量子比特重置为 $|0\rangle$
  结束循环
结束过程
▷ 总 $\mathrm{D}_{\mathrm{exact}}$：$O(m)$
▷ 总 $\mathrm{D}_{\mathrm{approx}}$：$O(m+\log(m/\epsilon))$
▷ 总 $\mathrm{SA}_{\mathrm{exact}}$：$O(m2^m)$
▷ 总 $\mathrm{SA}_{\mathrm{approx}}$：$O(m2^m+2^m\log(m/\epsilon))$

3.2 CSP 电路结构

受控态制备电路为每个 $n-m$ 可能的设置准备一个不同的量子比特态 M $|k\rangle \in \{|0\rangle \ldots, |M-1\rangle\}$，这些设置属于一个具有 m量子比特的控制寄存器。因此，对于每个 k，都有 $2^{n-m}-1=\binom{N}{M}-1$个角度，记为 $\theta_{s,\,p}^{(k)}$，其中 $s=0,\ldots,n-m-1$ 以及 $p=0,\ \ldots,\ 2^s-1$。这些角度可以根据$\mathbf{x}$ 的振幅，通过以下方程计算得到（参见方程(8)）

$$\theta_{s,p}^{(k)} = 2\cos^{-1}\left(\frac{\sqrt{\sum_{l=0}^{2^{n-m-s-1}-1}|x_{k2^{n-m}+p\cdot 2^{n-m-s}+l}|^2}}{\sqrt{\sum_{l=0}^{2^{n-m-s-1}}|x_{k2^{n-m}+p\cdot 2^{n-m-s}+l}|^2}}\right). \tag{15}$$

角度总数因此为 $N-M$。量子态$|\theta_{s\,p,}^{(k)}\rangle$ 定义如下方程(9)，而乘积态 $|\Theta_s^{(k)}\rangle$ 和 $|\Theta^{(k)}\rangle$ 定义如下方程(10)。我们在算法2中记录了 CSP 电路实现的伪代码。

算法 2 CSP 电路（见图8）

过程 CSP$(m,\ n,\ D,\ B,\ F)$
▷ D：大小为 $m-n$ 的数据寄存器
▷ B：大小为 $\binom{N}{M}-1$ 的缓冲寄存器
▷ F：大小为 $\binom{N}{M}-1$ 的标志寄存器
  对于 k，**在** range(2^m) **中执行**
    对于 s，**在** range$(n-m)$ **以及** p，**在** range(2^s) **do**
      经典计算 $\theta_{s,\,p}^{(k)}$，由 $\mathbf{x}$ ▷ 得出。使用公式（图3）
    结束循环
  结束循环
  对于 s，**遍历** range$(n-m)$，**且** p **遍历** range(2^s)，**执行**
    将 X门$(F_{s,\,p})$ 作用于新的“标志”辅助量子比特
  结束循环
  LOADF$(D,\ B,\ F)[\theta_{s,p}^{(k)}]$ 将角度加载到缓冲区辅助量子比特 ▷ $O(n)$
  SPF$(D,\ B,\ n-m)$以制备 $n-m$ 量子比特态带有垃圾 ▷ $O(n-m)$
  FLAG$(D,\ F,\ n-m)$ ▷ $O(n-m)$
  LOADF$^\dagger(D,\ B,\ F)[\theta_{s,p}^{(k)}]$ 以卸载缓冲区中的角度缓冲辅助量子比特 ▷ $O(n)$
  FLAG$^\dagger(D,\ F,\ n-m)$ ▷ $O(n-m)$
  对于 s，**取** range$(n-m)$；**并且对于** p，**取** range(2^s)，**执行**
    对 X门$(F_{s,\,p})$ 执行以重置“flag”辅助量子比特
  结束循环
结束过程
▷ 总深度 $\mathrm{D}_{\mathrm{exact}}$：$O(n)$
▷ 总深度 $\mathrm{D}_{\mathrm{approx}}$：$O(n+\log(n/\epsilon))$
▷ 总空间-面积 $\mathrm{SA}_{\mathrm{exact}}$：$O(2^n)$
▷ 总空间-面积 $\mathrm{SA}_{\mathrm{approx}}$：$O((n-m)2^{n-m}+2^n\log(n/\epsilon))$

图 8所示的受控态制备电路，是 Clader 等人提出的受控态制备电路的一个推广版本。 [20]CSP 电路的总体思路是，首先，在控制寄存器为 $|k\rangle$的条件下，载入正确的量子态 $|\Theta^{(k)}\rangle$。完成后，应用与 U_{SP} 协议相同的 SPF 电路，将正确的角度注入 $n-m$ 数据量子比特。随后，采用与 U_{SP} 协议类似的 FLAG 标志机制，使数据与角度寄存器解纠缠，并将[4] 角度寄存器重置为 $|0^{M-1}\rangle$。

加载和卸载通过一种我们称为 LOADF 的量子电路实现，它类似于参考文献 [20], 中定义的 LOADF（但并不完全相同）

[4]请注意，只有当B（图8中的寄存器）的输入为 $|0^{M-1}\rangle$时，才能保证角度寄存器最终处于 U_{CSP} 的输入为 $|0^{n-m}\rangle$。参见脚注2 和附录B。

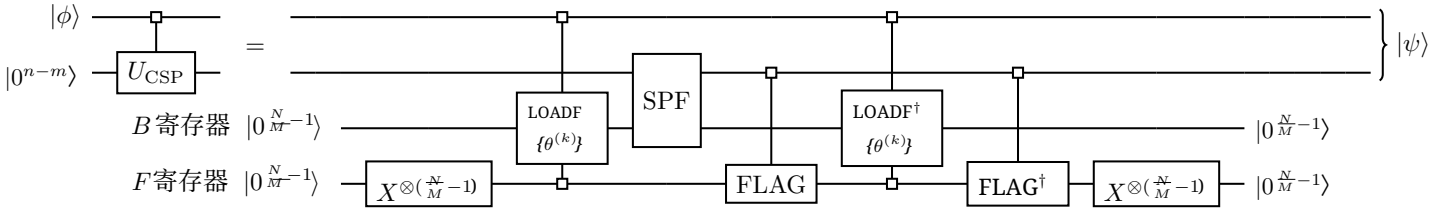


图8：实现 U_{CSP} 操作的量子电路。该操作针对m量子比特控制寄存器的每种设置，为每个 $n-m$ 量子比特态制备任意量子态，并借助 $O(2^{n-m})$ 个辅助量子比特；这些辅助量子比特起始和最终均处于 $|0\rangle$态。令 $M=2^m$ 和 $N=2^n$。注意，附录中给出的 LOADF、SPF 和 FLAG 的实现涉及 $O(N)$ 个额外的未显示辅助量子比特。我们还注意到，在实际实现中，可以将第一个 LOADF 操作简化为仅由 B 寄存器控制（而不是由 F寄存器控制；在量子电路的这一阶段，F寄存器保证处于 $|1^{N/M-1}\rangle$ 态），从而节省常数级深度。我们选择以这种方式定义 LOADF 算子，只是为了避免再次定义后续的 LOADF†。

作用

$$\begin{aligned}&\mathrm{LOADF}\left(|k\rangle\,|0^{N/M-1}\rangle\left[\bigotimes_{s=0}^{m-n-1}\bigotimes_{p=0}^{2^s-1}|f_{s,p}\rangle\right]\right)\\&=|k\rangle\left[\bigotimes_{s=0}^{m-n-1}\bigotimes_{p=0}^{2^s-1}|f_{s,p}\theta_{s,p}^{(k)}\rangle\right]\left[\bigotimes_{s=0}^{m-n-1}\bigotimes_{p=0}^{2^s-1}|f_{s,p}\rangle\right]\end{aligned}\tag{16}$$

请注意，如果所有标志位$f_{s,\ p}$ 均置为 1，量子态 $|\Theta^{(k)}\rangle$ 就会被制备到第二个寄存器中。我们的 LOADF 实现见附录中的图19 ，其中还需要额外的 $O(2^n)$ 辅助量子比特。

3.3 总体深度和时空分配

在本小节中，我们计算协议的总体深度和辅助量子比分配。为验证所述子程序的复杂度，请读者参阅实现这些子程序的章节。我们首先在 {U(2)、受控非门} 门集中陈述复杂度，然后在复杂度不同时，于{H门、S 门、T门、受控非门} 门集中给出相应的复杂度，并将其置于（括号）内。我们在表图3中汇总各项复杂度贡献及所得的总复杂度。

请注意，要使态制备或受控态制备的整体误差达到 ϵ ，只需将每个角度态$|\theta_{s,\ p}\rangle$ 制备到精度 ϵ/n。这是因为该协议等价于在数据量子比特上依次执行 n 多重受控旋转。若每个角度的误差均不超过 ϵ/n，则每个n 酉算子的实现误差在谱范数下不超过 ϵ/n，因此总误差至多为 ϵ。更详细的论证见 [20, 补充材料第五节 B] 和 [19,补充材料第八节 A]。

我们从 U_{SP}开始，其实现方式如图7所示。并行旋转门和平行受控旋转门可在深度 $O(1)$ ($\mathrm{D}_{\mathrm{approx}}=O(\log(m/\epsilon))$内完成，而 SPF 和 FLAG 电路的深度为 $O(m)$，因此总深度为 $O(m)$（总计[5]$\mathrm{D}_{\mathrm{approx}}=O(m+\log(1/\epsilon))=O(\log(2^m/\epsilon))$）。所需的总量子比特数为图7中出现的 $m+2M-2$，再加上执行 SPF 和 FLAG 操作所需的额外 $O(M)$。空间与深度的乘积直接给出时空分配的上界 $O(m2^m)$（$\mathrm{SA}_{\mathrm{approx}}=O(2^m\log(2^m/\epsilon))$；事实上，对于 U_{SP}，存在一个匹配的下界（相差一个常数因子）。

现在我们考虑 U_{CSP}。这里再次指出，要以 ϵ 精度执行该操作，只需使每个单量子比特旋转门的误差达到 ϵ/n。LOADF 操作的深度为 $O(n)$ ($\mathrm{D}_{\mathrm{approx}}=O(n+\log(1/\epsilon))$. 同时，在此情形下，SPF 和 FLAG 操作的深度均为 $O(n-m)$，因此总深度为 $O(n)$（总深度为 $O(n+\log(1/\epsilon))$。所需的总量子比特数为图 8 中出现的 $n+2N/M$，再加上执行 LOADF、SPF 和 FLAG 电路所需的额外 $O(N)$。空间与深度的乘积给出时空分配的直接上界 $O(n2^n)$ ($\mathrm{SA}_{\mathrm{approx}}=O(2^n\log(2^n/\epsilon))$)。不过，在这种情况下，实际时空分配优于该上界。尽管 LOADF 子程序作用于 $O(2^n)$ 个量子比特，深度为 $O(n)$ ($\mathrm{D}_{\mathrm{approx}}=O(n+\log(1/\epsilon))$)，其时空分配仍为 $O(2^n)$ ($\mathrm{SA}_{\mathrm{approx}}=O(2^n\log(n/\epsilon))$)。另一方面，FLAG 和 SPF 的时空分配为 $O((n-m)2^{n-m})$。总之，该过程的时空分配为 $O(2^n+(n-m)2^{n-m})$ ($\mathrm{SA}_{\mathrm{approx}}=O(2^n\log(n/\epsilon)+(n-m)2^{n-m})$).

将两部分相加，SP+CSP 协议的深度为 $O(n)$（$\mathrm{D}_{\mathrm{approx}}=O(n+\log(1/\epsilon))$，时空分配为 $O(m2^m+2^n+(n-m)2^{n-m})$（$\mathrm{SA}_{\mathrm{approx}}=O(m2^m+2^n\log(n/\epsilon)+(n-m)2^{n-m})$）。

若选择 m 使得

$$\lceil\log_2(n)\rceil\le m\le\lfloor n-\log_2(n)\rfloor,\tag{17}$$

我们看到，所述时空分配$O(2^n)$（$\mathrm{SA}_{\mathrm{approx}}=O(2^n\log(n/\epsilon))$）的结果成立。

[5]回顾一下，$O(m+\log(m/\epsilon))=O(m+\log(1/\epsilon))$，因为 $\log(m/\epsilon)=\log(m)+\log(1/\epsilon)$，并且 $O(m+\log(m))=O(m)$。

	D_{exact}	D_{approx}	# 量子比特	SA_{exact}	SA_{approx}
并行 Ry	$O(1)$	$O(\log(m/\epsilon))$	$O(2^m)$	$O(2^m)$	$O(2^m$ 对数$(m/\epsilon))$
SPF (U_{SP})	$O(m)$	$O(m)$	$O(2^m)$	$O(m2^m)$	$O(m2^m)$
并行 X门（U_{SP}）	$O(1)$	$O(1)$	$O(2^m)$	$O(2^m)$	$O(2^m)$
FLAG (U_{SP})	$O(m)$	$O(m)$	$O(2^m)$	$O(m2^m)$	$O(m2^m)$
并行受控 Ry 门	$O(1)$	$O(\log(m/\epsilon))$	$O(2^m)$	$O(2^m)$	$O(2^m$ 对数$(m/\epsilon))$
U_{SP}的总计	$O(m)$	$O(\log(2^m/\epsilon))$	$O(2^m)$	$O(m2^m)$	$O(2^m$ 对数$(2^m/\epsilon))$
并行 X（U_{CSP}）	$O(1)$	$O(1)$	$O(2^{n-m})$	$O(2^{n-m})$	$O(2^{n-m})$
LOADF	$O(n)$	$O(n+$对数$(1/\epsilon))$	$O(2^n)$	$O(2^n)$	$O(2^n$ 对数$(n/\epsilon))$
SPF（U_{CSP}）	$O(n-m)$	$O(n-m)$	$O(2^{n-m})$	$O((n-m)2^{n-m})$	$O((n-m)2^{n-m})$
FLAG (U_{CSP})	$O(n-m)$	$O(n-m)$	$O(2^{n-m})$	$O((n-m)2^{n-m})$	$O((n-m)2^{n-m})$
U_{CSP}的总计	$O(n)$	$O(n+\log(1/\epsilon))$	$O(2^n)$	$O(2^n+(n-m)2^{n-m})$	$O(2^n\log(n/\epsilon)+(n-m)2^{n-m})$
总计	$O(n)$	$O(n+$对数$(n/\epsilon))$	$O(2^n)$	$O(m2^m+2^n+(n-m)2^{n-m})$	$O(m2^m+2^n\log(n/\epsilon)+(n-m)2^{n-m})$
如果公式（17）满足	$\boldsymbol{O(n)}$	$\boldsymbol{O(n+$对数$(n/\epsilon))}$	$\boldsymbol{O(2^n)}$	$\boldsymbol{O(2^n)}$	$\boldsymbol{O(2^n\log(n/\epsilon))}$

