arXiv:2301.05122v1 [quant-ph] 12 Jan 2023

在量子随机存取存储器中查找最小值的量子算法

Anton S. Albino,[1], * Lucas Q. Galv˜ao,[1], † Ethan Hansen,[2], ‡ Mauro Q. Nooblath Neto,[1], § 和 Clebson Cruz[3], ¶

[1]拉丁美洲量子计算中心，*SENAI CIMATEC*，巴西萨尔瓦多。[2]*Zapata Computing*，加拿大[3]量子信息学小组，精密科学与技术中心，巴伊亚西部联邦大学 - *Edgard Santos* 校长校区。贝蒂奥加街 *892* 号，*Morada Nobre I*，*47810-059* 巴雷拉斯，巴伊亚州，巴西。

在无序数据库中查找最小值是计算机科学中一项常见且基础的任务。然而，最优的经典确定性算法查找数据库中的最小值时，其时间复杂度会随数据库中的元素数量线性增长。本文提出了一种用于查找数据库最小值的量子算法，其速度比最佳经典对应方法快一个数量级（平方加速）。我们假设存在一个量子随机存取存储器（QRAM）来存储数据库中的值，并基于预言机进行迭代搜索；预言机通过控制最高有效量子比特的状态来限制搜索值。我们进行了复杂度分析，以展示该量子算法相较于其经典对应方法的优势。此外，我们还展示了如何通过K-means 算法的量子版本，将所提出的算法用于无监督机器学习任务。

关键词：量子随机存取存储器、最小值搜索、Grover算法

I. 引言

随机存取存储器（RAM）是一种用途广泛的短期存储器，在计算中通过比特存储和检索信息 [1]。类似地，量子随机存取存储器（QRAM）的概念也应运而生，其目标相同，但使用量子比特来实现态叠加，从而在量子或经典计算应用中更快地获得结果 [2–11]。已有多项研究探讨了其应用潜力，例如优化量子算法的执行，包括在经典数据库中进行量子搜索 [10, 12–15], 碰撞查找 [12, 16–18], 以及求解线性方程组的算法 [19–22], 。

过去几年，这些成果引起了科学界的关注，并推动了量子随机存取存储器架构的发展；这些架构展现出在量子计算中实现高效结果的潜力[3, 4, 6, 7, 9, 23–25]。一些模型，例如扇出式量子随机存取存储器 [3, 7, 9] 和桶式接力量子随机存取存储器 [3, 23, 25],，展示了量子随机存取存储器在实际场景中未来可能的实现方式。此外，近期实验揭示了一种专为混合量子计算机设计的架构，该架构采用超导量子比特和自旋量子比特存储晶体；理论上，这种架构能够在实际系统中实现量子随机存取存储器 [11]。

此外，人们还努力构建能够在搜索 QRAM 单元中存储的特定值时，以最优方式访问量子随机存取存储器的量子算法[10]。一般而言，搜索类问题会使用著名的Grover算法，在无结构列表中搜索量子态 [16, 26–29]。另一方面，确定列表中最小值的一个知名例子是所谓的 D¨urr-Hoyer 最小值查找算法 [30], , 它将 Grover算法用作基本子程序，以查找列表中的最大或最小元素 [31]。

在这种情况下，基于 Durr-Hoyer 算法的核心概念，我们将 Grover算法用作子程序，以开发一种量子算法，用于在存储于 QRAM 中的经典数据集中找出最小值。所提出的量子最小值搜索（QMS）算法基于预言机函数的迭代变化，通过控制最高有效量子比特的状态来限制搜索值。首先，我们介绍 QMS 算法的描述，阐述 QRAM 的概念，并通过一个示例，说明如何使用所提出的算法在由四个实数值构成的列表中寻找最小值。接着，我们分析 QMS 算法与经典算法的复杂度。结果表明，经典算法的复杂度随数据库中元素数量线性增长，为 $O(N)$，因为经典算法会遍历列表中的全部 N 个项目；而本文提出的 QMS 算法的复杂度为 $O(\sqrt{Nt})$，其中 t 是标记态的数量。最后，我们展示了所提出算法在 K 均值问题中的应用，即确定 K 个质心的最优位置，以最小化各点与其对应质心之间的距离总和。

