arXiv:2603.26232v1 [cs.DC] 27 Mar 2026

ParaQAOA：面向超过 10,000 个顶点的大规模最大割问题的高效并行分治 QAOA

Po-Hsuan Huang，国立台湾大学，台湾李协儒，
国立成功大学，台湾庄驰，国立成功大学，台湾
Chia-Heng Tu*，国立成功大学，台湾Shih-Hao
Hung，国立台湾大学，台湾

量子近似优化算法（QAOA）作为一种采用量子-经典混合框架求解组合优化问题的有前景方案，已逐渐崭露头角。在各类组合优化问题中，最大割问题因其在多个领域的广泛适用性而尤为重要。尽管已有基于 QAOA 的最大割求解器，但它们主要侧重于解的准确性，而非执行效率，这在很大程度上限制了其在大规模问题中的实用性。为解决这一限制，我们提出 ParaQAOA，这是一种利用并行计算硬件高效求解大规模最大割问题的并行分治 QAOA 框架。ParaQAOA 将大规模问题划分为多个子问题并并行求解，在保持解质量的同时显著缩短运行时间。这一设计不仅能够扩展至包含数万个顶点的图，还能灵活调节准确性与效率之间的权衡，使 ParaQAOA 能够适应不同的性能需求。实验结果表明，在包含 400 个顶点的最大割问题上，ParaQAOA 相较于最先进的方法可实现最高 1,600 倍的加速比，同时解的准确性与已知最佳解的差距不超过 2 %。此外，ParaQAOA 仅用 19 分钟便求解了一个包含 16,000 个顶点的实例，而已知的最佳方法则需要 13.6 天以上。这些发现确立了 ParaQAOA 作为一种实用且可扩展的框架，可在严格的时间限制下求解大规模最大割问题。

CCS 概念： • **计算方法学** → **大规模并行与高性能模拟**； • **计算数学** → **近似算法**； • **计算机系统组织** → **量子计算**。

其他关键词和短语：量子计算、量子电路模拟、量子近似优化算法、并行计算、性能效率指数、最大割问题

1 引言

量子计算是一种新兴的计算范式，其发展超越了经典计算机的可扩展性限制。通过利用叠加、纠缠和干涉等量子力学现象， [14,34], 量子计算机能够实现指数级加速比 [3, 28, 36, 37]，从而解决特定计算任务，使其非常适用于密码学、机器学习和组合优化等领域的一系列计算密集型问题。在理论算法和硬件开发迅速进步的推动下，

*通讯作者

作者联系信息： Po-Hsuan Huang，aben20807@gmail.com，计算机科学与信息工程学系，国立台湾大学，台北，台湾； 李协儒，p76134587@gs.ncku.edu.tw，计算机科学与信息工程学系，国立成功大学，台南，台湾； 庄驰，ab321013@gmail.com，计算机科学与信息工程学系，国立成功大学，台南，台湾； Chia-Heng Tu，chiaheng@ncku.edu.tw，计算机科学与信息工程学系，国立成功大学，台南，台湾； Shih-Hao Hung，hungsh@csie.ntu.edu.tw，计算机科学与信息工程学系，国立台湾大学，台北，台湾。

量子计算已引起全球广泛关注，并日益被视为下一代计算解决方案的关键技术 [2, 42]。

在各种量子计算范式中，混合量子-经典算法通过发挥量子与经典计算资源的互补优势，展现出极强的适用性，可部署于当前量子硬件平台。这些混合方法在利用实际量子优势的同时，缓解了当代量子硬件固有的局限性。量子近似优化算法 [13] 是一个典型例子，它为解决二次无约束二进制优化（QUBO） [23] 问题提供了坚实的理论基础，并已在各种应用领域展现出良好的实际性能。特别是，QAOA 的量子-经典优化框架已被证明特别适合解决现实应用中面临的组合优化问题。

最大割问题是基于QAOA的优化的经典应用领域，既是计算复杂度领域的基本基准问题，也是具有现实影响的实际优化问题。最大割问题旨在将图的顶点划分为两个互不相交的集合，使跨越两个集合的边数最大化。作为一种典型的 NP 难组合问题，最大割在多个领域都有直接应用，例如 VLSI 电路设计优化 [4], 社交网络社区检测 [1, 24], 以及无线网络频率分配 [43]。该问题适用范围广且计算难度固有，因此需要高效、可扩展的算法，以便在保持解的质量的同时处理大规模实例。

已有许多基于量子近似优化算法的方案被用于解决最大割问题。随着问题规模扩大，传统的量子近似优化算法实现会消耗大量资源，其电路深度和门复杂度会随量子比特数呈多项式增长。这些资源需求使量子近似优化算法的实现在当前量子和经典硬件上难以实现，从而限制了其在现实应用中的实际部署。为此，近期的分治方法 [29, 31, 46] 采用问题分解技术，例如图聚类、层次划分和递归细分，并结合在经典计算机上进行的量子电路模拟。遗憾的是，尽管这些方法降低了子问题的计算负担，并优先考虑保留解的质量（以近似比衡量，如第2.1节所述），但通常会增加总体执行时间，从而限制了它们在大规模问题中的适用性。例如，在第4.3节的实验中，Coupling QAOA [31] 在一个含30个顶点的图上取得了约99%的近似比，但需要约八小时才能得出最大割结果。这一结果表明，在计算效率与解的质量之间取得平衡，仍然是将量子近似优化算法扩展至实际问题规模所面临的核心挑战。

在本研究中，我们提出了 *ParaQAOA*，这是一种并行分治 QAOA 框架，利用现代并行计算硬件高效解决大规模最大割问题。ParaQAOA 框架包含四个关键组成部分。首先，该框架采用一种线性时间图划分算法，可降低分解复杂度，并高效处理大型图。其次，框架利用现代并行架构构建并行化执行流水线，以处理子问题求解和解重构，从而缩短总体运行时间。第三，该框架提供系统化的参数化设计，以更好地控制并行执行，并管理执行效率与解的质量之间的权衡。第四，ParaQAOA 引入了一种统一指标，可综合评估解的质量和执行效率，从而支持对不同最大割解进行一致比较，并据此做出明智的权衡。该框架的主要特点将在第 2.5 节中进一步详述。本研究的贡献如下。