Tab第3项：态制备协议每一步的单独复杂度总结。($M=2^m$, $N=2^n$)

回顾起来，我们可以明确指出采用 SP+CSP 态制备方法的理由。问题在于，简单地执行带有 $n=m$ 的 U_{SP} 协议，需要 2^n-1 个辅助量子比特来存储 2^n-1 角度，并且这些辅助量子比特中的每一个都必须在量子电路的整个 $O(n)$ 深度期间保持分配状态。SP+CSP 协议通过首先制备 m量子比特态 $|\phi\rangle$ 来规避这一问题，该过程仅需 2^m-1 个角度；对于 2^m 个基态中的每一个，则在其余量子比特中制备一个不同的 $n-m$ 量子比特态。后一个操作需要 $2^{n-m}-1$ 个角度（存储在缓冲辅助寄存器 B 中），因此我们避免了将 $O(2^n)$ 个角度态在整个 $O(n)$ 深度期间保持分配的需求。关键在于，我们能够在所有 2^n-2^m 个角度中加载正确的 $2^{n-m}-1$ 组角度，同时仅使用 $O(2^n)$ 的时空分配；这一过程由附录中介绍的 LOADF 实现完成。

在附录中，我们还讨论了 LOADF 子程序中的一些部分：其中，$|0\rangle$态中的干净辅助量子比特可以替换为处于任意态的脏辅助量子比特。总体而言，我们构造中的固定比例辅助量子比特可以是脏的，但仍有固定比例必须保持干净。未来的工作将研究能否使辅助量子比特中的一个渐近趋近于 1 的比例变为脏辅助量子比特。如果确实如此，那么区分时空分配中对应于干净辅助量子比特的部分与对应于脏辅助量子比特的部分，或许就具有意义。

4 制备多个相互独立的量子态副本

本节将展示，我们如何利用第 3节提出的新颖编码方法，更高效地制备任意量子态（相同或不同）的多个副本，采用 {U(2)，受控非门} 门集。具体而言，我们考虑尽可能快速地制备由 w 个独立的 N 维量子态组成的乘积态这一任务。如果我们有 $O(wN)$ 个辅助量子比特，就可以并行地对 w 个副本分别进行态制备，总深度为 $O(\log(N))$。然而，在实际应用中，N 很可能很大，而我们可能没有那么多辅助量子比特。假设我们只有 $O(N)$ 个辅助量子比特——足以在 $O(\log(N))$ 深度内制备一个量子态，却不足以制备 w 个量子态。简单地说，我们可以依次制备这 w 个量子态，深度为 $O(w\log(N))$。然而，制备单个量子态所需的时空分配为 $O(N)$，因此制备 w 个量子态所需的时空分配为 $O(wN)$。这表明，如果最优地重复利用 $O(N)$ 个辅助量子比特，我们或许能将总深度降至 $O(w)$，而非 $O(w\log(N))$。事实上，下面将说明，利用我们的 SP+CSP 协议，可以在 $O(w+\log(N))$ 深度内完成这一任务；当 $w\geq\Omega(\log(N))$ 时，每个副本的深度为常数 $O(1)$。

我们在第 3 节中描述的单副本任意态制备方法具有两个基本性质：

1. 所有辅助量子比特都与数据量子比特解除纠缠，并返回到 $|0\rangle$ 态。之前使用的辅助量子比特已准备好作为新的数据量子比特或新的辅助量子比特。

2. 活跃辅助量子比特占用的时空仅为 $O(N)$，尽管深度为 $O(\log(N))$，且有 $O(N)$个辅助量子比特。

因此，一旦辅助量子比特被送回 $|0\rangle$，即使前一个态制备尚未完成，它们也可以开始协助进行下一次态制备。我们可以将多个态制备电路依次连接并堆叠起来。

具体而言，我们可以制备多个彼此不纠缠的相同量子态副本

$$|\psi\rangle^{\otimes w} = \left[\prod_{s=0}^{w-1} U_s\right] |0^n\rangle^{\otimes w} = \left(\frac{1}{\|\mathbf{x}\|}\sum_{i=0}^{2^n-1} x_i |i\rangle\right)^{\otimes w} \tag{18}$$

这里的 U_s是相同的量子态制备酉算子，但作用于不同的 $|0^n\rangle$ 态。上面的公式没有包含用于辅助实现 U_s 酉算子的辅助量子比特：尽管U_s作用于不同的 $|0^n\rangle$ 数据寄存器，但它们可以共用许多相同的辅助量子比特。

我们也可以制备许多份不同的非纠缠量子态

$$\bigotimes_{d=0}^{w-1} |\psi_j\rangle = \left[\prod_{d=0}^{w-1} U_d\right] |0^n\rangle^{\otimes w} = \bigotimes_{d=0}^{w-1}\left(\frac{1}{\|\mathbf{x}^d\|}\sum_{i=0}^{2^n-1} x_i^d |i\rangle\right) \tag{19}$$

这里的 U_d是不同的量子态制备酉算子，作用于不同的 $|0^n\rangle$ 态，并制备由系数向量$\mathbf{x}^d$描述的不同 N维态。尽管每个 U_d 在数学上都不同，但精确的量子门调度相同。唯一的区别在于单量子比特旋转角度。

4.1 SP+CSP 多副本

为了制备式 (19)中的联合态，现在可以令 $U_d = U_{d(\mathrm{CSP})}U_{d(\mathrm{SP})}$。我们首先并行执行所有$U_{d(\mathrm{SP})}$ 量子电路，并令 $m = n - \lceil \log_2(n)\rceil$。也就是说，我们首先制备态

$$\left[\prod_{d=0}^{w-1} U_{d(\mathrm{SP})}\right] |0^n\rangle^{\otimes w} = \bigotimes_{d=0}^{w-1}(|\phi_d\rangle \otimes |0^{n-m}\rangle), \tag{20}$$

其中 $|\phi_d\rangle$ 由$\mathbf{x}^d$ 按照方程(5)定义。然后，我们联合执行所有 $U_{d(\mathrm{CSP})}$ 电路。

$$\prod_{d=0}^{w-1} U_{d(\mathrm{CSP})} \left(\bigotimes_{d=0}^{w-1}(|\phi_d\rangle \otimes |0^{n-m}\rangle)\right) = \bigotimes_{d=0}^{w-1} |\psi_d\rangle \tag{21}$$

我们没有足够的辅助量子比特来并行执行这些电路，因此会在它们之间留出一定的“间隔” k。也就是说，我们在第 0 层开始执行 $d = 0$ 协议，在第 k层开始执行 $d = 1$ 协议，在第 $2k$层开始执行 $d = 2$ 协议，依此类推。

我们在图9中将上述操作表示为量子电路。若选择 $m = n - \lceil \log_2(n)\rceil$，则式（20）中描述的第一个操作可在 $O(\log(N))$ 深度内完成，时空分配为 $O(N)$。由于每个 $U_{d(\mathrm{SP})}$ 都使用 $^{N}\log(N)$ 个量子比特，因此最多可

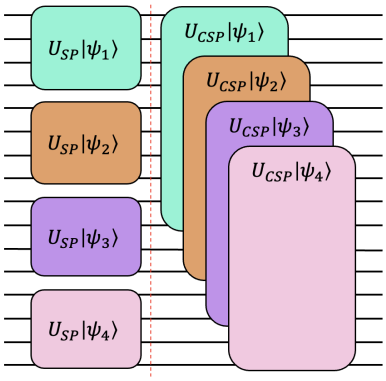


图9：将不同的 SP+CSP 电路堆叠在一起的示意图（$w = 4$）

以这种方式并行化 $O(\log(N))$ 个联合态，而无需引入额外的辅助量子比特。

现在，我们来仔细看看方程(21)中描述的第二个操作。CSP 电路中最消耗辅助量子比特的步骤是 LOADF 操作，尤其是附录图19中出现的双重控制旋转（CCR$_y$）操作层，它需要 $O(N)$ 个辅助量子比特，用于A 寄存器。请注意，在并行化的 CCR$_y$ 操作之后，借助CopySwap 操作，可以在 $O(N)$ 时空分配下快速释放这些辅助量子比特，如附录图19所示。因此，我们可以将后续U_{CSP} 部分的U_d操作错开若干层 k。

除了使用 $O(N)$ 个辅助量子比特的 A 寄存器外，在 U_{CSP} 操作中，我们还有 C、B 和 F 寄存器（如图 19 所示）。通过选择$m = n - \lceil \log_2(n)\rceil$，$C$寄存器（$m \cdot \frac{N}{M} = m\log(N)$ 个辅助量子比特）、B 寄存器（$M = {}^{N}\log(N)$ 个辅助量子比特）和 F寄存器（$\frac{N}{M} = n$ 个辅助量子比特）所需的辅助量子比特数量将可以忽略不计。因此，它们不会对总体时空分配成本产生太大影响。

我们在图10中给出了一个玩具模型的门级示意图。可以看到，量子电路的深度将为$2w + \log(N)$，而不是 $w\log(N)$。所使用的辅助量子比特数量上界为 $8N$。

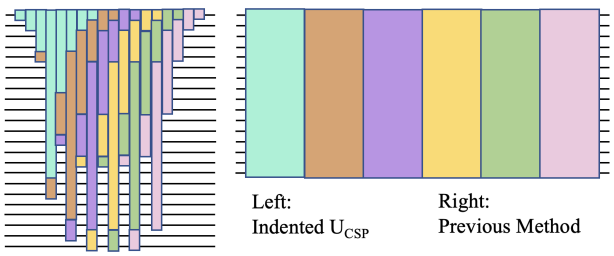


图10：将 U_{CSP} 量子电路（LOADF）的一部分以缩进 $k = 2$ 进行堆叠的示意图。

当 w 与 $\log(N)$ 的比值增大时——这在近期应用中往往如此，例如

用于量子机器学习的振幅编码（如第5.1节所示），我们实际上可以得到深度为 $O(w)$的态制备电路，与[18]中提出的协议所用的深度相同。

使用 {H、S、T、受控非门} 门集时，可以根据特定的 ϵ 值，制定利用这种早期无辅助比特结构的特定编译策略。从渐近角度来看，由于每次态制备占用的时空资源为 $O(N\log(\log(N)/\epsilon))$，因此使用 $O(N)$ 个辅助量子比特，就有可能以深度$O(w\log(\log(N)/\epsilon)+\log(N)+\log(1/\epsilon))$制备 w 个量子态。

5 应用

一般来说，量子测量具有概率性。因此，对于许多量子算法，我们通常需要在相同的量子电路配置下进行多次采样。也就是说，为了获得足够的**测量精度**，我们需要多次执行整个量子电路（态制备、算法量子电路和测量）。因此，对于许多需要多次采样/执行并进行任意量子态制备的量子算法，我们新颖的态制备方法最多可以将每次采样的平均时间缩短与量子比特数量成线性比例的倍数。

然而，还有一些更具体的例子表明，我们新提出的编码方法能够带来优势，而不仅仅是提高测量精度，包括但不限于：量子机器学习、哈密顿量模拟，以及使用类似 Harrow–Hassidim–Lloyd (HHL) 算法 [13] 的算法求解线性系统。

5.1 Quantum 机器学习——批处理

机器学习任务中一种提出的量子优势，是利用量子计算机指数级大的希尔伯特空间并行编码和处理数据，从而有望比经典机器学习方法更快。实现这种优势的一大瓶颈在于，将经典数据以指数级紧凑的形式编码到量子计算机中（振幅编码 [8]）代价高昂。在振幅编码中，我们希望将特征向量$\mathbf{x}=[x_0,x_1,...,x_{N-1}]$编码到量子态的振幅中，使得

$$|\psi\rangle=\frac{1}{\|\mathbf{x}\|}\sum_{i=0}^{N-1}x_i|i\rangle \tag{22}$$

请注意，$N=2^n$，其中 n是所需量子比特的数量。这表明我们可以用指数级紧凑的方式编码特征向量。

例如，如果我们有一幅 2×2 图像，其像素值数组为 $\mathbf{x}=[232, 31, 62, 137]$（每个像素的颜色值是介于 0 到 255 之间的整数），我们只需要 $\log_2 4=2$ 量子比特即可将这些值编码到振幅中：

$$|\psi\rangle=0.834\,|00\rangle+0.111\,|01\rangle+0.223\,|10\rangle+0.492\,|11\rangle \tag{23}$$

在典型的机器学习任务中，模型通过输入多份训练数据副本进行训练，并调整模型参数，以最小化根据这些数据计算出的损失函数。计算多个数据点的损失和后，可以更新参数（例如，批量梯度下降 [32]）。