II. 量子最小值搜索（QMS）算法

搜索问题是经典计算机科学中一个广泛讨论的主题 [26]。该问题是在一个包含 N 个元素的无序数据库中，找出满足预先设定的搜索条件 $x = y$ 的数据库项（x）的索引，其中 y 是要查找的元素。在这种情况下，可以构造所谓的响应函数（$R(x)$）：如果数据库条目 x 符合搜索条件（$x = y$），则将其映射为 True；如果 $x \neq y$，则映射为 False。这可以通过使用所谓的预言机子程序来实现，该子程序会不断查询数据库，直到找到所需的项。

* anton.albino@fieb.org.br
† lucas.g5847@ufob.edu.br
‡ iethanhansen@proton.me§ mauro.neto@fbter.org.br
¶ clebson.cruz@ufob.edu.br

因此，所请求元素在列表中的位置越靠后，找到它所需的查询次数就越多。因此，这项任务的复杂度与列表中的项目数完全成正比 [26, 27]。平均而言，需要进行 $N/2$ 次查询，因此经典搜索问题的复杂度定义为 $O(N)$ 量级 [29]。

Grover提出的著名量子搜索算法用于搜索非结构化数据集，是展示量子计算相较于经典对应方法优势的一种应用 [26]。量子叠加概念的引入使算法能够同时映射所有数据库条目，从而减少查询总次数，提高搜索过程的效率 [27]。在这方面，Grover算法的复杂度呈$O(\sqrt{N})$阶增长，比其经典对应方法快一个平方级 [29]。因此，许多算法将Grover方法用作子程序，以优化某些量子过程。一个著名的例子是所谓的D¨urr-Hoyer算法，用于在非结构化数据库中寻找最小值 [32]。该算法基于Grover算法，在最小值搜索问题中实现平方级加速，其复杂度可表示为$O(\sqrt{Nt})$，其中t是标记态的数量 [30]。

在此背景下，本研究探讨量子最小值搜索（QMS）问题，并提出一种量子算法，用于在量子随机存取存储器（QRAM）中寻找最小值，其速度比任何经典同类算法快一个二次方。我们描述了该算法，并考虑由向量 $\vec{y}$ 表示的数据集，该向量可改写为计算基中的量子态。问题是使用基于 Durr-Hoyer方法的 Grover算法作为子程序，找出列表y_{min}中的最小值。

A. 量子随机存取存储器（QRAM）

为了使用 Grover算法在经典数据集中寻找最小值，一种方案是使用 QRAM，通常指一种大型经典存储器，可在量子叠加状态下进行查询。它可以通过一个等效的量子电路构建，其中经典数据以二进制形式存储在量子寄存器中。具体做法是创建两个量子寄存器（分别包含 n 和m 个量子比特），其初始化应为

$$|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes |0\rangle^{\otimes m}, \tag{1}$$

这可以通过操作$H^{\otimes n} \otimes I^{\otimes m}$ 来实现：该操作在第一寄存器中创建均匀叠加态，并使第二寄存器保持在状态 $|0\rangle^{\otimes m}$。通过在第二寄存器中应用由多重受控非门操作构成的算子U_X，即可实现量子随机存取存储器。目标是将经典值 $\vec{y} = \{y_0, y_1, y_2, ..., y_k\}$ 存入量子状态，以便获得

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \otimes |y_x\rangle. \tag{2}$$

因此，量子随机存取存储器可以存储 2^n 个数据值。在这种情况下，我们需要选择第二寄存器所用量子比特的数量m。由于我们要搜索整个数据集中的最小值，可以将一个随机索引作为第一次迭代。因此，可以指定数量m，使得与该索引关联的数能够以计算二进制基表示。

B. 在量子随机存取存储器中寻找最小值

为了在量子随机存取存储器中找到存储的最小值，采用的策略是通过单次测量分析最高有效位（如果我们要找最大值，则分析最低有效位）来执行搜索。完整的量子电路见图1，其中，负责根据最高有效量子比特进行搜索的特殊子程序是迭代相位翻转，由算子P表示。

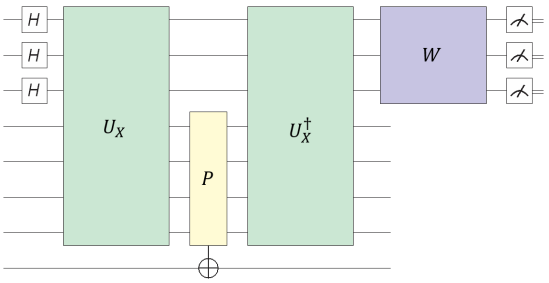

图 1. 用于最小值搜索的完整量子电路。U_X 表示量子随机存取存储器；算子P 会根据上一次测量得到的最高有效量子比特进行迭代调整；W 是扩散算子。需要强调的是，最后一个量子比特的状态初始化为 $|-\rangle$。