（1）提出了一种并行分治 QAOA 框架 ParaQAOA，利用经典计算机上的并行计算架构高效求解大规模最大割问题。据我们所知，ParaQAOA 是首个展示如何在经典计算机上高效求解包含超过 10,000 个顶点的最大割问题的工作，并能够控制解的质量与执行效率之间的权衡；当用户希望快速获得解、即使这意味着牺牲解的质量时，这一点尤为重要。

(2) 解的质量与执行效率之间的权衡是最大割求解器设计中的一个重要考量，也是关键的设计要素。在评估不同的最大割解时，尤其是针对大规模问题，这一点至关重要。为此，我们提出了性能效率指数（PEI），这是一种综合考量近似质量和运行时间效率的新型评估指标。此外，ParaQAOA 的参数化设计使得这一权衡可控。

(3) 我们实验表明，ParaQAOA 能有效平衡解的质量与执行效率。例如，在 400 个顶点的实例（中等规模问题）上，ParaQAOA 的加速比最高可达 1,600× ，同时近似比与已知最佳解相差不超过 2%。此外，先前已知的最佳方法需要 13.6 天才能求得一个 16,000 个顶点的图的结果，而 ParaQAOA 仅需 19 分钟即可得出结果。这些结果展现了 ParaQAOA 在大规模最大割问题中的适用性，尤其适用于时间限制严格的场景。

本文其余部分安排如下。第2 节介绍最大割问题中 QAOA 的背景，介绍现有的基于 QAOA 的最大割问题解决方案，分析其局限性，并详细阐述我们提出该框架的动机。第3 节详细说明 ParaQAOA 框架的整体系统设计及其关键组件。第4 节给出实验评估与分析，以展示我们方法的有效性。最后，第5 节总结本文并探讨未来工作的潜在方向。

2 背景与动机

本节阐述所提出的 ParaQAOA 框架的理论基础与动机，重点关注最大割问题及其在文献中量子优化研究方面的意义。第2.1节介绍最大割问题的形式化定义。第2.2 节概述如何应用量子近似优化算法求解最大割实例，第2.3 节讨论 QAOA 的量子电路模拟。随后，我们在第2.4 节回顾现有的求解最大割问题的基于 QAOA 的方法，并总结其策略和局限性。最后，在第2.5节中，我们介绍促使我们开发 ParaQAOA 的观察结果。这些观察基于现有方法的局限性，如第2.4节所述。此外，我们还强调了本框架在处理大规模最大割问题时的特点。

2.1 最大割问题

最大割问题 [16, 22] 因其在计算复杂性理论和实际应用中的双重重要性，已成为量子优化算法的基础基准 [1, 4, 11, 24, 43]。最大割属于 NP 难组合优化问题，即使是在多项式时间范围内寻求近似解，经典算法也难以在大型实例上求得最优解。形式上，最大割问题定义在无向图 $G = (V, E)$ 上，其中 V 表示顶点集，包含 $|V|$

个节点，而 E 表示由 $|E|$ 连接的边集。每条边 $(i, j) \in E$ 都可以关联一个非负权重 w_{ij}，不过通常研究的是无权情形，即所有边的 $w_{ij} = 1$ 均为 1。割 $C = (S, \bar{S})$ 表示将顶点集 V 划分为两个互不相交的子集 S 和 $\bar{S} = V \setminus S$。要最大化的目标函数是割值，定义为 $\text{CutVal}(C) = \sum_{(i,j)\in E: i\in S, j\in\bar{S}} w_{ij}$，用于衡量跨越划分边界的边的总权重。最大割问题旨在从顶点集的所有可能二分划分中，找出使该目标函数最大化的划分 $(S^*, \bar{S}^*)$。

由于在大规模最大割实例中寻找精确解具有固有的计算复杂度，通常使用近似比这一指标来评估算法。该指标用于量化解的质量相对于最优值的水平。具体而言，对于一个最优割值为 $\text{CutVal}_{\text{OPT}}$ 的最大割实例，以及一个产生割值 $\text{CutVal}_{\text{ALG}}$的算法，近似比定义为 $\text{AR} = \text{CutVal}_{\text{ALG}}/\text{CutVal}_{\text{OPT}}$，其中 $\text{AR} \in [0, 1]$数值越高，表示性能越好。例如，考虑一个最优最大割值为 10 的图，若某个优化算法找到的割值为 9，则相应的近似比为 $\text{AR} = 9/10 = 0.9$（即 90%）。这一指标使我们能够有意义地比较不同问题实例和不同规模下的算法性能，尤其是在计算上无法确定精确最优解时。Goemans-Williamson（GW）算法是最大割领域著名的经典算法 [16]，它利用半定规划技术，可保证近似比至少达到 0.878。

此外，作为一个无约束离散优化问题，最大割可以自然地编码为 QUBO 表述，或等价地编码为伊辛模型。这一特性使最大割问题非常适用于量子优化框架，例如量子近似优化算法。

2.2 量子近似优化算法

量子近似优化算法 [13] 是最具代表性的量子启发式算法，可用于在近期量子设备上求解组合优化问题，尤其是那些采用 QUBO 问题形式表述的问题。量子近似优化算法 [7, 40], 属于更广泛的变分量子算法类别，它们利用量子-经典混合计算范式，发挥量子叠加与经典优化技术各自的优势。该算法通过迭代框架运行，由经典优化器系统地调整参数化量子电路的参数，以最大化特定问题代价函数的期望值。这种变分方法使量子近似优化算法特别适合在含噪中等规模量子（NISQ）设备上实现；这类量子处理器的特点是量子比特数量有限，且易受噪声影响（进而影响门保真度）。得益于量子-经典混合计算框架 [32, 39, 45]，量子近似优化算法能够容忍此类硬件限制，仍有望实现量子优势。