在量子计算环境中，可以按如下方式执行一次量子机器学习迭代（图 11 也对此进行了说明）：

1. 同时将多个数据点编码到不同的量子态中；2. 同时处理所有量子态，例如分别使用不同的量子神经网络电路；3. 同时测量每个电路的输出（采用互不相交的测量或联合测量），并计算总损失。

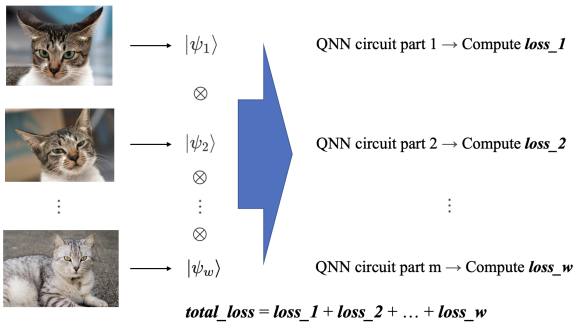


图11：同时执行方法示例

由于编码过程可能占用所需量子资源的相当一部分，因此必须考虑态制备的最佳策略。

使用我们的 SP+CSP 编码方法，并采用第 4 节所述的缩进方法，只需使用 $O(N)$ 个量子比特，即可实现 $O(w+\log(N))$ 的电路深度。原则上，给定 $O(N)$ 个辅助量子比特，就能以有效的常数深度编码许多数据点。在近期量子机器上运行时，这在理论上有效地达到了 Yuan 等人提出的协议所能实现的最佳性能 [18]，该协议使用 {U(2), CNOT} 门集。近期量子机器上的实际最佳策略将取决于机器属性，例如额外可用的原生门（如受控交换门）以及连接性限制。对于容错级量子机器学习算法（例如，使用 {H, S, T, CNOT} 门集），我们的 SP+CSP 协议在深度和时空分配方面都具有明显优势。

5.2 哈密顿量模拟——酉算子的线性组合

哈密顿量模拟是指在给定时长 t 和某个哈密顿量的描述 H 的情况下，合成时间演化酉算子 $U(t)$ 的任务。例如，哈密顿量模拟可以让人们测量随时间演化的量子态的性质

$$|\psi(t)\rangle = U(t)\,|\psi(0)\rangle \tag{24}$$

其中 $|\psi(0)\rangle$ 是初始态，并且

$$U(t) = e^{-itH}\,. \tag{25}$$

进行哈密顿量模拟的一种方法是使用酉算子的线性组合（LCU） [9, 33]。该方法的思路是用截断至 K 阶的泰勒展开来近似哈密顿量演化算子，然后将其展开为泡利矩阵（酉算子）的线性组合 并将其展开为泡利矩阵（酉算子）的线性组合

$$U(t) \approx \sum_{k=0}^{K} \frac{(-itH)^k}{k!} \tag{26}$$

$$= \sum_{j=0}^{\Gamma-1} \beta_j V_j \tag{27}$$

其中 V_j 是泡利矩阵，线性组合中泡利矩阵的数量为 $\Gamma = \sum_{k=0}^{K} L^k$，其中 L 是哈密顿量 H 中泡利项的数量。令 $a = \lceil \log_2(\Gamma) \rceil$。

接着，通过构造 SELECT算子和态制备算子 B 来实现酉算子的线性组合，其定义如下：

$$\mathrm{SELECT} = \sum_{j=0}^{\Gamma-1} |j\rangle\langle j| \otimes V_j \tag{28}$$

$$B\,|0^a\rangle = \frac{\sum_{j=0}^{\Gamma-1} \sqrt{\beta_j}\,|j\rangle}{\sqrt{\sum_{j=0}^{\Gamma-1} \beta_j}}\,. \tag{29}$$

将这些组合起来，得到

$$W = (B^\dagger \otimes I)\mathrm{SELECT}(B \otimes I)\,, \tag{30}$$

我们看到

$$(\langle 0^a| \otimes I)\,W\,(|0^a\rangle \otimes I) = \frac{\sum_{j=0}^{\Gamma-1} \beta_j V_j}{\sum_{j=0}^{\Gamma-1} \beta_j} \approx \frac{U(t)}{\sum_{j=0}^{\Gamma-1} \beta_j} \tag{31}$$

上述方程等价于这样一个陈述： W 是 $U(t)$[34, 35]，的近似“块编码”，块编码因子为 $\sum_{j=0}^{\Gamma-1} \beta_j$。因此，可以通过应用 W 并后选择辅助态 $|0^a\rangle$来实现$U(t)$，其成功概率为$(\sum_{j=0}^{\Gamma-1} \beta_j)^{-2}$。通过将演化时间 t 划分为 r 个长度为 t/r 的片段，选择特定的 r，并应用无意识振幅放大 [9, 10]，即可将成功概率提升至 1。

无论如何，态制备算子 B是该算法的关键子程序。使用我们的态制备方法，可以用较浅的深度实现它。在典型的量子化学模拟设置中，所考虑的二次量子化哈密顿量包含二电子积分，有时还包含三电子积分，这会使 Γ的标度为 $O(n^4)$ [36] ，有时为 $O(n^6)$ [37, 38]（其中 n 表示轨道数）。线性组合酉算子中的项数甚至会更多（取决于截断参数）；不过，我们的低深度态制备方法的深度标度为$O(\log(n))$。

近期研究 [39–41]表明，**对**多个量子态副本进行联合测量（例如， $|\psi(t)\rangle$ 见方程(24)）优于对末态分别进行测量。这样可以获得更好的样本复杂度标度，最终通过减少达到特定测量阈值所需的量子电路总执行次数，降低总体电路复杂度。

5.3 Quantum 线性方程组求解 — 重复直至成功

经典线性系统问题定义如下：给定一个 $N \times N$ 可逆矩阵 A和一个$N \times 1$ 向量 b，求解向量 x，使得Ax = b。量子线性系统问题则是执行以下相关任务：制备一个量子态|x⟩ = $\frac{1}{\|\mathbf{x}\|}\sum_i x_i\,|i\rangle$，其振幅与解向量 x 成正比。虽然量子态 |x⟩无法让我们访问向量 x 的所有 N 元素，但可以利用 |x⟩的多个副本来估计可观测量 M 的期望值 ⟨x| M |x⟩。首个用于求解量子线性系统问题的量子算法由 Harrow、Hassidim 和 Lloyd（HHL）提出 [13]。他们的方案首先制备量子态 |b⟩ = $\|\mathbf{b}\|^{-1}\sum b_i\,|i\rangle$，其中 b_i 是向量 b 的各个元素。然后，若 A 是稀疏矩阵，且假设可以在 polylog(N) 时间内相干查询 A 的元素，则可以在 poly(κ, log(N)) 时间内以高精度制备量子态 |x⟩，其中 κ 是 A的条件数，也就是最大奇异值与最小奇异值之比。当 $\kappa = O(1)$时，与处理长度为N的向量的经典方法（例如高斯消元或共轭梯度下降 [42]）相比，这种方法的速度快了指数级。然而，在某些应用中，例如求解微分方程 [43]，从量子态 |x⟩中读取有用信息往往会抵消这种指数级加速（但仍可能带来多项式级加速）。即便如此，由于相关数据可以通过量子方式进行指数级紧凑的表示，空间复杂度仍可能具有指数级优势

量子态。与此相关的是，在通信复杂度设定下，利用这种紧凑性可以在线性系统问题上获得无条件的指数级量子优势 [44]。

需要注意的是，即使 A 是稀疏矩阵，向量$\mathbf{b}$ 也往往是稠密的。例如，通过映射到线性方程组来求解微分方程时，就会出现这种情况： [43], $\mathbf{b}$ 编码了问题的任意边界条件。因此，作为量子算法第一步来制备量子态 $|\mathbf{b}\rangle$ ，正是对我们态制备协议的一种应用。

我们的低深度态制备协议非常适合用于量子线性系统问题的算法，原因有几个。当 $\mathbf{b}$ 是稠密向量时，除非采用低深度的态制备方法，否则制备 $\mathbf{b}$ 将占据算法运行时间的主要部分。此外，从解向量 $|\mathbf{x}$中提取信息可能需要多次重复运行算法，因而需要制备许多份 $|\mathbf{b}$；如第 4 节所述，我们的方法具有最优的时空分配，因此可以快速制备多份副本，同时高效使用辅助量子比特。此外，根据所采用的量子方法，算法本身可能需要制备许多份 $|\mathbf{b}$，才能生成一份 $|\mathbf{x}$。例如，在现代方法中，例如 [34], ，可以通过酉算子“块编码” U_A访问矩阵 A。利用信号处理技术，可以创建 A^{-1}的块编码 $U_{A^{-1}}$ 。酉算子 $U_{A^{-1}}$ 使用 $O(\kappa)$次对酉算子 U_A的调用，并执行如下操作（误差很小）

$$U_{A^{-1}}|\mathbf{b}\rangle|0^{\ell}\rangle \approx c|\mathbf{x}\rangle|0^{\ell}\rangle + |\perp\rangle \tag{32}$$

其中， $|\perp$ 位于投影算子 $I\otimes|0^{\ell}\,0^{\ell}|$ 的零空间中，并且 $|c| = O(1/\kappa)$。随后，对辅助寄存器进行测量，以概率 $|c|^2 = O(1/\kappa^2)$ 得到所需的量子态 $|\mathbf{x}$。因此，要制备$|\mathbf{x}$，只需创建 $O(\kappa^2)$ 个 $|\mathbf{b}$ 的副本，并针对每个副本向 U_A 发出 $\tilde{O}(\kappa)$ 次查询，总查询次数为 $\tilde{O}(\kappa^3)$。原则上，这些副本可以并行处理；如上所述，我们的方法仅使用 $O(N)$ 个辅助量子比特，即可提供一种深度最小的方式来制备 $|\mathbf{b}^{\otimes(O(\kappa^2))}$。使用振幅放大，可将对 U_A 的查询次数减少到 $\tilde{O}(\kappa^2)$，但这需要围绕量子态 $|\mathbf{b}$ 进行 $O(\kappa)$ 次串行反射，而我们的协议可以最优地执行这一操作（见附录 B）。最后，使用可变时间振幅放大[14, 34] 或借鉴绝热量子计算[45, 46], 中的思想，可以将 U_A 的查询复杂度进一步降低到 $O(\kappa)$。这一过程还需要串行应用我们的态制备协议 $O(\kappa)$ 次。

5.4 块编码

如上文所述，块编码作为一种框架，可用于使量子算法——例如哈密顿量模拟或量子线性方程组求解器——访问某些底层数据[34, 35, 47]。量子态制备也是构造块编码的一个有用组成部分。

矩阵 A 的块编码定义为酉算子 U_A，其中 A 出现在左上角的块中，并由某个常数 α 缩放，使得 $\|A/\alpha\| \le 1$:

$$U_A = \begin{bmatrix} A/\alpha & \cdot \\ \cdot & \cdot \end{bmatrix}. \tag{33}$$

例如，给定一个表示某个物理系统的哈密顿量 H ，我们可以寻求它的块编码U_H ，随后便可用它构建生成该系统基态或热态的算法。它还可用于通过量子比特化 [12, 35], 进行哈密顿量模拟；与前文讨论的线性组合酉算子方法相比，这种技术的性能可能更优。

块编码 U_A 的构造方式取决于 A 的内容以及算法的具体背景。许多最常见的方法都明确需要量子态制备或受控量子信号处理（参见[35]第 4.2 节）。

例如，当 A 是格拉姆矩阵——也就是说，其元素为 $A_{ij} = \langle\psi_i|\phi_j\rangle$ ，其中对应某些集合$\{|\psi_i\rangle\}_i$、$\{|\phi_j\rangle\}_j$——我们可以构造A 的块编码 $U_A = U_L^{\dagger}U_R$，其中 U_L 和 U_R 是酉算子，可用于制备任意量子态 $|\psi_i\rangle$ 和 $|\phi_j\rangle$ [35, （引理47]) 。类似的方法也可用于构造任意矩阵 [20, 34] 的块编码。

二者 U_L 和 U_R 均可通过我们受控态制备协议构造，其优化后的深度为 $O(\log(N))$，且需要 $O(N)$个辅助量子比特。

6 代码可用性

我们给出了两个量子门级实现示例 https://github.com/guikaiwen/QSP_Paper_Artifact, 包括:

- a $2+2$ case $(m=2, n=4)$ case without COPY
- a $1+2$ case $(m=1, n=3)$ case with COPY

致谢

感谢 Connor Hann、Yunong Shi、Erhan Bas、Pei Zeng、Ming Yuan、Siteng Kang、Kyle Brubaker、Martin Schuetz、Fei Chen、Liang Jiang、Eric Kessler、Senrui Chen、Sisi Zhou 和 Gideon Lee 提供的宝贵讨论。本工作部分由美国国家科学基金会计算领域远征项目 EPiQC 资助，资助编号为 CCF-1730449。FTC 是 Infleqtion 量子软件首席科学家，同时担任 Quantum Circuits, Inc. 的顾问。