该算法的核心思想在于 P 算子的动力学。附加寄存器（图 1 中进一步向下的量子比特）用于表示量子随机存取存储器中经典值的存储位置，同时也是搜索进行的地方。众所周知，如果最高有效量子比特处于 0 状态，则在十进制中，该数小于最高有效量子比特处于 1 状态时的数值。基于这一逻辑，可以通过多重受控非门构造 P 算子，其中量子比特既可以以 0 为控制态，也可以以 1 为控制态。因此，支配 P 动力学的算法如算法框 1 所述。

算法 1 在量子随机存取存储器中寻找最小值

- **输入** 经典数据库 $\vec{y}$

  1. 随机选取一个值 $y_i = f(x_i)$，其二进制表示需要 m 位。
  2. 将量子计算机初始化为状态 $|\psi_0\rangle = \frac{1}{\sqrt{2}} \sum_{=0}^{2^n} |\ \rangle|0\rangle^{\otimes m}|-\rangle$ $_n$ $_x x$。 3. 将经典值存储在量子随机存取存储器中，以获得状态 $|\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n} |x\rangle|y_x\rangle|-\rangle$ 4. 应用预言机算子 P，以确保最高有效位为 0，也就是说，标记态小于 y_i。 5. 应用扩散算子 W，以放大标记态。 6. 在计算基中进行测量，得到 $y_{i+1} < y_i$。 7. 如果所有量子比特均已分析：– **结束 if** 否则：– **重复**这些步骤

- **返回** y_i

因此，可以迭代使用Grover算法来放大对应于比上一个更小值的状态（索引），其速度比经典方法快平方级。例如，假设有以下数据集 $\vec{y} = \{5, 4, 12, 10, 8\}$，则列表中的条目可用计算基（四个量子比特）表示为 $\vec{y} = \{|0101\rangle、|0100\rangle、|1100\rangle、|1010\rangle、|1000\rangle\}$。

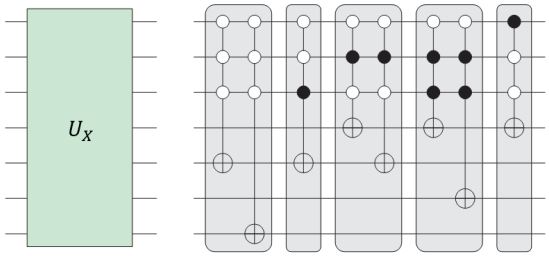

图 2. 量子随机存取存储器的实现，其由算子 U_X 表示，并以量子电路形式呈现。每个灰色模块存储数据库中的一个经典值。

如果第一个猜测（纯粹基于经典方法）是 $10 \rightarrow 1010$，那么这个数最小的可能性很低。可以通过查找最高位量子比特为 $|0\rangle$ 的数来验证这一点。

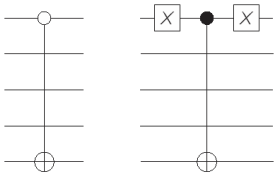

图 3. 用于搜索最高有效量子比特处于状态 $|0\rangle$ 的所有状态的量子算子 P。

因此，使用这个预言机进行一次 Grover迭代，会标记所有最高位量子比特处于 $|0\rangle$ 状态的状态。测量前状态的示意图见图 4。

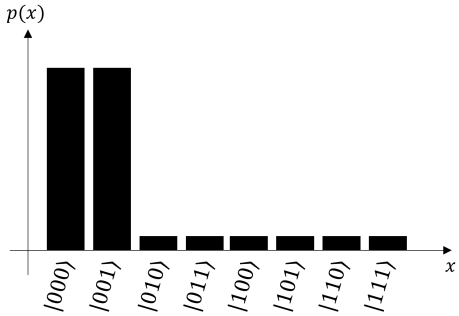

图 4. 量子态概率的示意图。状态 $|x_0\rangle\rangle = |000\rangle$ 和 $|x_1\rangle = |001\rangle$ 会被放大，因为 $f(x_0) = |0101\rangle$ 和 $f(x_1) = |0100\rangle$ 的最高有效量子比特均为 $|0\rangle$。