量子近似优化算法的数学基础建立在量子绝热定理之上，并通过离散量子门对绝热量子计算进行近似。对于编码为一个含 n 个量子比特的成本哈密顿量 H_C 的组合优化问题，量子近似优化算法通过交替应用两个参数化酉算符来构造试探量子态 $|\psi(\gamma, \beta)\rangle$：成本算符 $U_C(\gamma) = e^{-i\gamma H_C}$ 和混合算符$U_M(\beta) = e^{-i\beta H_M}$，其中 H_M 通常选为横场哈密顿量。对于 p 层，试探态表示为 $|\psi(\gamma, \beta)\rangle = U_M(\beta_p)U_C(\gamma_p)\cdots U_M(\beta_1)U_C(\gamma_1)|+\rangle^{\otimes n}$，其中 $|+\rangle^{\otimes n}$ 为均匀叠加初态。优化目标是最大化期望值 $\langle\psi(\gamma, \beta)|H_C|\psi(\gamma, \beta)\rangle$，这需要使用梯度下降或进化策略等经典算法进行迭代参数优化。

量子近似优化算法的实际实现涉及多个影响其性能和可扩展性的关键因素 [6,17]。试探电路深度由量子近似优化算法层数决定 p，在解的质量与

量子资源需求之间引入了权衡：更深的电路通常能获得更高的近似比，但需要更长的相干时间和更高的门保真度。参数初始化策略也会显著影响解的质量以及所需的优化迭代次数 [35, 45]。

2.3 量子近似优化算法的经典模拟

鉴于当前硬件在量子比特数量、门保真度和可访问性方面的限制，在经典计算机上进行量子电路模拟 [44] 是量子算法开发的重要工具。混合量子-经典算法（如量子近似优化算法）通常会在经典系统上进行评估，以便在现有量子设备无法处理的问题实例上开展原型设计、理论验证和性能基准测试。模拟通用量子电路的计算成本会随量子比特数量呈指数级增长，从而构成根本性的规模限制。状态向量模拟 [5, 21, 38] 会保留完整的量子态，并通过矩阵-向量乘法应用量子门，从而得出精确结果，但对于 n 个量子比特，需要 $O(2^n)$内存和计算资源（例如，采用双精度时，$n = 32$ 需要 64 GiB），因此仅适用于规模相对较小的系统。张量网络方法 [33, 41]，（例如矩阵乘积态及相关分解方法）利用纠缠有限这一特点来降低复杂度。当纠缠受到限制时，这种方法可以模拟更大的量子系统，但可能会牺牲准确性。密度矩阵模拟 [27] 将状态向量方法扩展至纳入量子噪声和退相干，从而更逼真地模拟含噪声中等规模量子设备，但会带来额外的计算开销。

一些已有研究探索了 QAOA 电路的模拟，尤其关注针对 QAOA 电路独特结构优化模拟过程。例如，Lin 等人 [30] 提出了针对成本层的优化，并利用多个 GPU 加速 QAOA 电路的模拟。FOR-QAOA [8] 是一个优化 QAOA 电路混合器层的框架，并支持多节点并行化以加快模拟过程。这些研究表明，尽管 QAOA 电路能够在经典硬件上高效模拟，但量子态表示的指数级扩展仍是一项根本性挑战，尤其是在处理规模较大的问题实例时。

2.4 用于求解最大割问题的分治 QAOA

量子计算中的分治范式可追溯至 Dunjko 等人 [10], ，他们提出了将大型量子优化问题分解为较小子问题并独立求解的思想。此后，Ge 等人 [15]进一步拓展了这一概念。在量子近似优化算法的背景下，分治方法旨在解决大规模最大割问题实现中的可扩展性限制。具体而言，分治范式将原始实例划分为较小且易于处理的子问题，独立求解这些子问题或仅考虑有限的相互依赖关系，然后合并各子问题的解，以重构原问题的结果。这一策略缓解了量子优化算法的指数级扩展瓶颈，使得在当前量子硬件或经典模拟器上求解包含数百乃至数千个顶点的最大割实例成为可能。其有效性取决于所采用的分解方案、子问题求解方法，以及用于保持解的质量的整合流程。

分治 *QAOA* 示例。图中展示了一个使用分治 QAOA 方法求解最大割问题的示例。1。该方法包含四个阶段。

(1) 图划分。原始最大割图被划分为更小的子图。一些顶点在多个子图中共享，之后会用于解重构阶段。

(2) *QAOA* 执行。量子近似优化算法分别应用于每个子图，以获得近似解。每个子图都用一个 比特串表示，用于编码其划分方案，映射方式如图2所示。例如，子图 1 得到比特串 `110`，表示顶点 v_1 和 v_2 属于集合 S，而 v_3 属于$\bar{S}$。

(3) 合并 . 通过拼接比特串来组合子图解
此步骤要求共享顶点的赋值保持一致。例如，在图 1中，只有当两个子图都将v_3 赋值到同一集合时，才能将它们合并。由于子图 1 和 2 的比特串中，顶点 v_3 的取值为`0`，因此可以将这两个子图合并，得到比特串`11010`。

(4) 结果根据原始最大割目标函数评估合并后的比特串，以最大化 S 和 $\bar{S}$之间的边数。如图 1右上方所示，得到的解可能并非最优解（割值为 3）。因此，使用近似比来评估解的质量，其定义为合并后比特串的割值与原始图的最优割值之比（例如，图中为 3/4）。