参考文献

[1] Jacob Biamonte、Peter Wittek、Nicola Pancotti、Patrick Rebentrost、Nathan Wiebe 和 Seth Lloyd。“量子机器学习”。《自然》549，195–202（2017年）。[2]Seth Lloyd、Masoud Mohseni 和 Patrick Rebentrost。“量子主成分分析”。《自然·物理》10，631–633（2014年）。[3] Iordanis Kerenidis 和 Anupam Prakash。“量子推荐系统”。载于第八届理论计算机科学创新会议（ITCS 2017）。LIPIcs 第67卷，页码49:1–49:21。（2017年）。[4] Patrick Rebentrost、Adrian Steffens、Iman Marvian 和 Seth Lloyd。“非稀疏低秩矩阵的量子奇异值分解”。《物理评论A》97，012327（2018年）。[5]Iordanis Kerenidis、Jonas Landman、Alessandro Luongo 和 Anupam Prakash。“q-means：一种用于无监督机器学习的量子算法”。神经信息处理系统进展（2019年）。[6] Iordanis Kerenidis 和 Jonas Landman。“量子谱聚类”。《物理评论A》103，042415（2021年）。[7] Patrick Rebentrost、Masoud Mohseni 和 Seth Lloyd。“用于大数据分类的量子支持向量机”。《物理评论快报》113，130503（2014年）。[8]Maria Schuld 和 Francesco Petruccione。“量子计算机机器学习”。施普林格。（2021年）。[9] Dominic W Berry、Andrew M Childs、Richard Cleve、Robin Kothari 和 Rolando D Somma。“利用截断泰勒级数模拟哈密顿动力学”。《物理评论快报》114，090502（2015年）。[10] Dominic W Berry、Andrew M Childs 和 Robin Kothari。“对所有参数均具有近乎最优依赖关系的哈密顿量模拟”。载于2015年 IEEE 第56届计算机科学基础年度研讨会。页码792–809。IEEE（2015年）。[11]Guang Hao Low 和 Isaac L Chuang。“通过量子信号处理实现最优哈密顿量模拟”。《物理评论快报》118，010501（2017年）。[12] Guang Hao Low 和 Isaac L Chuang。“通过量子比特化进行哈密顿量模拟”。Quantum 3，163（2019年）。[13] Aram W Harrow、Avinatan Hassidim 和 Seth Lloyd。“用于线性方程组的量子算法”。《物理评论快报》103，150502（2009年）。[14] Andris Ambainis。“可变时间振幅放大及用于线性代数问题的量子算法”。载于 STACS’12（第29届研讨会计算机科学理论方面研讨会 - 文）。第14卷，636–647页。LIPIcs（2012年）。[15] Leonard Wossnig、Zhikuan Zhao 和 Anupam Prakash。《稠密矩阵的量子线性系统算法》。《物理评论快报》120，050502（2018年）。[16] Guang Hao Low、Vadym Kliuchnikov 和 Luke Schaeffer。《在态制备和幺正合成中以脏量子比特换取T门》. arXiv.1812.00954（2018年）。[17] Xiaoming Sun、Guojing Tian、Shuai Yang、Pei Yuan 和 Shengyu Zhang。《量子态制备和一般幺正合成的渐近最优电路深度》。《IEEE计算机辅助设计集成电路与系统汇刊》（2023年）。[18] Pei Yuan 和 Shengyu Zhang。《利用任意数量辅助量子比特的量子电路实现最优（受控）量子态制备与改进的幺正合成》。《Quantum》7，956（2023年）。[19] Xiao-Ming Zhang、Tongyang Li 和 Xiao Yuan。《最优电路深度的量子态制备：实现与应用》。《物理评论快报》129，230504（2022年）。[20] B David Clader、Alexander M Dalzell、Nikitas Stamatopoulos、Grant Salton、Mario Berta 和 William J Zeng。《对经典数据矩阵进行块编码所需的量子资源》。《IEEE量子工程汇刊》（2023年）。[21]Gregory Rosenthal。《通过Grover搜索获得量子幺正算子的查询与深度上界》。arXiv.2111.07992（2021年）。[22] Neil J. Ross 和 Peter Selinger。《z旋转的最优无辅助量子比特 Clifford+T近似》。《量子信息与计算》（2016年）。[23] Ryan Babbush、Craig Gidney、Dominic W Berry、Nathan Wiebe、Jarrod McClean、Alexandru Paler、Austin Fowler 和 Hartmut Neven。《用线性T复杂度在量子电路中编码电子光谱》。《物理评论X》8，041015（2018年）。[24] Israel F Araujo、Daniel K Park、Francesco Petruccione 和 Adenilton J da Silva。《量子态制备的分治算法》。《科学报告》11，1–12（2021年）。[25] Vivek V. Shende 和 Igor L. Markov。《关于Toffoli门的受控非门成本》。《量子信息与计算》（2009年）。[26] John A Smolin 和 David P DiVincenzo。《五个双比特量子门足以实现量子 Fredkin门》。《物理评论 A》53，2855（1996年）。[27] Edward Walker。《CPU小时的真实成本》。《计算机》42，35–41（2009年）。[28]Yongshan Ding、Xin-Chuan Wu、Adam Holmes、Ash Wiseth、Diana Franklin、Margaret Martonosi 和 Frederic T Chong。

《Square：通过成本效益高的反计算实现模块化量子程序的战略性量子辅助比特复用》。载于2020年ACM/IEEE第47届年度国际计算机体系结构研讨会（ISCA）。第570–583页。IEEE（2020年）。[29] Martin Plesch 和 ˇCaslav Brukner。《利用通用量子门分解进行量子态制备》。《物理评论 A》83，032302（2011年）。[30] Xiao-Ming Zhang 和 Xiao Yuan。《用于编码经典数据的量子访问模型的电路复杂度》. arXiv.2311.11365（2023年）。[31] Michael A Nielsen 和 Isaac Chuang。《量子计算与量子信息》。美国物理教师协会。（2002年）。[32]Sebastian Ruder。《梯度下降优化算法概述》。arXiv.1609.04747（2016年）。[33]Andrew M Childs、Dmitri Maslov、Yunseong Nam、Neil J Ross 和 Yuan Su。《迈向首次实现具有量子加速的量子模拟》。《美国国家科学院院刊》115，9456–9461（2018年）。[34]Shantanav Chakraborty、Andr´as Gily´en 和 Stacey Jeffery。《块编码矩阵幂的威力：通过更快速的哈密顿量模拟改进回归技术》。载于第46届国际自动机、语言与程序设计研讨会（ICALP）论文集。第33:1–33:14页。（2019年）。[35] Andr´as Gily´en、Yuan Su、Guang Hao Low 和 Nathan Wiebe。《量子奇异值变换及其拓展：量子矩阵运算的指数级改进》。载于第51届ACM计算理论研讨会（STOC）论文集。第193–204页。（2019年）。[36]Trygve Helgaker、Poul Jorgensen 和 Jeppe Olsen。《分子电子结构理论》。约翰·威立父子出版公司。（2013年）。[37] Mario Motta、Tanvi P Gujarati、Julia E Rice、Ashutosh Kumar、Conner Masteran、Joseph A Latone、Eunseok Lee、Edward F Valeev 和 Tyler Y Takeshita。《利用相互关联哈密顿量进行电子结构量子模拟：以更小的量子计算机资源占用获得更高精度》。《物理化学化学物理》22，24270–24281（2020年）。[38]Sam McArdle 和 David P Tew。《利用相互关联变换方法提高量子计算化学的精度》。arXiv.2006.11181（2020年）。[39] Sebastien Bubeck、Sitan Chen 和 Jerry Li。《最优量子性质测试离不开纠缠》。载于2020年IEEE第61届计算机科学基础年度研讨会（FOCS）。第692–703页。IEEE（2020年）。[40] Sitan Chen、Jordan Cotler、Hsin-Yuan Huang 和 Jerry Li。“使用量子存储器与不使用量子存储器进行学习之间的指数级分离”。载于《2021年IEEE 第62届年度基础计算机科学研讨会（FOCS）》。第574–585页。IEEE（2022年）。[41]Hsin-Yuan Huang、Michael Broughton、Jordan Cotler、Sitan Chen、Jerry Li、Masoud Mohseni、Hartmut Neven、Ryan Babbush、Richard Kueng、John Preskill 等。“从实验中学习的量子优势”。Science 376，1182–1186（2022年）。[42] Jonathan Richard Shewchuk 等。“共轭梯度法简介：告别令人痛苦的烦恼”。1994年技术报告（1994年）。[43] Ashley Montanaro 和 Sam Pallister。“量子算法与有限元法”。《物理评论A》93，032324（2016年）。[44] Ashley Montanaro 和 Changpeng Shao。“线性回归的量子通信复杂度”。ACM 计算理论汇刊（2023年）。[45]Yi˘git Suba¸sı、Rolando D. Somma 和 Davide Orsucci。“受绝热量子计算启发的线性方程组量子算法”。《物理评论快报》122，060504（2019年）。[46]Pedro CS Costa、Dong An、Yuval R Sanders、Yuan Su、Ryan Babbush 和 Dominic W Berry。“通过离散绝热定理实现最优标度的量子线性系统求解器”。PRX Quantum 3，040303（2022年）。[47]John M. Martyn、Zane M. Rossi、Andrew K. Tan 和 Isaac L. Chuang。“量子算法的大统一”。PRX Quantum 2，040203（2021年）。[48] Craig Gidney。“Quirk：一款拖放式量子电路模拟器”。https://algassert.com/quirk （2016年）。[49]Alexander M Dalzell、B David Clader、Grant Salton、Mario Berta、Cedric Yen-Yu Lin、David A Bader、Nikitas Stamatopoulos、Martin JA Schuetz、Fernando GSL Brand˜ao、Helmut G Katzgraber 等。“量子内点法与投资组合优化的端到端资源分析”。PRX Quantum 4，040325（2023年）。

A 电路实现

A.1 量子比特数据复制

我们的 SPF 和 FLAG 电路实现需要额外的辅助量子比特，用作相干数据复制子程序的一部分。也就是说，我们执行等距映射，例如在参考文献 [17, 18] 中也使用了这种映射

$$\alpha|0\rangle + \beta|1\rangle \mapsto \alpha|0^c\rangle + \beta|1^c\rangle \tag{34}$$

仅使用 CNOT 门，并且深度为 $\lceil \log_2(c) \rceil$。我们将需要这一子程序，因为我们经常希望执行许多目标互不重叠、但由同一寄存器控制的 CSWAP 门。通过复制控制寄存器并并行应用这些 CSWAP 门，我们可以以更小的深度完成此操作。Clader 等人 [20] 通过假设能够在单个时间步内执行具有任意数量目标的 FANOUT-CNOT 来避免这一问题；后者是一种 Clifford 门。

复制协议由 $\lceil \log_2(c) \rceil$ 个连续的深度为 1 的操作组成，标记为 $\bigoplus_0$，…，$\bigoplus_{\lceil \log_2(c) \rceil - 1}$。我们将这些称为复制层。特别地，$\bigoplus_j$ 由单个层构成，其中包含 2^j 个并行 CNOT 门，这些 CNOT 门的目标位是新的辅助量子比特。如图 12 所示，并在算法 3 中进行了描述。值得注意的是，尽管量子比特总数为 c，深度为 $\lceil \log_2(c) \rceil$，但大多数辅助量子比特直到协议接近结束时才需要分配，因此时空分配为 $\Theta(c)$，而非 $O(c\log(c))$。

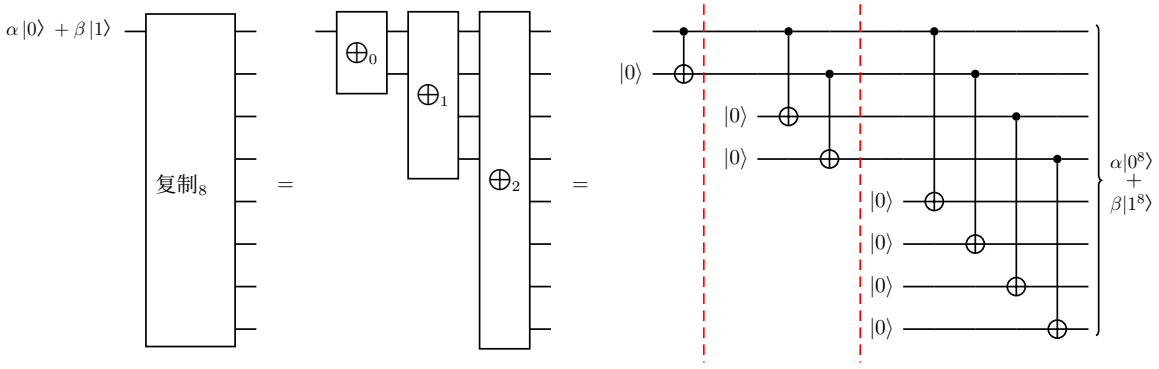


图12：用于实现 $c = 8$ 个量子比特状态复制子程序的示例量子电路。通过一系列 t 层受控非门（记作 $\bigoplus_0$，……，$\bigoplus_{t-1}$），将任意单量子比特态 $\alpha|0\rangle + \beta|1\rangle$ 复制到 $c = 2^t$ 个量子比特上（即映射为 $\alpha|0^c\rangle + \beta|1^c\rangle$）。$c-1$ 个量子比特是新引入的辅助量子比特，初始状态为 $|0\rangle$。深度为 $\log_2(c)$，时空分配仅为c；如右侧示意图所示，这是通过等到最后需要使用辅助量子比特时才分配它们实现的。

算法 3 STATE-COPY 子程序

过程复制$_c(R)$
    ▷ R 是一个大小为 c 的寄存器，其中为简便起见，假设 c 是 2 的幂
    对于 i，取值范围为 range($\log_2(c)$) **执行**　▷ i 的每个值占用 $O(1)$ 深度
        $\bigoplus_i(R)$
    结束循环
结束过程
过程 $\bigoplus_t(R)$
    对 j 遍历 range(2^t) **执行**　▷ 并行执行 j 的所有取值
        受控非门($R_{jc2^{-t}}$, $R_{jc2^{-t}+c2^{-t-1}}$)
    结束循环
过程结束