之后，在该最高有效量子比特上执行 Grover 搜索，状态 $|0100\rangle$ 和 $|0101\rangle$ 被测得的概率将相等。如果测量后得到状态 $|0101\rangle$，下一步就是搜索前两个二进制位为 $|00\rangle$ 的值。

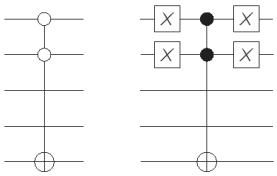

图 5. 用于搜索前两个最高有效量子比特处于状态 $|00\rangle$ 的所有状态的量子算子 P。

如果有一个或多个值满足条件，则测量到小于 $|0101\rangle$ 的数的概率较高（$y_i > y_{i+1}$）；如果没有，则测量到较大数的概率较高（$y_i < y_{i+1}$），因为初始状态未执行任何旋转。

在 $y_i < y_{i+1}$ 的情况下，该过程表明最小值为 $|0101\rangle$，或者是一个更小的数，其前两个最高有效量子比特也为 $|01\rangle$，因此有必要搜索第三个最高有效量子比特为 $|010\rangle$ 的值。在这种情况下，剩下的唯一任务是验证 $|0100\rangle$ 是否存在于量子随机存取存储器中，因为它是三个最高有效量子比特处于 $|010\rangle$ 状态的可能数值中最小的。

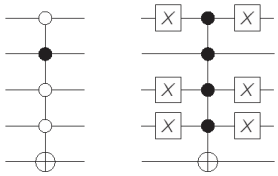

图 6. 用于检查值 $|4\rangle \equiv |0100\rangle$ 是否在量子随机存取存储器中的量子算子P。

在这个例子中，状态 $|0100\rangle$以约 100% 的概率被测量得到（见图 7）。该过程对其余量子比特迭代执行，以确定最小值$y_{min} = |0100\rangle$ 确实如此。

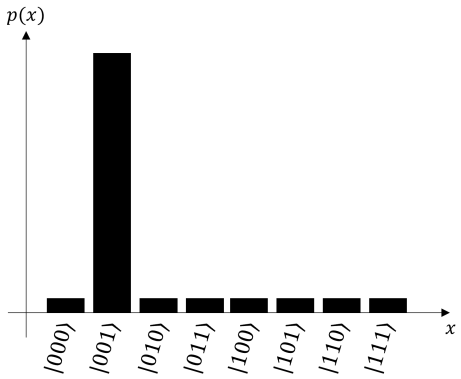

图 7. 最后一次迭代的最终分布。在这种情况下，$f(001) = y_{min} = |0100\rangle$。

向量中的最小值可通过最优算法找到，时间复杂度为 $O(|\vec{y}|)$。本文提出的量子算法通过在 Grover 预言机中进行 $O(c\sqrt{|\vec{y}|}t)$ 次查询，在量子计算机上解决同一问题。其中，c 是一个常数，其值为初始值二进制表示中的位数，而t 是标记态的数量。根据复杂度理论，常数不会影响时间复杂度，因此可以将该算法的时间复杂度改写为$O(\sqrt{|\vec{y}|}t)$。

C. 复杂度分析

为了分析并比较经典算法与量子算法在不同场景下的时间复杂度，我们选取了两种复杂度不同的经典算法。对于所提出的量子算法，也采用了相同的方法，但通过增加数据库值的位数来提高复杂度（见图 8）。我们知道，经典算法和量子算法的复杂度分别为 $O(c_c N)$ 和 $O(c_q\sqrt{N})$。常数 c_c 和 c_q 分别表示每种经典算法固有的常数复杂度因子，以及量子算法初始猜测所包含的位数，正如流程中所述。

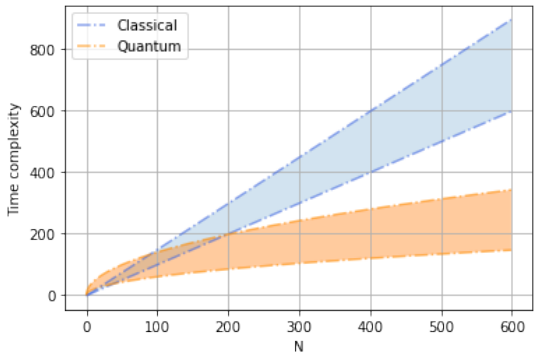

图 8 线条之间的阴影区域表示经典算法与量子算法的复杂度范围。经典算法（蓝色）的上界和下界的时间复杂度分别为 $O(\frac{3}{2}N - 2)$ 和 $O(N - 1)$。对于量子算法（橙色），分别绘制了 $c_q = 14$ 和 $c_q = 6$ 两种情况下的上限和下限。