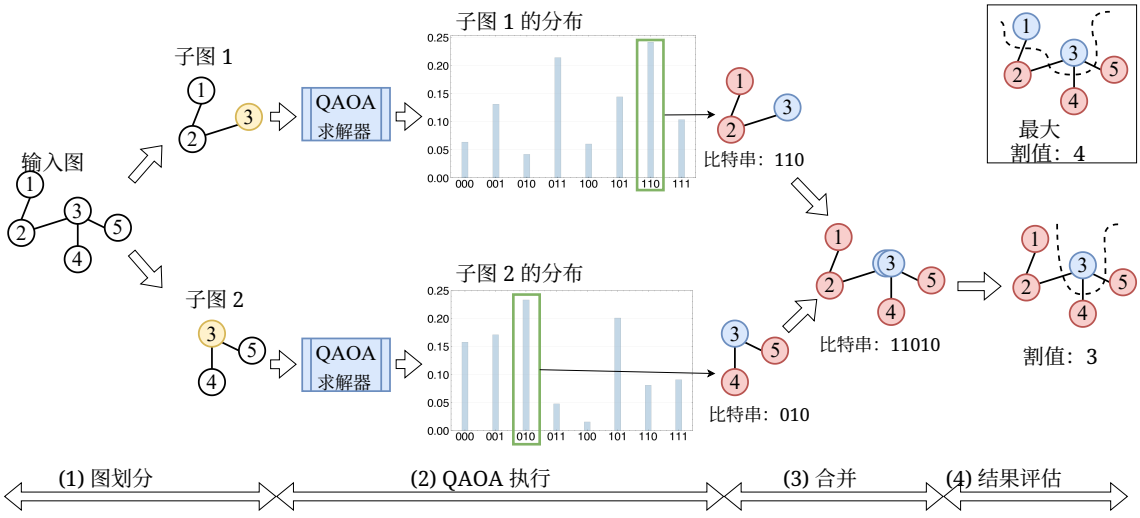


图 1. 用分治 QAOA 求解最大割问题的示例。

现有图划分策略。 图划分是最大割等组合优化分治法中的关键预处理步骤。其目标是将图拆分为较小的子图，同时尽量减少分区间边。减少这些边可以降低子问题复杂度、支持并行执行，并保留有助于获得高质量解的结构属性。传统划分算法的计算复杂度往往较高，随顶点数量呈二次或更高阶增长。对于大型实例，这可能造成显著的预处理瓶颈，尤其是在依赖复杂划分来获得较高近似比的框架中。在量子优化中，特别是在基于分治法的框架中，划分能够使大规模最大割实例适应硬件限制，例如量子比特容量。虽然将图拆分为较小的子图可以降低计算需求，但移除分区间边可能会降低近似比。这就形成了计算效率与解的质量之间的权衡。现有策略在权衡取舍方面各不相同。

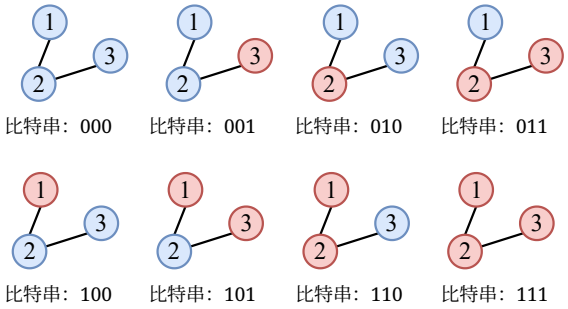


图 2. 不同划分下的图及其对应的比特串表示。

DC-QAOA [29] 采用大图划分（LGP）算法，通过平衡子图来获得较高的近似比，但其较长的运行时间和连通性约束限制了可扩展性。量子近似优化算法^2 [46] 采用随机划分以加快执行速度。该方法通过先进的合并策略弥补近似比可能出现的损失，但会带来额外的合并开销。Coupling QAOA [31] 采用二进制分解，通过哈密顿量中的耦合项保留连通性，但仅限于二元划分，且无法有效扩展到更大的图。

现有分治 *QAOA* 方法的局限性。现有的分治 QAOA 变体，例如 DC-QAOA [29], 量子近似优化算法^2 [46], 和 Coupling QAOA [31], , 主要关注解的质量，但存在显著局限，降低了其实用性和可扩展性。这些局限可分为三大类。首先，执行时间瓶颈 源于计算开销巨大的预处理，尤其是时间复杂度相对于顶点数为二次或更高的图划分算法。例如，DC-QAOA 依赖复杂的划分方法，其复杂度为 $O(|V|^2)$ 或更高，这可能会主导大型实例的总运行时间，并抵消潜在的量子加速。由于 DC-QAOA 的实现未公开，我们的评估基于根据其描述重新实现的版本。量子近似优化算法^2 和 Coupling QAOA 还需对所有子问题组合进行穷举搜索，导致运行时间和内存使用量呈指数级增长。其次，参数优化 通常缺乏系统性，甚至完全没有，因而造成收敛问题和解的质量不稳定。缺少系统的参数初始化和调优时，性能会因问题实例而显著变化，限制了其在实际部署中的可靠性。第三，可扩展性约束 限制了问题分解的程度以及可处理的最大问题规模。Coupling QAOA 仅支持二元分解，因此限制了问题规模的缩减。DC-QAOA 在稠密图上会出现性能退化，因为这类图难以划分。量子近似优化算法^2 缺乏对子问题执行进行高效并行化的方法，因而限制了可扩展性的提升。Chuang 等人 [9] 也指出了这些挑战，并强调需要采用更全面、设计更系统的方法。然而，他们的评估仅涵盖小规模最大割实例（最多 26 个顶点），不足以评估分治量子近似优化算法方法的可扩展性。

2.5 研究动机

如第 2.4 节所述，当前的分治量子近似优化算法方法更注重解的质量，而非执行效率。这种设计选择限制了它们在大型或复杂问题实例中的效用。对于顶点数超过 100 的问题，由于执行时间存在显著差异，评估变得颇具挑战。根据我们的实验，对于一个包含 400 个顶点的中等规模高密度图，量子近似优化算法 QAOA^{2}需要约 4.7 小时才能计算出解，