▷ 总深度$_{\text{exact}}$：$\log_2(c)$▷ 总深度$_{\text{approx}}$：$\log_2(c)$▷ 总时空复杂度$_{\text{exact}}$：$2c-2$▷ 总时空复杂度$_{\text{approx}}$：$2c-2$

A.2 并行 CSWAP

借助 STATE-COPY 子程序，我们现在可以有效地使用单量子比特门和双量子比特 CNOT 门并行执行受控交换门操作，而无需使用参考文献中要求的 FANOUT-CNOT 门。 [20] 该门用于在共享相同控制位时并行执行受控交换操作。

我们将一层 2^t 并行受控交换门记为CS_t，如图图13 和算法4所示。

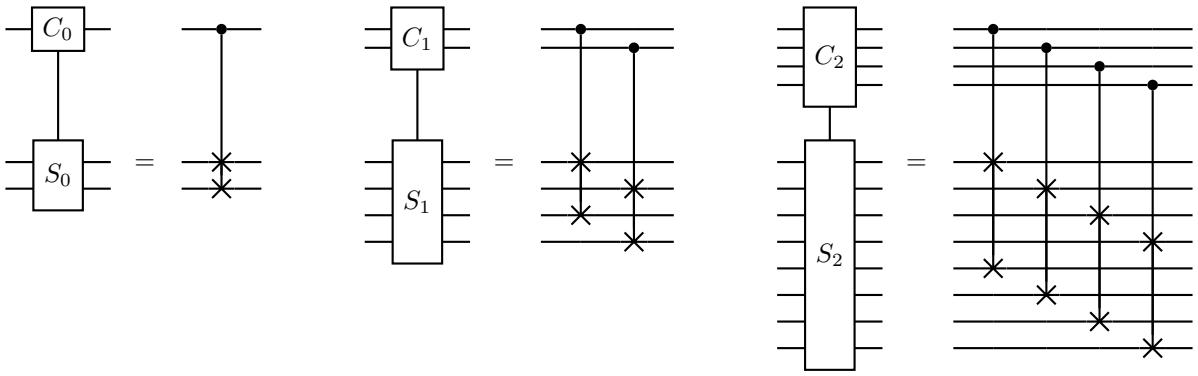


图13： CS_t 的实现，其中 $t = 0$，1，2 可通过一层并行 CSWAP 门实现 s.

算法 4 并行 CSWAP

1: **过程**$\mathrm{CS}_t(R,\ S)$▷ t：并行受控交换的对数数量，R：至少含有 2^t 个量子比特的控制位数据寄存器，S：至少含有 2^{t+1} 个量子比特的目标位角度寄存器（注意，此处的下标标记每个单独寄存器的量子比特索引）2: **对于** i ，遍历 range(2^t) **执行**

▷ 所有 i 的值并行执行

图3: 受控交换门($R_i,\ S_i,\ S_{i+2^t}$)

图4: **结束循环**图5: **结束过程**

A.3 复制层应用示例

在图14中，我们展示了如何在恒定的辅助量子比特数量和总电路深度开销下，有效并行化多个控制位相同、目标位不同的受控交换门。相同的思路也适用于 Toffoli 门。

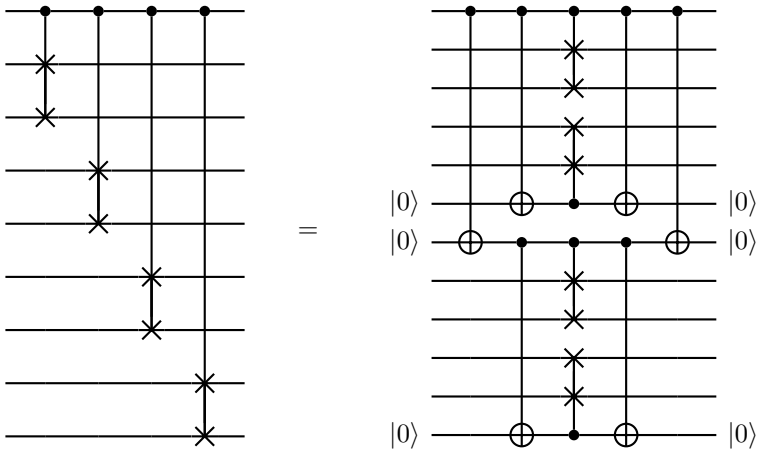


图14：使用和不使用复制层时的量子电路等价性：请注意，我们仅在 $|0\rangle$ 态中增加了 $O(\log(N))$ 个受控非门层和 $O(N)$ 个辅助量子比特，其规模与量子电路总深度和辅助量子比特总数相同。

A.4 SPF

现在我们介绍 SPF 电路的实现。该电路作用于 $m + M - 1$ 个量子比特，而我们的实现使用额外的 $M/2 - m$ 个辅助量子比特，它们初始和最终都处于 $|0\rangle$。该电路与参考文献 [20]，中描述的电路类似，但在我们的实现中，所有量子门都是双量子比特门。其思路是依次对 m 个数据量子比特寄存器施加 $R_y(\theta_{s,p})$ 旋转，但第 s 次旋转所用的寄存器对 (s, p) 取决于前 $s - 1$ 个寄存器的取值。我们假设每个 $|\theta_{s,p}\rangle$ 量子态都已制备好。对于每个 $s = 0, \ldots, m - 1$，通过将正确的 $|\theta_{s,p}\rangle$ 量子态与第 s 个数据量子比特交换（“注入”），来施加这些旋转。对于每个 s，恰好会注入一个 $\theta_{s,p}$。由于 (0, 0) 是唯一满足 $s = 0$ 的 (s, p)，第一步是将 $|\theta_{0,0}\rangle$ 与第一个数据量子比特交换。第二步是在第一个数据寄存器处于 $|1\rangle$ 的条件下，交换分别存放 $|\theta_{1,0}\rangle$ 和 $|\theta_{1,1}\rangle$ 的寄存器，然后将原先存放 $|\theta_{1,0}\rangle$ 的寄存器注入第二个数据寄存器（使用一个 SWAP门）。一般而言，第 s 次旋转的实现方法是，先将正确的 $|\theta_{s,p}\rangle$ 交换到一个大小为 2^s 的寄存器顶部，该寄存器最初存放着 $|\theta_{s,p}\rangle$ 量子态；然后将顶部寄存器与第 s 个数据量子比特交换。直接实现似乎至少需要 $O(m^2)$ 深度，因为共有 m 次旋转，而每次都要确定注入哪个量子比特，这需要 $O(m)$ 轮受控交换。参考文献 [20] 指出，若能使用具有 $O(M)$ 个目标且可在单位时间内执行的 FANOUT-CNOT，则可将深度降至 $O(m)$。要将这样的 FANOUT-CNOT 门分解为双量子比特门，通常至少需要 $O(m)$ 深度；因此，在 $\{U(2), \text{CNOT}\}$ 门集中，Clader 等人的 [20] SPF 实现深度为 $O(m^2)$，而非 $O(m)$。

我们给出一种电路编译方法，通过使用少量额外的辅助量子比特，将深度降低至 $O(m)$。具体做法是穿插使用受控交换门和复制层。特别是，对于每个 $u = s + 1,\ s + 2,\ \ldots\ldots,\ m - 1$，数据量子比特 s 需要用来控制角度寄存器上的 2^{u-1-s} 受控交换门；该寄存器初始处于量子态 $|\Theta_u\rangle$。因此，对其应用 $u - 1 - s$ 个复制层后，我们便可以自由地并行执行受控交换门；我们将并行执行 2^t 个受控交换门的操作记作 CS_t，如图 13所示。在以数据量子比特 s为控制位执行受控交换门时，我们可以并行复制其他数据量子比特。总而言之，我们只需$O(m)$ 深度即可完成该操作。

我们从多个方面描述 SPF 电路。在图 15中，我们给出了 $m = 4$的 SPF 完整示例，展示了如何使用辅助量子比特来存放数据量子比特的副本，以便控制交换操作。然而，当 m的值较小时，很难充分看出其中的规律。在图 16中，我们展示了一个较大 $m = 7$ 示例的 SPF 电路前半部分，并标注了各个寄存器。通过该图可以轻松验证，其深度为 $O(m)$。最后，在算法 5中，我们给出了 SPF 实现的伪代码，其标注与图 16中的标注一致。为了理解这些量子电路背后的思路，下面我们说明必须遵守的几条规则（以及相应的原因）：

1. 若 $j < i, \forall q$，则 $\bigoplus_j(D_q)$ 必须在 $\bigoplus_i(D_q)$ 之后执行：
   - 根据构造可知这一点成立，如图12所示。
2. 若 $\forall j, \forall q$，则 $\bigoplus_j(D_q)$ 必须在 $\text{SWAP}(D_{q,\,0},\ A_{q,\,0})$ 之后执行：
   - 这是因为，必须先将角度态注入数据量子比特 q，之后才能开始复制数据量子比特 q。
3. $\text{CS}_k(D_q,\ \cdot)$ 必须在 $\bigoplus_{k-1}(D_q)$ 之后执行，$\forall q$：
   - 这样可以确保有足够的量子比特供受控交换序列用作控制位，从而保证最大程度的并行化。

图4. $\text{CS}_j(A_q)$ 需要按照从 $q - 1$ 到 0 的顺序遍历所有 j 值，然后再执行 $\text{SWAP}(D_{q,\,0},\ A_{q,\,0})$，$\forall q$：

- 这样可以确保，在根据先前量子比特的振幅将当前量子比特泵入数据寄存器之前，所有值都已交换到当前量子比特对应的正确位置。

由于存在条件 #2 和 #4，我们不能像下一小节（算法 6）中介绍的 FLAG 电路编译那样，从每个 D_q 仅执行一次 STATE-COPY 子程序。这样做会导致 $O(m^2)$ 深度。因此，我们必须将受控交换序列与复制层交错进行。

我们需要的用于数据量子比特 j 的复制层总数为 $m-2-j$（针对 $j = 0$、……、$m-3$）。数据量子比特 $m-2$ 和 $m-1$ 无需复制。因此，所需辅助量子比特总数为 $\sum_{j=0}^{m-3}(2^{m-2-j}-1) = 2^{m-1}-m$。

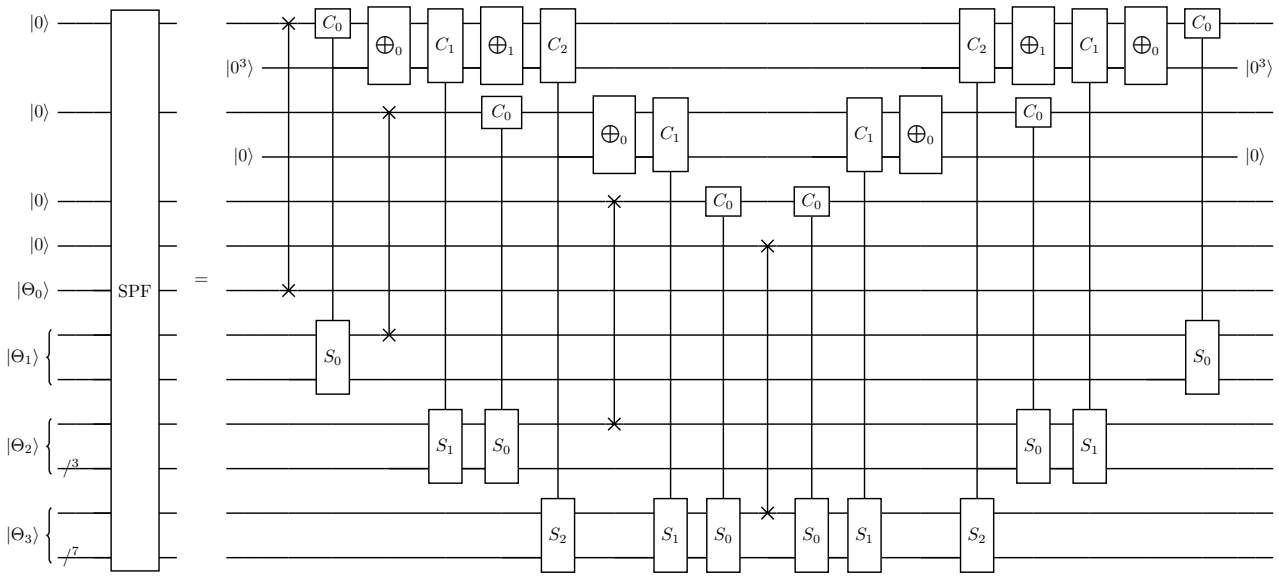


图15：用于制备含有 $m = 4$ 个量子比特的量子态的 SPF 电路，这些量子比特与 $2^q - 1$ 个量子比特（垃圾）保持纠缠。该量子电路的深度为 $O(m)$，并额外使用 $O(2^m)$ 个辅助量子比特，它们以 $|0\rangle$ 开始并结束。因此，总时空分配的上界为 $O(m2^m)$。SPF 电路的后半部分与前半部分的逆过程非常相似，但不包含 SWAP 门。