III. K 均值聚类

为了展示最小值搜索在计算机科学重要任务中的应用，我们实现了名为 K均值 的聚类算法；该算法在统计学和无监督机器学习领域广为人知。给定欧几里得空间中的一组点，该算法旨在确定 K 个质心的最佳位置，从而使各点与其对应质心之间的距离总和最小化。目标函数可表示为

$$f(x,y) = \sum_{j=0}^{K}\sum_{i=0}^{|S|} \|p_i^{(j)} - c_i\|^2 \tag{3}$$

其中 p_i 是欧几里得空间中的一个观测点，$|S|$ 是观测点的总数，c_i 是一个质心的位置，而K 是预先确定的质心数量。图 9展示了笛卡尔平面中的点分布，以及在优化过程开始前随机初始化的质心。虽然这是一个简化的示例，但它有助于直观地说明算法的工作原理。

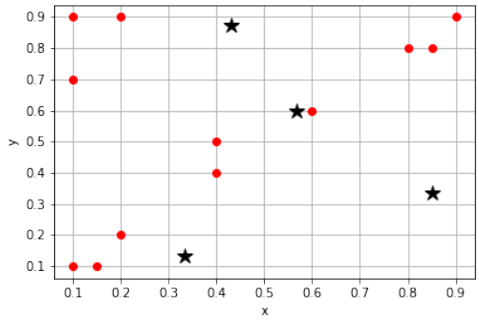

图 9. 二维欧几里得空间中 $|S| = 12$ 点（红色）的分布。在此示例中，四个质心（$K = 4$）以星形（黑色）表示，并通过随机初始化得到。

我们假设，在 K均值的量子版本中，每个观测点 p_i 与所有质心 $S = \{c_0, c_1, c_2, c_3\}$之间的距离都以恒定时间存储在QRAM中，即 $O(1)$。在每次迭代中，根据每个点与其最近质心之间的接近程度形成簇。计算每个新簇坐标的平均值，并将其作为新的质心。该过程持续进行，直到满足某个停止准则。图 10展示了该算法找到的最佳聚类结果。

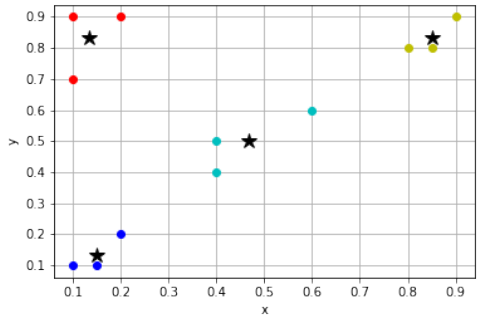

图 10. 通过 K 均值的量子版本找到的最优解。算法找到的四个簇分别用不同颜色表示。

请注意，此过程与其经典对应方法之间唯一的区别是各点与所有质心之间的距离都存储在 QRAM 中，并使用量子最小值搜索来找出最小值。尽管这里用量子最小值搜索来处理一个具体示例，但它也适用于大量计算任务，例如无监督机器学习问题。

IV. 结论

从计算复杂度较低的问题到计算复杂度较高的问题，量子计算在广泛的问题领域中都能胜过经典计算。经典最小值搜索问题具有线性复杂度，并与计算机科学领域中的广泛应用相关。在此背景下，本文提出了一种用于查找数据库中最小值的量子算法，其速度比最佳经典对应方法快一个数量级。该算法基于 D¨urr-Hoyer 查找无序列表中最小值的方法，并将 Grover算法作为子程序应用于存储了指定数据库数值的 QRAM。虽然这并非一项复杂任务，但我们的结果表明，对于数据库规模极大的情况，所提出的 QMS 算法有望显著缩短最小值搜索算法的执行时间。此外，我们还分析了所研究问题的复杂度，以突出该量子算法相较于其经典对应方法的优势。我们还展示了如何通过对 K 均值算法进行量子改编，将所提出的方法应用于无监督机器学习任务。总之，我们的结果表明，可以利用 QRAM 中存储的信息在经典数据库中搜索最小值，这为容错量子算法的发展作出了重要贡献。