而对于一个包含 16,000 个顶点的大规模高密度图，预计需要约 13.6 天。更多结果见第 4 节。关键观察是，先前的研究开发了新颖的图划分算法，以降低子图的计算负担（例如，DC-QAOA 中的 LGP 和 Coupling QAOA 中的二进制分解），并采用复杂的合并策略来减轻近似比的下降，但代价是执行时间增加。不幸的是，这些方法未能充分利用高性能计算硬件，例如多核处理器和多个 GPU；这些硬件可用于并行执行子问题求解和解重构。因此，它们在大型最大割问题上的可扩展性和效率仍然有限。

认识到现有方法的局限性，并考虑到高性能计算硬件的存在，我们提出了*ParaQAOA*，这是一个面向最大割问题的框架，融合了高效的图划分算法、并行化执行流程、系统化的参数化设计以及性能评估指标。这些创新的特点如下。首先，我们开发了一种线性时间图划分算法。通过利用结构特性，该算法将分解复杂度从二次或更高降低至与顶点数呈线性关系，从而消除了预处理瓶颈，使大型图的高效划分成为可能。其次，我们引入了完全并行化的执行流程。该并行执行流水线涵盖子问题求解和解重构，能够有效利用现代并行架构，缩短整体运行时间。第三，我们提出了系统化的参数化设计。该设计将硬件平台规格纳入参数配置，并留出可调参数供用户管理执行效率与解的质量之间的权衡。这些参数将在下一节介绍该框架时引入，并在第4.2节给出如何设置这些参数的具体示例。第四，我们引入了统一的评估指标。该指标综合考量解的质量和执行效率，从而能够在不同算法和配置之间进行一致的比较，并为权衡决策提供依据。这对于大规模问题尤为重要，因为执行时间的显著差异可能会增加评估算法有效性的难度。总之，这些创新使 ParaQAOA 能够以更强的可扩展性和更高的计算效率解决大规模最大割问题。

3 ParaQAOA

本节介绍我们用于求解大规模最大割问题的方法。所提出的 ParaQAOA 框架概述见第 3.1 节。随后，我们将详细介绍 ParaQAOA 框架的三个关键组成部分：第 3.2 节的图划分策略、第 3.3 节的并行化 QAOA 执行，以及第 3.4 节的层级感知并行合并过程。最后，我们将在第 3.5 节介绍所提出的性能效率指数，以系统地评估用于处理大规模最大割问题的方法所涉及的性能权衡。

3.1 框架概述

所提出的 ParaQAOA 框架采用分层且高度并行的架构，借助量子计算能力高效求解大规模最大割问题。该架构通过将大规模图分解为易于处理的子图、并行处理以及有序汇总结果，应对其计算复杂度。该框架融入了多项关键优化，以提高计算效率，并改善求解最大割问题时的解的质量。如图 3 所示，这些优化分为三个连续阶段，逐步将输入图转化为最终输出解。此外，还设有性能

评估阶段，用于生成所提出的性能效率指数，为评估所得最大割解的效率提供量化指标。该框架的关键组成部分如下所述。

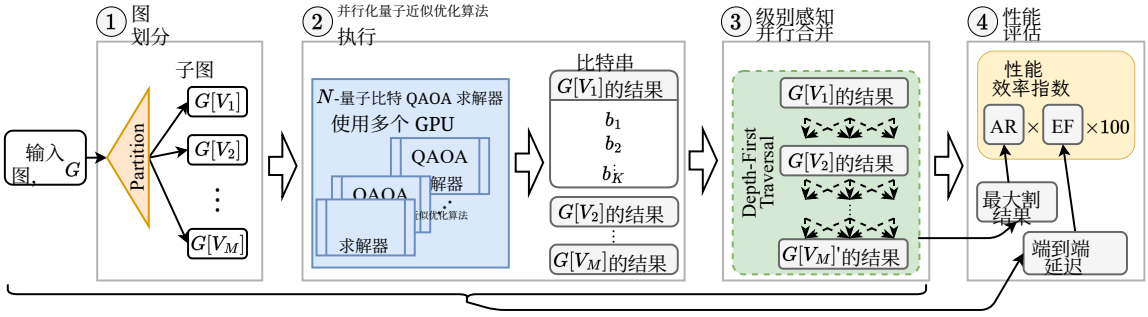


图3. ParaQAOA 框架概览。

①图**划分**阶段将输入图 G 分解为多个较小的子图$\{G[V_i]\}_{i=1}^{M}$，其中每对相邻子图最多共享一个节点。此分区确保问题规模保持在可控范围内，因为 QAOA 求解器通常会受到可处理量子比特数量的限制。尽管有多种分区策略可供选择，本框架采用保连通性的分区算法，该算法能够高效地将图划分为预先指定数量且规模大致相等的子图，同时保留关键的连通性信息。这种方法能够在可用量子资源间实现有效的负载均衡，从而促进大规模图的可扩展并行处理。

②并行化**QAOA 执行**阶段将每个子图分配给一个 QAOA 求解器，并使其在多个计算单元（如 GPU）上并发运行，从而高效探索各子图的解空间。子图$\{G[V_i]\}_{i=1}^{M}$ 将被分批送入多个 QAOA 求解器。对于每个子图，QAOA 求解器会输出一组排名靠前的K比特串$\{b_i\}_{i=1}^{K}$，其中 $b_i \in \{0,\ 1\}^{|V_i|}$按对应概率 π_i排序；也就是说，每个 b_i 都是长度为 $|V_i|$ 的二进制串，表示顶点集 V_i的二分划分。此外，我们提出了一种选择性分布探索策略，可灵活调整比特串的数量 K，以平衡计算资源与解的质量。此阶段利用并行执行高效探索各子图的解空间，并可灵活调整比特串的数量，以平衡计算资源与解的质量。

③层级感知并行合并 阶段通过对比特串集合的笛卡尔积空间进行深度优先遍历，重建所有子图的比特串结果 $\{b_i\}_{i=1}^{K}$，以枚举可能的全局解。对于每个重建的全局比特串，都会重新计算其潜在的最大割值，计算时考虑原始输入图中的所有边 G。随后，算法选择产生最大割值的比特串作为最终输出。此外，该方法还具有动态灵活性，可通过调整候选池大小，在计算效率和解的质量之间取得平衡。

④ **性能评估**阶段将系统评估大规模最大割求解的有效性，以平衡执行时间和近似比。我们引入一种新的性能指标，性能效率指数，这是一种复合指标，旨在量化解的质量与计算效率之间的基本权衡，以便对不同的最大割问题求解方法进行基准测试。

The 以下小节将详细介绍我们在 ParaQAOA 框架中对各个组件的实现 .