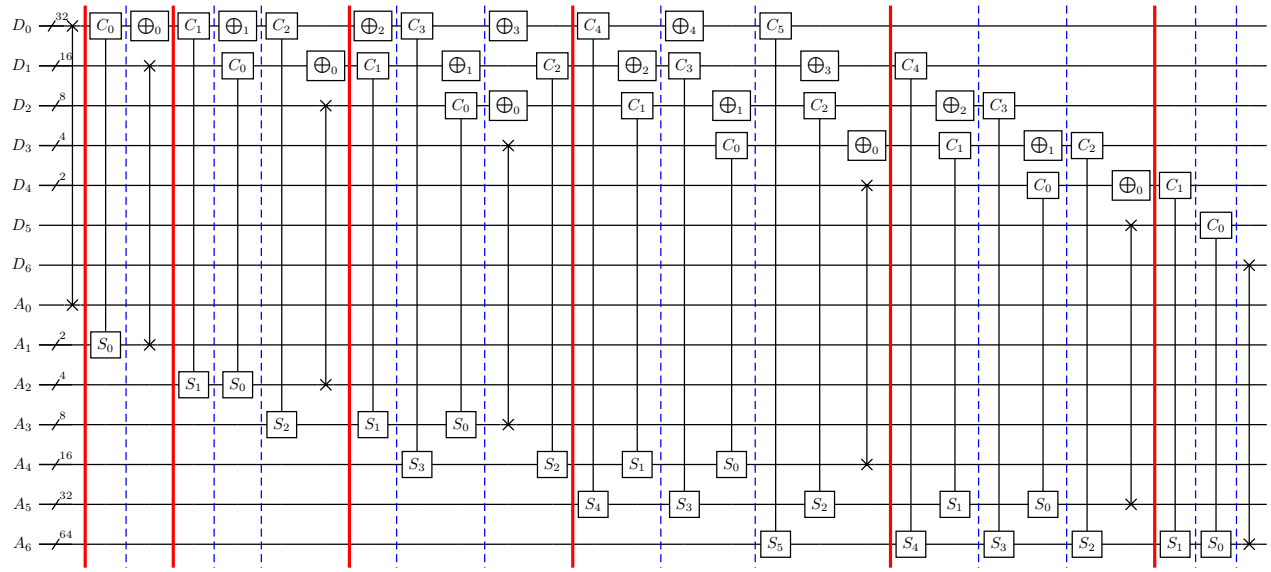


图16：SPF 的第一部分，适用于 $m = 7$。量子电路由 7 个数据寄存器（D_0 - D_6）和 7 个辅助寄存器（A_0 - A_6）组成。量子电路中每两条红线之间的空间表示算法 i 中 5 个值的一次迭代。每个红线间隔中有 3 个并行化的量子门序列，由蓝色虚线分隔。第一个并行化的量子门序列对应第 3 行至第 9 行的第一个 for 循环。第二个并行化的量子门序列对应第 10 行至第 16 行的第二个 for 循环。第三个并行化的量子门序列对应第 17 行至第 25 行的第三个 for 循环。（请注意，根据具体条件，并非所有方框中都会包含全部 3 个序列，例如量子电路开头处。）

算法 5 SPF 子程序

过程 SPF(D, A, m)

▷ D 是大小为 m 的数据寄存器

▷ $A = A_0$、…、A_{m-1} 是角度寄存器，其中 A_j 的大小为 2^j

对于 i，**在** range(m) **中执行**

▷ 每个 i 值占用 $O(1)$ 个深度，如图中的一个方框所示。16

对于 q，**执行** range$(i-1)$

▷ 1^{st} 图16中红线之间空间内的并行化序列。

如果 $(i-q)$ 是奇数**且** $^{3}\overline{(i-q-1)}_{2}$ $-1 \le m-q-3$ **则**

$\bigoplus_{\frac{3(i-q-1)}{2}-1}(D_q)$

否则，如果 $(i-q)$ 是偶数**且** $^{3}\overline{(i-q)}_{2}$ $-2 \le m-q-2$ **则**

$\mathrm{CS}_{\frac{图3(i-q)}{2}-2}(D_q,\ A_{i-1+\frac{i-q}{2}})$

结束条件判断

结束循环

对 q **执行** range(i)**次循环**

▷ 2^{nd} 图中红线之间空间里的并行化序列 图16。

如果（$i-q$）为奇数**且** $^{3}\overline{(i-q-1)}_{2}$ $\le m-q-2$，**则**

$\mathrm{CS}_{\frac{图3(i-q-1)}{2}}(D_q\ A_{i+\frac{i-q-1}{2}})$

否则，如果（$i-q$）为偶数**且** $^{3}\overline{(i-q)}_{2}$ $-2 \le m-q-3$，**则**

$\bigoplus(\frac{3(i-q)}{2}-2,\ D_q)$

结束如果

循环结束

对 q **在** range$(i+1)$ **中执行**

▷ 3^{rd} 图16中红线之间空间里的并行化序列16。

如果 $i == q$ **则**

交换$(D_{q,图4}, A_{i,0})$

否则如果 $(i-q)$ 为奇数 **且** $^{3}\overline{(i-q-1)}_{2}$ $\le m-q-3$ **则**

$\bigoplus_{\frac{3\ (i-q-1)}{2}}(D_q)$

否则，如果$(i-q)$ 是偶数 **且** $^{3}\overline{(i-q)}_{2}$ $-1 \le m-q-2$ **则**

$\mathrm{CS}_{\frac{图3(i-q)}{2}-1}(D_q,\ A_{i+\frac{i-q}{2}})$

结束条件判断

结束循环

结束循环

对于 i **在** reversed(range$(1, m)$) **执行**

▷ i 的每个值占用 $O(1)$ 深度，按逆序运行

对于 q **在** range$(i+1)$ **执行**

▷ q 的所有值并行执行

如果 $i == q$ **则**

继续

否则，如果$(i-q)$ 为奇数**且** $^{3}\overline{(i-q-1)}_{2}$ $\le m-q-3$ **则**

$\bigoplus_{\frac{3\ (i-q-1)}{2}}(D_q)$

否则，如果$(i-q)$ 为偶数**且** $^{3}\overline{(i-q)}_{2}$ $\le m-q-2$ **则**

$\mathrm{CS}_{\frac{图3(i-q)}{2}-1}(D_q,\ A_{i+\frac{i-q}{2}})$

结束条件判断

结束循环

对 q **执行** range(i)**循环**

▷ 并行执行 q 的所有值

如果（$i-q$）为奇数**且** $^{3}\overline{(i-q-1)}_{2}$ $\le m-q-2$，**则**

$\mathrm{CS}_{\frac{\text{<style id='图4'>}(D}{2}\ q\ A_{i+\frac{i-q-1}{2}})$

否则，如果（$i-q$）为偶数**且** $^{3}\overline{(i-q)}_{2}$ $-2 \le m-q-3$，**则**

$\bigoplus_{\frac{3\ (i-q)}{2}-2}(D_q)$

结束如果

结束循环

对于 q **在** range$(i-1)$ **中执行**

▷ q 的所有值并行执行

如果$(i-q)$ 为奇数 **且** $^{3}\overline{(i-q-1)}_{2}$ $-1 \le m-q-3$，**则**

$\bigoplus_{\frac{3(i-q-1)}{2}-1}(D_q)$

否则，如果 $(i-q)$ 为偶数 **且** $^{3}\overline{(i-q)}_{2}$ $-2 \le m-q-2$，**则**

$\mathrm{CS}_{\frac{图3(i-q)}{2}-2}(D_q,\ A_{i-1+\frac{i-q}{2}})$

结束条件判断

▷ 总 $\mathrm{D}_{\text{exact}}$： $O(m)$

结束循环

▷ D 总计$_{\text{approx}}$： $O\ (m)$

结束循环

▷ SA 总计$_{\text{exact}}$： $O\ (m2^m)$

结束过程

▷ SA 总计$_{\text{approx}}$： $O\ (m2^m)$

A.5 FLAG

FLAG 电路的实现方式与 SPF 电路类似，只是做了一些简化。这里，数据量子比特仅作为控制位，不会向数据量子比特注入角度。因此，我们不需要在复制层和 CSWAP层之间交替执行；只需先完成所有复制，再执行所有受控交换门，最后完成所有取消复制操作。FLAG 实现背后的基本思路是，先标记量子比特 0，再将量子比特 0 交换到 p^{th} 标志量子比特中的 p^{th} 位置；为此，使用一系列 CSWAP层，并由不同的数据量子比特作为控制端。我们在图 $m = 4$ 中给出 17 的 FLAG 示例，并在算法 6中给出 FLAG 的伪代码。

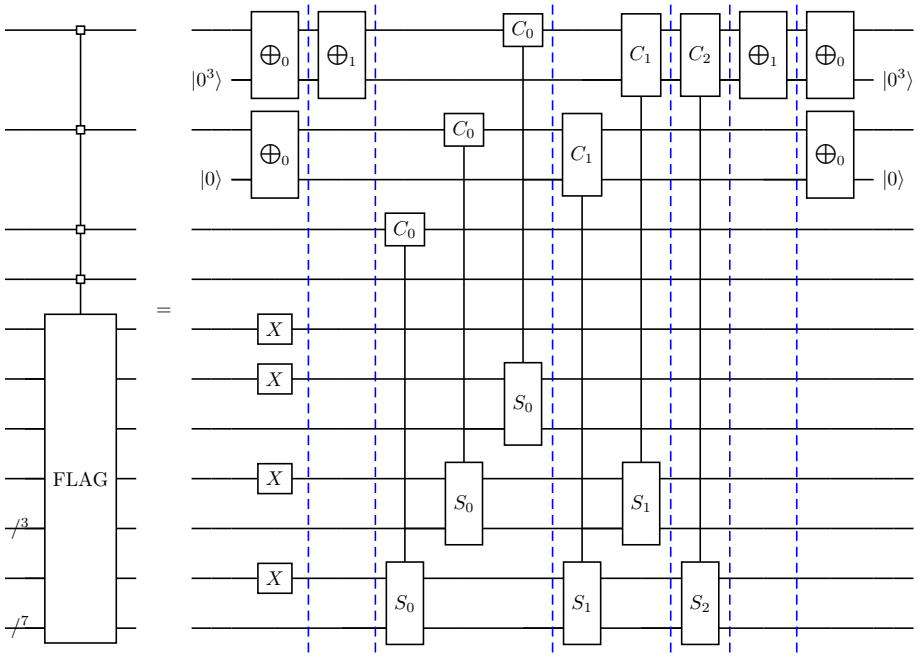


图17： $m = 4$的 FLAG 电路。两条蓝色虚线内的所有量子门序列均可并行执行。

算法6 FLAG 操作

过程 FLAG 子程序(D， F， m)
▷ $D = D_0$，…， D_{m-1} 是数据寄存器，其中 D_j 的大小为 max(1， 2^{m-j-2})
▷ $F = F_0$，…， F_{m-1} 是标志寄存器，其中 F_j 的大小为 2^j
对 j **执行**range(m) **循环**
▷ j 的所有值并行执行
X门($F_{j,图4}$)
结束循环
对 i **执行**range $(m-1)$ **循环**
▷ i 的所有值并行执行
复制(D_i)
▷ 最多占用 $O(m)$深度
结束循环
对于 i **在** range($m-1$) **中执行**
▷ i 的每个值占用 $O(1)$深度
对于 q **在** range($m-i-1$) **中执行**
CS$_i$(D_q<style id='图3'>, F_{q+1+i})
结束循环
结束循环
对于 i，**在** range $(m-1)$ **中执行**
▷ 所有 i 值均并行执行
复制†(D_i)
▷ 最多占用 $O(m)$深度
结束循环
结束过程
▷ 总 D$_{\text{exact}}$： $O(m)$
▷ 总 D$_{\text{approx}}$： $O(m)$
▷ 总 SA$_{\text{exact}}$： $O(m2^m)$
▷ SA 总计$_{\text{approx}}$： $O\ (m2^m)$

A.6 复制交换操作

为简化 LOADF 子程序的电路逻辑，我们定义了另一个子程序操作 CopySwap。要定义 CopySwap，令 $|k\rangle$ 为一个 m量子比特的计算基态，以二进制写作 $k_{m-1}k_{m-2}\ldots k_0$，其中 k_0 表示最低有效位。令 $|\xi\rangle$ 为任意单量子比特态。CopySwap 操作实现了从 $m+1$ 量子比特态到 $2M-1$ 量子比特空间的等距映射

$$\overline{\text{CopySwap}}\left(|k\rangle|\xi\rangle\right) = |k_{m-1}\rangle^{\otimes 2^{m-1}}|k_{m-2}\rangle^{\otimes 2^{m-2}}\ldots|k_1\rangle^{\otimes 2}|k_0\rangle|0^k\rangle|\xi\rangle|0^{M-k-1}\rangle \tag{35}$$

对于任意 m-量子比特计算基态 $|k\rangle$ 和任意单量子比特态 $|\xi\rangle$，高效实现 CopySwap需要结合复制层和并行 CSWAP 门层，如图 18 所示，并使用附录 $\bigoplus_t$ 和 CS_t 中定义的子程序。 A.1与前几节类似，在相应的 $\text{CS}\bigoplus_t$ 操作序列之前尽早执行多个$_t$ ，旨在确保 CS_t 序列能够在 1 步内实现最大程度的并行化。