致谢

我们感谢拉丁美洲量子计算中心和 SENAI CIMATEC 的高性能计算中心对本研究的支持。我们还感谢量子开源基金会（QOSF）的导师计划，该计划中开发的项目促成了本文的诞生。

[1] A. Sedra 和 K. Smith，微电子电路（牛津大学出版社，2004 年）。[2] V. Giovannetti、S. Lloyd 和 L. Maccone，《物理评论快报》**100**，160501（2008 年）。[3] V. Giovannetti、S. Lloyd 和 L. Maccone，《物理评论A》**78**，052310（2008 年）。[4] D. K. Park、F. Petruccione 和 J.-K. K. Rhee，《科学报告》**9**，1（2019 年）。[5] P. Yuan 和 S. Zhang，arXiv 预印本 arXiv:2202.11302（2022 年）。[6] K. Phalak、J. Li 和 S. Ghosh，arXiv 预印本 arXiv:2210.14804（2022）。[7] R. Asaka、K. Sakai 和 R. Yahagi，《量子科学与技术》**6**，035004（2021）。[8] T. M. De Veras、I. C. De Araujo、D. K. Park 和 A. J. Da Silva，《IEEE计算机汇刊》**70**，2125（2020）。[9] C. T. Hann、G. Lee、S. Girvin 和 L. Jiang，《PRX量子》**2**，020311（2021）。[10] O. Di Matteo、V. Gheorghiu 和 M. Mosca，《IEEE量子工程汇刊》**1**，1（2020）。

[11] M. Blencowe, 《自然》**468**，44（2010）。[12] T. Hur、L. Kim 和 D. K. Park, 《量子机器智能》**4**，1（2022）。[13] B. Broda, 《欧洲物理学杂志：特刊》**131**，1（2016）。[14] V. Giovannetti、S. Lloyd 和 L. Maccone, 《物理评论快报》**100**，230502（2008）。[15] S. Lu、Y. Zhang 和 F. Liu, 《量子信息处理》**12**，3265（2013）。[16] A. Hosoyamada、Y. Sasaki、S. Tani 和 K. Xagawa, 《理论计算机科学》**842**，100（2020）。[17] M. Naya-Plasencia、A. Schrottenloher、A. Chailloux 和 L. Grassi，载于《*QuAC*：用于密码分析的量子算法》（2019）。[18] X. Bonnetain、A. Chailloux、A. Schrottenloher 和 Y. Shen，arXiv 预印本 arXiv:2205.14023（2022）。[19] B. Duan、J. Yuan、C.-H. Yu、J. Huang 和 C.-Y. Hsieh, 《物理快报A》**384**，126595（2020）。[20] L. Wossnig、Z. Zhao 和 A. Prakash, 《物理评论快报》**120**，050502（2018）。[21] I. Kerenidis 和 A. Prakash, 《物理评论A》**101**，022316（2020）。[22] C. Shao 和 H. Xiang, 《物理评论A》**101**，022322（2020）。[23] S. Arunachalam、V. Gheorghiu、T. Jochym-O'Connor、M. Mosca 和 P. V. Srinivasan, 《新物理学杂志》**17**，123010（2015）。[24] L. Bugalho、E. Z. Cruzeiro、K. C. Chen、W. Dai、D. Englund 和 Y. Omar，arXiv 预印本 arXiv:2210.13494（2022）。[25] A. Paler、O. Oumarou 和 R. Basmadjian, 《物理评论A》**102**，032608（2020）。[26] M. A. Nielsen 和 I. Chuang，量子计算与量子信息（美国物理教师协会，2002）。[27] C. Figgatt、D. Maslov、K. A. Landsman、N. M. Linke、S. Debnath 和 C. Monroe, 《自然·通讯》**8**，1（2017）。[28] R. Seidel、C. K.-U. Becker、S. Bock、N. Tcholtchev、I.-D. Gheorge-Pop 和 M. Hauswirth, 《量子科学与技术》（2023）。[29] P. J. Szabłowski, 《量子信息处理》**20**，1（2021）。[30] C. Durr 和 P. Hoyer，arXiv 预印本 quant-ph/9607014（1996）。[31] N. Wiebe、A. Kapoor 和 K. M. Svore, 《量子信息与计算》**15**，318（2015）。[32] Y. Chen、S. Wei、X. Gao、C. Wang、Y. Tang、J. Wu 和 H. Guo, 《量子信息处理》**19**，1（2020）。