3.2 图划分

图划分阶段是 DC-QAOA 类框架的关键组成部分。它将输入图拆解为更小的子图，从而能够高效并行处理大规模最大割问题。我们通过提出保持连通性的分区 [29] 算法，改进了 DC-QAOA^2 [46] 和量子近似优化算法算法。该算法将输入图 G划分为 M子图$\{G[V_1],\ G[V_2],\ \ldots\ldots,\ G[V_M]\}$。子图 M的数量根据可用 QAOA 求解器的容量和底层硬件资源确定。最重要的是，每个子图都保留了必要的连通性信息，如图4所示，同时将问题规模缩减至适合 N量子位 QAOA 求解器的限制范围内。

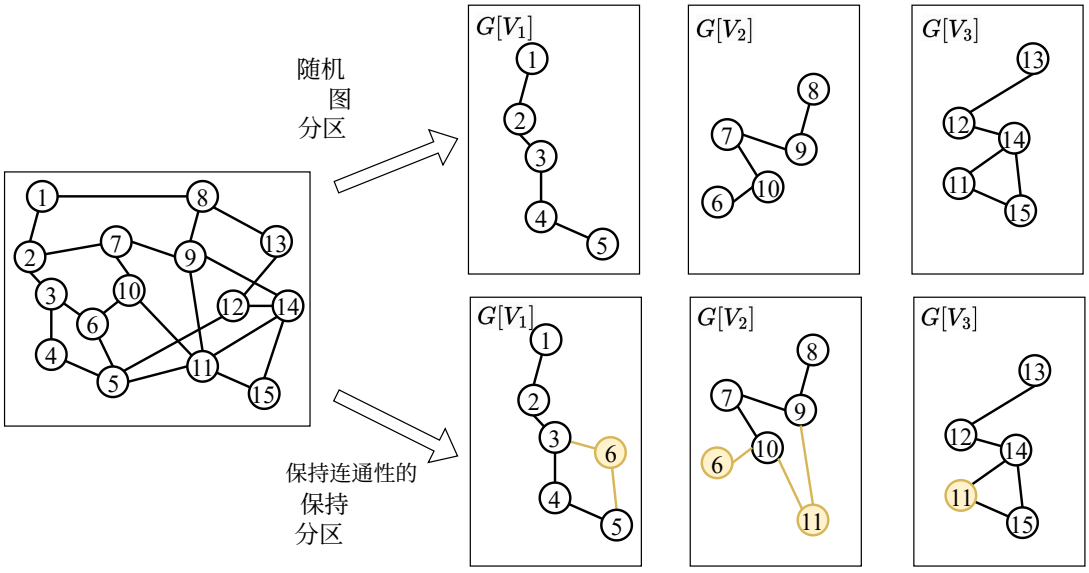


图 4. 随机分区算法与所提出的保持连通性的分区算法对比。

3.2.1 划分约束。给定一个含有 $|V|$ 个顶点的输入图 $G=(V,\ E)$，我们旨在将其划分为 M 个子图，同时确保后续量子近似优化算法处理在计算上可行。划分函数P：$G \rightarrow \{G[V_i]\}_{i=1}^{M}$ 满足以下约束。

(1) 相邻子图连通性：对于所有 $i \in \{1, 2, \ldots, M-1\}$，$|V_i \cap V_{i+1}| = 1$，这意味着子图 i 和子图 $i+1$ 恰好共享一个节点。此属性会形成一条由连通子图构成的顺序链，以保留连通性信息。(2) QAOA 兼容性：对于每个子图 $G[V_i]$，$|V_i| \leq N$，其中 N 表示 QAOA 求解器可用的量子比特数量。此属性确保每个子图都能由 QAOA 求解器处理，而不会超出求解器的量子比特容量。

(3) 规模均衡：对于每个子图 $G[V_i]$，$|V_i| \le \lceil |V|/M \rceil$，确保各分区中的顶点分布均衡。一种更简单的方法是对每个子图设置 $|V_i| = \lfloor |V|/M \rfloor$，这样即可满足约束。将图划分为规模相近的子图，可确保计算负载均衡。

3.2.2 分区算法。 我们采用高效的保持连通性的分区方法，在保持解的质量可接受的同时，将计算复杂度控制在 $O(|V|+|E|)$。该算法以图 $G=(V, E)$ 和期望的子图数量 M 为输入，并输出 M 个子图（$G[V_1], G[V_2], \ldots\ldots, G[V_M]$）。核心方法将 G 的顶点划分为 M 个大小近似相等的组，然后为每组生成一个导出子图，并保留同一组内顶点之间的所有原始连接。

算法1 详细说明了我们高效图划分方法的实现。首先，将基础分区大小计算为 $s = \lfloor |V|/M \rfloor - 1$，以确保顶点在各分区间均衡分布。减去1是为了计入稍后要添加的共享节点。随后，算法从1 迭代至 M，在执行图划分的同时，跟踪分配到这 M 个分区中各自的顶点，并采用节点共享技术，在子图之间保留连通性信息。最后一个分区会容纳所有剩余顶点，以处理 $|V|$ 不能被 M整除的情况。对于每个分区，使用 $G[V_i]$ 调用 GetSubgraph生成子图，该函数收集所有$(u,v) \in E$，其中两个端点都属于同一个顶点集 V_i。最后，返回分区子图列表$\{G[V_i]\}_{i=1}^{M}$。