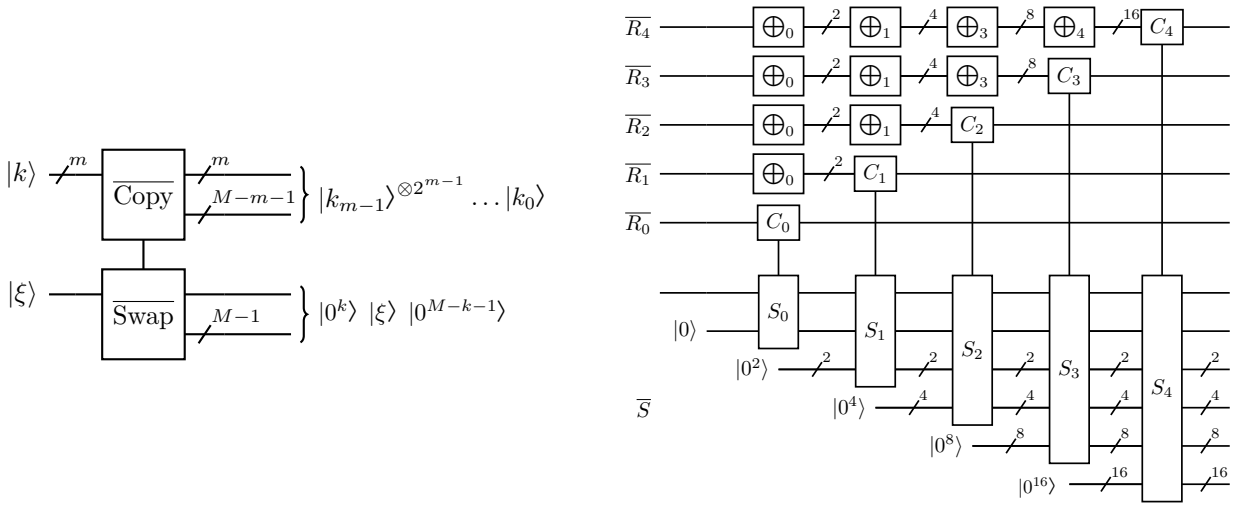


图18：左：CopySwap 操作的作用是同时复制设置为 $|k\rangle$ 的 m 个控制位，并将目标寄存器移至目标寄存器的 k^{th} 位置。输入包含 $m+1$ 个量子比特，输出包含 $2M-1$ 个量子比特，因为协议中引入了若干新的辅助量子比特。右：使用复制层和交换操作实现 CopySwap 操作，适用于 $m=5$。总深度为 m。在第 $t=0$、.....、$m-1$ 层，有 $(m+(2^t-1)(m-t)+2^{t+1})$ 个活动量子比特，因此总时空分配为 $\sum_{t=0}^{m-1}(m+(2^t-1)(m-t)+2^{t+1}) = O(2^m)$。

算法7 复制交换操作

子程序 复制交换子程序(R, S, m) ▷ m 是控制位的数量

▷ $\overline{R} = \overline{R_0}, \ldots, \overline{R_{m-1}}$是控制寄存器，其中 R_j的大小为 $\max(1, 2^{m-j-2})$

▷ $\overline{S}$ 是大小为 2^m的目标寄存器

对于 i **，遍历**$\text{range}(m)$ **，执行** ▷ $O(m)$

对于 j，遍历 range(1, m - i) **执行** ▷ j 的所有值均与 CS 操作并行执行

$\bigoplus_i(R_j)$

结束循环

$\text{CS}_i(R_i, S)$

结束循环

结束过程

▷ 总 D_{exact}： $O(m)$▷ 总 D_{approx}： $O(m)$▷ 总 SA_{exact}： $O(m2^m)$▷ 总 $\text{SA}_{\text{approx}}$： $O(m2^m)$

A.7 LOADF

LOADF 量子电路或许是整个协议得以运行的关键。它有两个控制寄存器、一个大小为 m 的数据˜寄存器，以及一个大小为 B 的标志寄存器：$= N/M - 1$。如果数据寄存器被设为 $|k\rangle$，LOADF 就会读入˜创建对应的 ($n-m$) 量子比特态所需的 B 角度态，但只有当位置 (s,p) 上的标志设为 1 时，才会读入与索引 (s,p) 相关联的角度。从数学上来说，LOADF 实现了以下操作：

$$\mathrm{LOADF}\left(|k\rangle\,|0^{N/M-1}\rangle\left[\bigotimes_{s=0}^{m-n-1}\bigotimes_{p=0}^{2^s-1}|f_{s,p}\rangle\right]\right)=|k\rangle\left[\bigotimes_{s=0}^{m-n-1}\bigotimes_{p=0}^{2^s-1}|f_{s,p}\theta_{s,p}^{(k)}\rangle\right]\left[\bigotimes_{s=0}^{m-n-1}\bigotimes_{p=0}^{2^s-1}|f_{s,p}\rangle\right] \tag{36}$$

LOADF 的一个关键特点是，所有单量子比特旋转门都位于量子电路的同一层。这使其在近似设置下能够实现 $O(\log(N)+\log(1/\epsilon))$ 深度。另一个特点是，$O(N)$ 辅助量子比特只需短暂地充当双重受控旋转门的复制控制位。旋转门执行完毕后，辅助量子比特便可迅速释放，使该协议实现 $O(N)$ 时空分配。LOADF 协议如图19所示，相应的伪代码见算法 8。

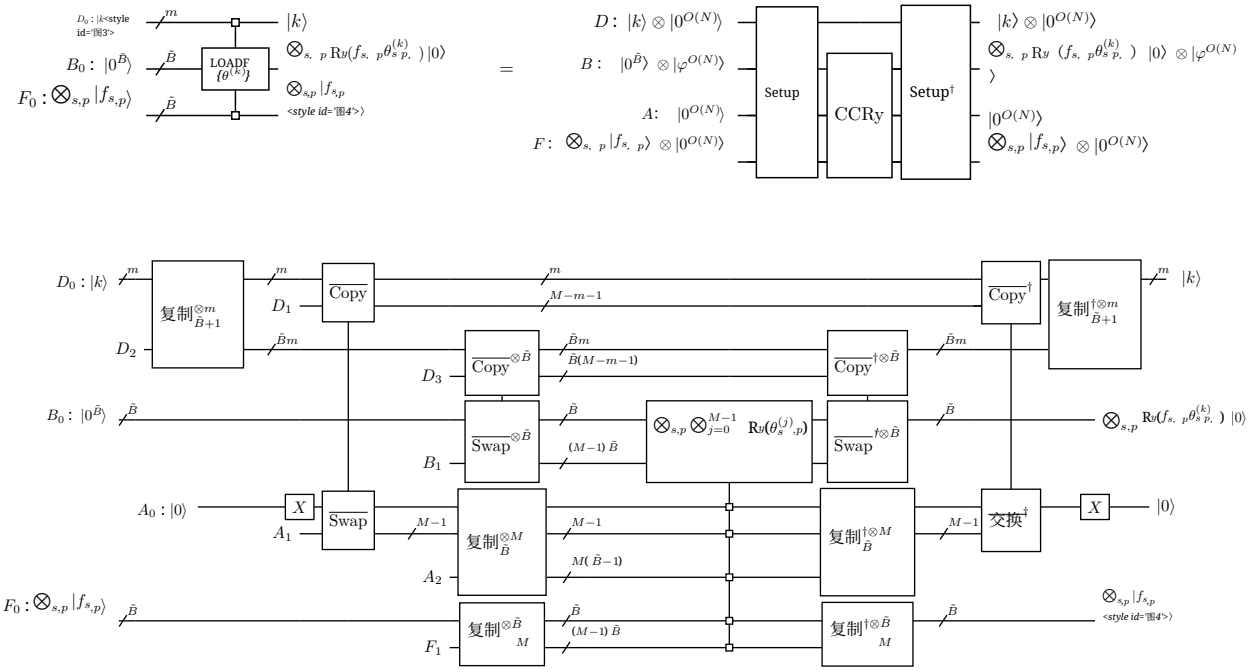


图19：上：LOADF 的模块化抽象。Setup 用于根据地址 $|k$ 准备扩展地址。随后，CCRy 门利用扩展地址并行注入角度。最后，Setup† 反计算所有扩展地址。下：实现 LOADF 的量子电路。数值 $\tilde{B}$ 定义为 $\tilde{B} := N/M - 1$。旋转门层表示 $M\tilde{B} = N - M$ 个角度各不相同的并行 CCR$_y$ 门：$\theta_s^{(k)}{}_{,p}$，以及 $k = 0,\ldots,M-1$, $s = 0,\ldots,n-m-1$ 和 $p = 0,\ldots,2^s-1$。对于每个 CCR$_y$ 门，最后两个寄存器中的一个量子比特（对应于 (s,p) 的选择），以及最后两个寄存器上方三个寄存器中的一个量子比特（对应于 k 的选择）充当控制位。在 CCR$_y$ 序列之前设置复制层是必要的，这样才能并行执行所有这些操作。在 CCR$_y$ 序列之后，我们并行执行 $\tilde{B}$ CopySwap† 操作，以将 $N-M$ 个角度加载到大小为 $\tilde{B}$ 的缓冲辅助量子位中。可以验证，LOADF 实现了 $O(N)$ 的时空分配：只有在旋转门（深度为 $O(1)$）以及随后的取消复制步骤中，$O(N)$ 个辅助量子比特才处于活动状态。不过，图 18 和图 12 显示，对于 $O(N)$ 个量子比特，取消复制步骤仅占用 $O(N)$ 的时空分配。除非另有标注，所有辅助量子比特均以 $|0$ 态开始并结束。我们还注意到，寄存器 B 大部分可以用处于 $|0$ 态的量子比特替换为处于 $|\varphi^{O(N)}$ [16] 的脏量子比特。

已于 Quantum 2024年2月2日接收，点击标题进行验证。以 CC-BY 4.0 许可发布。

算法 8 LOADF 子程序

过程 LOADF 子程序(D, B, F)
▷ 此子程序还需要额外的 O(N) 个辅助量子比特。无需为每个辅助寄存器组预先分配这些量子比特，直到需要单独调用它们时再分配。反计算后，可以释放它们（即可以参与不同的计算任务）。
▷ D_0 是大小为 m 的地址寄存器。 D_1, D_2, D_3 包含所需的 $O(N)$ 个额外辅助量子比特，用于完成所有并行化受控交换门和受控受控 Ry 门。
▷ B_0 是大小为.....的“缓冲区”辅助寄存器 $\tilde{B}$ 用于保存 SPF 将注入数据寄存器的角度。 B_1 包含完成所有并行化受控交换门和受控受控 Ry 门所需的 $O(N)$ 个额外辅助量子比特。
▷ F_0 是大小为 $2^{n-m}-1$ 的 FLAG 寄存器（仅 LOADF† 需要，可选用于 LOADF。由于 FLAG 位全部处于 $|1\rangle$ 态，因此能够保证后续 CCRy 门的控制要求，我们可以有效地将其视为仅由 A 寄存器控制的 CRy 门。）
F_1 包含完成所有并行化受控交换门和受控受控 Ry 门所需的 $O(N)$ 个额外辅助量子比特。
▷ A 是一个大小为 $M-N$ 的寄存器，用于存储 $M-N$ 扩展角度地址（图 8 中未显示）。
▷ 为便于说明，我们忽略子程序参数中的具体索引。我们在6 中给出了包含所有指定索引的精确电路级实现。
    Setup 子程序
    对于 i，遍历 range($N-M$) **执行** ▷ 全部并行，将 $N-M$ 个角度加载到 $N-M$ 个量子比特中
        CCRy(F, A, B, θ) ▷ $O(1)$($O(\log(n/\epsilon))$) 深度
    结束循环
    Setup† 设置子程序
结束过程
过程 设置子程序(D, B, A, F)
    X(A) ▷ $O(1)$ 深度
    对于 i 在 range(m) **执行** ▷ 所有 i 并行执行
        复制$^\dagger_{B+1}(D)$ ▷ $O(n-m)$ 深度
    结束循环
    CopySwap(D, A, m) ▷ $O(m)$ 深度
    ▷ 以下 3 个循环可以在并行
    对于 i 范围内($\tilde{B}$) **执行** ▷ 全部 i 并行
        CopySwap(D, B, $n-m$) ▷ $O(n-m)$ 深度
    结束循环
    对 i 在 range(M) **执行** ▷ 所有 i 并行执行，以在 A 寄存器的 CCRy 层控制位中准备控制比特寄存器
        复制 $_{\tilde{B}}(A)$ ▷ $O(n-m)$ 深度
    结束循环
    遍历 i 范围（ $\tilde{B}$) **执行** ▷ 全部并行执行，以在 F 寄存器
        复制$_M(F)$ ▷ $O(m)$ 深度
    结束循环
结束过程 ▷ 总 D_{exact}： $O(n)$
▷ 总 D^{approx}： $O(n+\log(1/\epsilon))$
▷ 总 SA_{exact}： $O(2^n)$
▷ 总 SA^{approx}： $O(2^n\log(n/\epsilon))$

将 CSP 电路编译为 {H, S, T, CNOT} 门集时，并行化 CCRy 门会带来显著的量子资源开销（$O(\log(n/\epsilon))$ 深度和 $O(2^n\log(n/\epsilon))$ 时空分配）。如图 19 所示，寄存器 B 中执行并行化 Ry 门的大多数量子比特可以是脏量子比特（参见态制备任务 ($m=1$, $n=3$) 的验证代码；其中我们在每个片段前后使用含时门，以模拟 B_1 寄存器中的脏量子比特）。在 LCU [9]等容错算法中，这类量子比特通常供应充足。如果 $\log(1/\epsilon)$ 与 n 的比值较大，那么执行 Toffoli 操作所需的辅助量子比特（即 A 和F 寄存器）在 Toffoli 门执行完毕后即可通过 COPY† 立即反计算，因此所需的干净量子比特时空分配为 $O(N/M\log(\log(N)/\epsilon)+N)$，而非 $O(N\log(\log(N)/\epsilon))$。未来的工作还可以尝试修改电路结构，使例如 D、 A 和/或 F 寄存器也能使用脏量子比特。

已于 Quantum 2024年2月2日接收，点击标题进行验证。根据 CC-BY 4.0 许可发布。

当考虑在整个量子电路中执行 Ry 门所需的干净量子比特数时（SP 电路和 CSP 电路都需要并行化 Ry 门），可以看出，对于 SP 电路，我们实际上需要 $O(M)$ 个干净量子比特，而对于 CSP 电路，则需要 $O(N/M)$ 个干净量子比特。在使用 {H,S,T,CNOT}门集达到较高量子态精度、且有足够脏量子比特供应的情况下，可以令 $M = \sqrt{N}$（换句话说， $m = n/2$，这也满足公式（17）），因此干净量子比特总数现在将上界限定为 $O(\sqrt{N})$，从而使干净量子比特的时空分配为 $O(\sqrt{N}\log(\log(N)/\epsilon) + N)$。