算法 1： 保持连通性的分区：一种保留连通性的高效图划分方法
通过共享节点获取相邻子图的连通性信息。

输入： G，输入图，
M，子图数量
输出： $\{G[V_1], G[V_2], \ldots\ldots, G[V_M]\}$，分区子图

```
V ← V（G）；                                   ▷ G中的顶点集合，索引范围从0 到 |V| − 1
s ← ⌊|V|/M⌋ − 1;                              ▷ 负载均衡划分大小；“−1”用于后续的节点共享
个子图 ← ∅;
对于 i ← 1 到 M 执行
    起始_索引 ← (i−1) × s;
    如果 i = M 则
        结束_索引 ← |V|;
    else
        结束_索引 ← 起始_索引 + s + 1;             ▷ ‘+1’ 表示节点共享
    V_i ← {v ∈ V | 起始_索引 ≤ 索引(v) < 结束_索引};
    G[V_i] ← GetSubgraph(G, V_i);
    子图 ← 子图 ∪ {G[V_i]};
返回子图;
功能 GetSubgraph(G, V_i):
    初始化 G[V_i] = (V_i, E_i)，其中 E_i = ∅;
    对每个 (u,v) ∈ E 执行
        如果 u ∈ V_i 且 v ∈ V_i，则
            E_i ← E_i ∪ {(u, v)};
    返回 G[V_i];
```

与复杂的划分方案相比，这种方法看起来或许较为简单，但实验结果表明，它在保持解的质量的同时，能够快速处理大规模实例。该方法尤其适用于边分布近似均匀的稠密图。这一方法的主要优势包括：1）适用于大规模实例的线性时间预处理；2）不受图结构影响的确定性执行时间；3）内存使用量极低；4）出色的可扩展性。这些特点使我们的划分方法特别适合在 ParaQAOA 框架中求解大规模最大割问题，因为在该框架中，计算效率至关重要。

3.3 并行化 QAOA 执行

并行化 QAOA 执行阶段是我们框架的主要计算阶段，在此阶段中，所有分区子图上都会同时执行量子计算。如图 5 所示，该阶段接收 M由图划分阶段生成的子图，并在每个子图上独立执行 QAOA。对于每个子图 $G[V_i]$，执行会生成一组排名靠前的K比特串 $\{b_1,\ b_2,\ \ldots\ldots,\ b_K\}$，其中每个比特串都编码一个顶点赋值（0 或 1），并根据其在 QAOA 输出分布中的高概率被选出。这些筛选后的比特串将作为后续合并阶段的候选解。

我们的框架采用一个包含 N 个量子比特的量子近似优化算法求解器池，其中 N 根据可用的量子处理单元（QPUs）或经典计算资源确定。我们利用多 *GPU* 计算架构 并行执行量子近似优化算法，每个子图分配给一个量子近似优化算法求解器。每个求解器都会执行量子电路，为其分配的子图计算解分布。此外，我们提供选择性分布探索策略，允许用户配置参数 K，并可对其进行调整，以期提高近似比（K 值越大）。也就是说，通过参数 K控制在计算资源与解的质量之间进行权衡的灵活性。

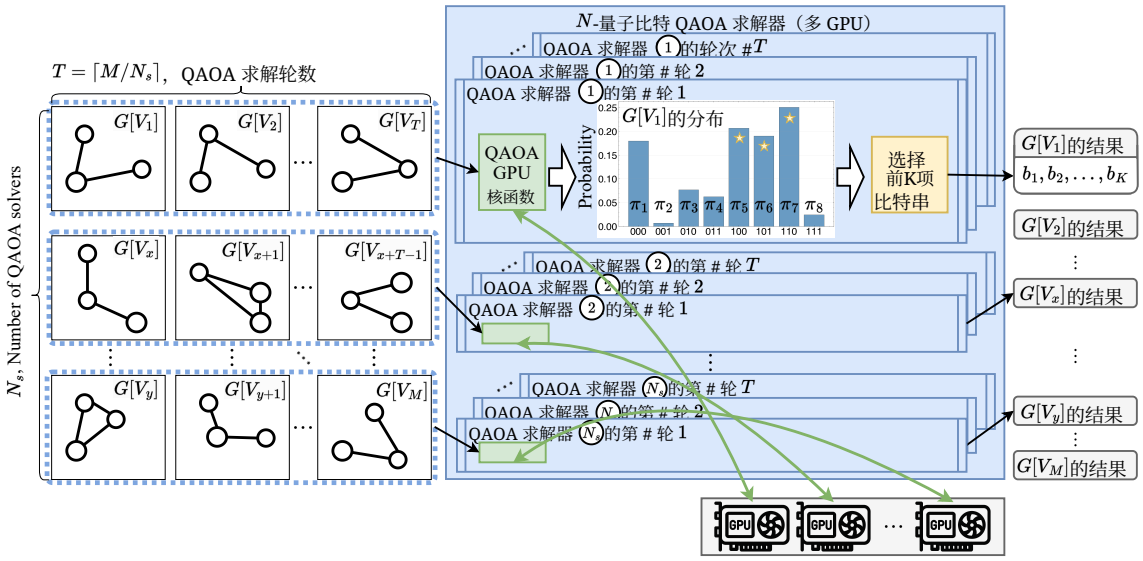


图 5. 所提出的并行化 QAOA 执行工作流与多 GPU 系统集成。

*3.3.1 多 GPU 计算架构。*针对多个子图模拟量子电路，需要采用有效的分布式计算方法。我们的多 GPU 架构通过动态资源分配和并行执行来应对这一挑战。如图 5中部所示，系统采用负载均衡策略，根据子图规模和当前 GPU 利用率，将子图分配到各个 GPU，从而确保资源得到最佳利用并尽可能减少空闲时间。该并行执行框架可同时处理多个量子电路，每个子图均独立计算。随着 GPU 数量增加，这种设计可实现接近线性的加速比；对于大型问题实例而言，这一点至关重要，因为顺序处理的速度会过慢。通过同步汇总结果，确保所有量子近似优化算法执行完成后再进入下一阶段。