我们使用 Quirk [48] 在图20和图21中展示了辅助量子比特提前释放相较于以往工作的优势。如第1节末尾所述，以往的方法（例如图20所示的 Clader 等人方法，见代码）要求 $O(n)$ 个数据量子比特与 $O(N)$ 个辅助量子比特纠缠，持续 $O(n)$ 的深度，从而导致 $O(n2^n)$ 的时空分配。本工作（如图21所示，见代码；也见图1右侧）只需短暂分配大多数 $O(N)$ 个量子比特。

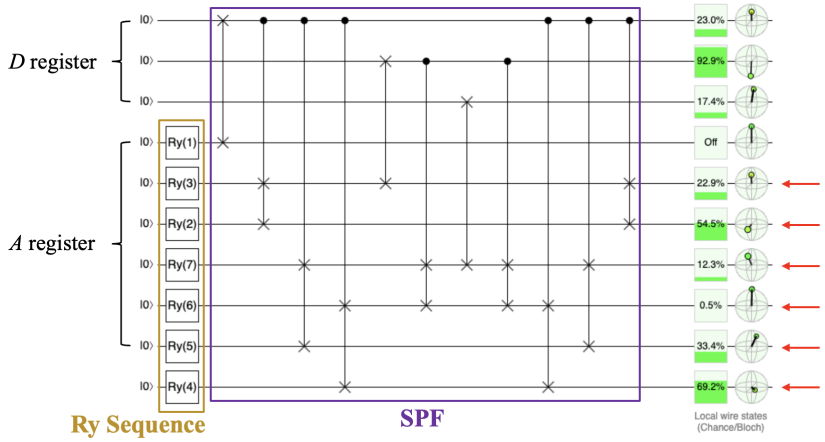


图20：Clader 等人工作的第一部分 [20] （即 U_{SP} 不含复制）。请注意， A 寄存器中的所有辅助量子比特都保持与寄存器中的数据量子比特 D 纠缠（由红色箭头指示）。这些 O（N）辅助量子比特直到执行完深度为$|0\rangle$ 的O（n）FLAG 子程序后才会被返回。

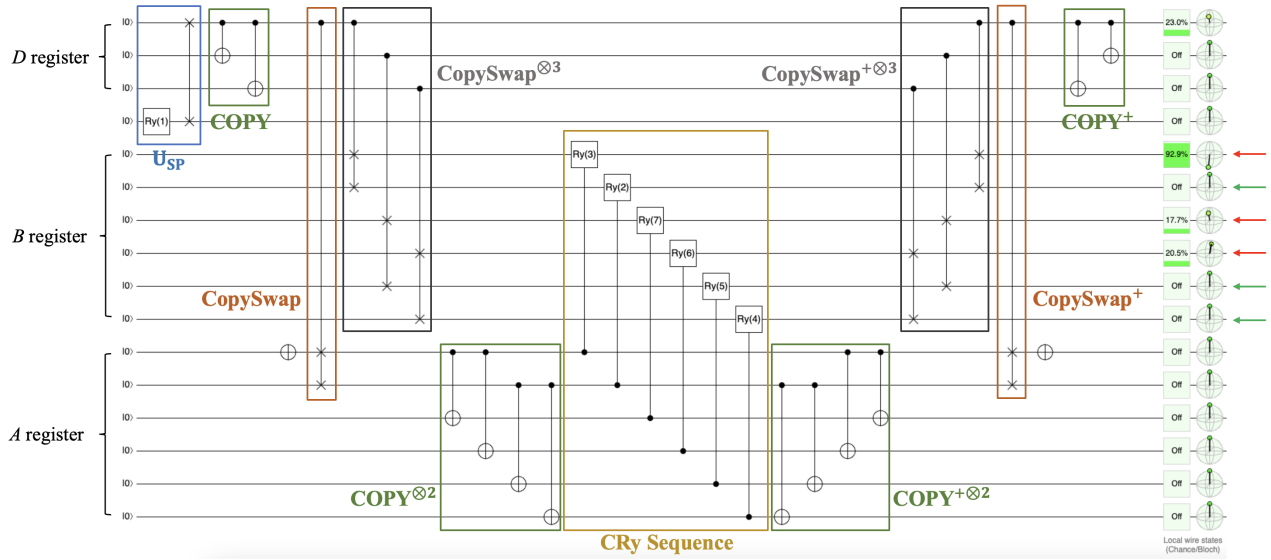


图21： U_{SP} + 本工作开发的量子电路中的 LOADF 部分，以 $m = 1$ 和 $n = 3$为例。请注意， B 寄存器中的大多数辅助量子比特都已释放（绿色箭头所示）。剩余的 O（$\frac{M}{N}$）个辅助量子比特（红色箭头所示）会在后续 SPF 电路之后释放；该电路的深度仅为 O（m），而前一种情况下的深度为 O（n）。 D、 B和 A 寄存器中的所有其他辅助量子比特几乎在初始化后立即释放。请注意，随着 N 增大，且公式（17）中的关系成立时，红色箭头所指量子比特与绿色箭头所指量子比特之间的比例会大得多，这体现了本工作方法的全部优势。

B 当输入态不是 $|0^n\rangle$时我们协议的作用

我们的态制备过程作用于 n 个数据量子比特，并使用数量为 $\ell = O(N)$ 的辅助量子比特。该过程在 $n+\ell$ 个量子比特上实现酉操作 U，将 $|0^n\,|0^\ell$ 映射为 $\mapsto |\psi\,|0^\ell$，也就是说，所有辅助量子比特起始和最终都处于 $|0$。然而，如果对某个量子态 $|z\,|0^\ell$ 执行相同的酉操作（其中 $z \neq 0^n$，或者更一般地，量子态是所有 $z \neq 0^n$ 的叠加态），那么就不再能保证所有辅助量子比特都复位到 $|0$。这是由第 A.4 节所述的 SPF 电路结构导致的：该结构通过将 n 个数据量子比特交换到辅助寄存器中来执行旋转。它利用了数据量子比特已知初始处于 $|0$ 这一事实，从而保证辅助量子比特最终处于量子态 $|0$。

在态制备酉算子属于更大算法的一部分，且并非总是作用于量子态 $|0^n\rangle$ 的应用中，这一特性可能会带来问题。在这些其他情况下，该过程是否仍然有用？它是否仍能实现高效的时空分配？我们在此论证，我们的过程所具备的优良性质，尤其是其最优深度和时空分配，同样适用于态制备常见的其他场景，例如实现反射和投影。

B.1 实现关于任意量子态的反射

在更大算法中使用态制备的一种常见方式，是对特定量子态$|\psi\rangle$实施反射，即该操作

$$R = I_n - 2\,|\psi\rangle\,\langle\psi|\,, \tag{37}$$

其中，I_n 表示作用于 n 个量子比特的恒等算子。

现在我们讨论如何使用 U 实现这一算子。首先，我们区分态制备协议中的不同辅助寄存器。对于协议的 SP 部分，我们有辅助寄存器A和 F，每个寄存器的大小为 $M-1$，如图7所示。对于 CSP 部分，我们还有一个大小为$N/M-1$ 的额外寄存器 B（寄存器 F 与 SP 部分中的相同）。图中显示寄存器 A 和 B始终处于 $|0$，但这仅仅是因为某些数据寄存器的输入为 $|0$。另一方面，F 寄存器具有这样的性质：无论其他寄存器的状态如何，它当且仅当最终处于 $|0$时，初始状态才为 $|0$。此外，SPF、FLAG 和 LOADF 的实现会引入额外的辅助寄存器，如图15、17、19所示；但这些辅助寄存器与 F 寄存器类似：无论量子电路中其他位置的状态如何，它们当且仅当最终处于 $|0$时，初始状态才为 $|0$。因此，我们将 ℓ 辅助量子比特分为两组：一组是位于寄存器 A 和 B中的 $\ell' = M + N/M - 2$ 辅助量子比特，另一组是其他$\ell'' = \ell - \ell'$ 辅助量子比特。

基于这一点，我们可以表示为

$$R \otimes |0^{\ell'}\rangle \otimes |0^{\ell''}\rangle = U\Big[I_n \otimes I_{\ell'} \otimes I_{\ell''} - 2\,|0^n\rangle\,\langle 0^n| \otimes |0^{\ell'}\rangle\,\langle 0^{\ell'}| \otimes I_{\ell''}\Big]U^\dagger\Big[I_n \otimes |0^{\ell'}\rangle \otimes |0^{\ell''}\rangle\Big]\,. \tag{38}$$

让我们验证上述公式。当输入为 $|\psi$ 时，$U^\dagger$ 的作用会得到 $|0^n\,|0^\ell$，并施加一个符号；应用 U 后，输出量子态 $-|\psi\,|0^\ell$，正如预期。当输入为与 $|\psi$ 正交的 $|\perp$ 时，应用 $U^\dagger$ 会得到量子态 $|\perp'\,|0^{\ell''}$，此时我们唯一能保证的是 $|\perp'$ 与 $|0^n\,|0^{\ell'}$ 正交。反射操作不会施加符号，随后应用 U 会输出 $|\perp|0^\ell$，正如预期。至关重要的是，这要求我们同时对数据量子比特 n 和构成寄存器 A 与 B 的辅助量子比特 ℓ' 执行反射，使它们全都处于 $|0$（但不包括其他辅助量子比特，因为可以保证它们已经处于 $|0$）。对处于量子态 $|0$ 的 t 个量子比特执行反射，可通过树形方式计算所有量子比特是否都被置为 $|0$，并在结果为 1 时施加相位，使用深度 $O(\log(t))$、$O(t)$ 个辅助量子比特和 $O(t)$ 个 Toffoli 门来实现。这里 $t = n + N/M + M - 2$，因此深度为 $O(n)$。时空分配的上界为深度乘以活动量子比特数，即 $O(n \cdot \max(N/M, M))$。由于我们选择 M，使得 $\max(N/M, M) = O(N/n)$，因此时空分配至多为 $O(N)$。总之，我们的态制备方法可用于对 n 个量子比特上的任意量子态执行反射，深度为 $O(n)$，时空分配为 $O(N)$。

B.2 实现涉及投影的投影和块编码

另一个与状态制备相关的算子是投影到任意量子态 $|\psi\rangle$ 的正交补上，也就是

$$P = I - |\psi\rangle\langle\psi| \ . \tag{39}$$

例如，该算子会出现在构成查询最优量子线性系统求解器所使用绝热路径的矩阵中 [46]。在该背景下，参考文献 F 的附录 F [46] 解释了如何利用一个将 $|0\rangle \mapsto |\psi\rangle$映射为 P的态制备酉算子，构造相关矩阵的块编码 $|\psi\rangle$（参见参考文献 F 的图 1–6， [49], 尤其是图 4，其中给出了参考文献 F 附录 F 中讨论内容的量子电路解释 [46]）。如上所示，我们的方法能够以最优深度和时空分配执行反射，因此也可应用于此场景。

C 复振幅

正文中介绍的构造可以制备具有实非负系数的任意量子态。将该构造扩展到适用于任意系数的情况非常直接。SP 协议中的 SP+CSP 部分保持不变，因为态 $|\phi\rangle$（如公式 (5) 中所定义）不受向量 $\mathbf{x}$ 中的任何相位影响y_i（所有$k = 0$、...、 $M - 1$均为正数），另一方面，协议的 CSP 部分必须进行略微修改。对于每个 $j = 0$、...、$N/M - 1$，定义相位

$$e^{i\varphi_j^{(k)}} = \frac{x_{j+kN/M}}{|x_{j+kN/M}|} \tag{40}$$

（若 $x_{j+kN/M} = 0$，则相位可以任意定义）。
接下来，我们重新定义量子态 $|\theta_{s\,p,}^{(k)}\rangle$:

$$|\theta_{s,p}^{(k)}\rangle = \begin{cases} \cos(\theta_{s,p}^{(k)}/2)\,|0\rangle + \sin(\theta_{s,p}^{(k)}/2)\,|1\rangle & \text{if } s < n-m-1 \\ e^{i\varphi_{2p}^{(k)}}\cos(\theta_{s,p}^{(k)}/2)\,|0\rangle + e^{i\varphi_{2p+1}^{(k)}}\sin(\theta_{s,p}^{(k)}/2)\,|1\rangle & \text{if } s = n-m-1 \end{cases} \tag{41}$$

要制备 $|\theta_{s\,p}^{(k)}$， 我们需要先施加绕 y 轴的单量子比特旋转；然后，若 $s = n - m - 1$，则必须施加绕 z 轴的单量子比特旋转，并加上一个全局相位（在加入控制位时，全局相位会很重要）。若要制备具有复振幅的量子态，协议中唯一需要改变的部分是 LOADF 操作；该操作在图 8 中 U_{CSP} 的量子电路里出现两次。如图 19 所示，实现 LOADF 需要一个双重控制的 R_y 量子门。为了处理复振幅，我们必须在这一步加入一个双重控制的 R_z 量子门，以及一个双重控制的全局相位（这等价于一个单重控制的 R_z 量子门）。这些新增操作最多会使深度和时空分配增加一个常数因子。在近似 {H, S, T, CNOT} 门集中，为了使态制备协议的总误差达到 ϵ，所有旋转只需合成到误差 $O(\epsilon/n)$，每个旋转仅需 $O(\log(n/\epsilon))$ 个量子门。本文中的复杂度陈述均不受影响。