在当前的原型实现中，我们使用 Lu 等人提供的 QAOA GPU 内核 [31], ，该内核利用 *Numba* [25] 包加速 NVIDIA GPU 上的 QAOA 执行。此内核会计算每个子图的完整量子态分布，其中每个比特串对应一个潜在解。它根据子图的哈密顿量高效地应用量子门，并借助 GPU 并行性输出由此得到的比特串概率。

如图 5 所示，给定 N_s 个可用的 QAOA 求解器和 M 个子图，执行分为 $T = \lceil M/N_s \rceil$ 轮进行。每一轮中，一个 QAOA GPU 内核处理一个子图，并输出 2^N 个比特串上的概率分布 $\{\pi_1, \pi_2, \ldots, \pi_{2^N}\}$，其中 N 为该子图中的量子比特数。例如，第一轮处理子图 $G[V_1]$，生成分布 $\{\pi_{(1)1}, \pi_2^{(1)}, \ldots, \pi_{(1)8}\}$。对所有子图重复这一过程，每个 QAOA 求解器都会以比特串及其对应概率的形式输出最大割问题的候选解。

3.3.2 选择性分布探索策略。 QAOA 求解器为每个子图生成 2^N 个比特串，其中 N 是与子图规模相对应的量子比特数。为了降低计算复杂度并聚焦于高质量候选解，我们采用一种选择性分布探索策略，根据测量得到的概率筛选比特串。因此，每次 QAOA 执行都会为第 i 个子图生成一组 K 个比特串 $\{b_{(i)1}, b_2^{(i)}, \ldots, b_{(i)K}\}$，代表量子优化所识别出的最有希望的解。

每个比特串 $b_{(i)j} \in \{0,1\}^{|V_i|}$ 都编码了子图 $G[V_i]$ 的一种二元划分，其中 $|V_i|$ 是顶点数，且 $1 \le j \le K$。值 $b_j^{(i)}[v] = 0$ 将顶点 v 分配到一个分区，而值 $b_j^{(i)}[v] = 1$ 将其分配到另一个分区。尽管高概率比特串通常对应着更优的割，但该策略也通过纳入概率较低的候选解来保留多样性，从而考虑到全局最优解可能位于状态空间中概率较低的区域。

参数 K 控制 QAOA 执行后每个子图保留的候选解数量。较大的 K 会增加后续合并阶段所考虑解的多样性，可能提高近似比，但也会带来更高的计算开销。相反，较小的 K 会将选择范围限制在表现最佳的比特串中，从而降低复杂度，同时聚焦于概率最高的解。如图 5所示的简化示例中，当 $K = 3$时，ParaQAOA 框架会选取概率最高的 3 个比特串，例如 $\{b_{(1)1}, b_{(1)2}, b_3^{(1)}\} = \{100, 101, 110\}$，对应子图 $G[V_1]$。随后，这些比特串会作为候选结果存储起来，以供合并阶段使用。

这一可调参数可在计算效率与解的质量之间进行权衡，使用户能够根据不同的应用约束调整框架。尽管每个子图的理论解空间包含 $2^{|V_i|}$ 种配置，量子近似优化算法（QAOA）会自然地将概率质量集中在少量高质量解上，并借助量子叠加高效识别有前景的候选解。

3.4 层级感知并行合并

层级感知并行合并阶段负责根据并行量子近似优化算法（QAOA）的运行结果重建全局解。虽然这一阶段不会影响近似比，但它是计算和从大量潜在解中找出最大割的计算量最大的后处理步骤。要实现最佳性能，需要大量硬件资源。合并过程通过合并（串接）子图结果，重建问题的解，以找出使原始图全局割值最大化的配置。串接过程采用对由子图结果构成的笛卡尔积空间进行并行深度优先遍历合并的方式，其中每一级对应一个子图的解集，分支则代表穿越组合空间的潜在解路径。此外，为充分利用并行处理能力，合并阶段采用层级感知方案，以便利用额外的硬件资源高效执行合并过程并找出最大割结果。

3.4.1 并行深度优先遍历合并。合并阶段的工作流程如图6所示，其中以逐级方式展示多个子图的结果，并将其组合以形成全局候选解。每个级别对应一个子图，每个分支代表一条穿过组合空间的潜在解路径。此过程对由子图解集构成的解空间进行并行深度优先遍历，每个工作进程负责探索一个特定分支，该分支从比特串 $G[V_1]$开始。当遍历到达最后一级时（对应子图总数，如图6底部所示），会调用割值$(\cdot)$ 功能，计算拼接后比特串的割值。随后，算法会在发现更优解时更新当前最大割值。

合并阶段的候选解总空间定义为各子图候选比特串集的笛卡尔积。令 $B_i = \{b_{(i)1}, b_{(i)1}, b_2^{(i)}, b_{(i)2}, \ldots, b_{(i)j}, b_{(i)j}, \ldots, b_{(i)K}, b_{(i)K}\}$ 表示子图 i 的候选比特串集，其中 b 表示 b的按位取反，从而涵盖每个解对应的两种可能分组分配。笛卡尔积空间定义如下：

$$B_1 \times B_2 \times \cdots \times B_M = \left\{(b^{(1)}, b^{(2)}, \ldots, b^{(M)}) \mid b^{(i)} \in B_i \text{ for each } i = 1, 2, \ldots, M\right\}.$$

该笛卡尔积空间表示在合并阶段从每个子图中各选取一个比特串进行评估所形成的候选全局解总集。尽管每个 B_i 包含 $2K$ 个比特串，以涵盖原始表示及其反转表示，但由于分区过程中保留了连通性约束，因此每个级别只能选取一半 。具体而言，对于来自子图 i的任意比特串 $b_j^{(i)}$，下一级可能的拼接结果为 $b_{(i+1)j}$或其反转形式 $b_{(i+1)j}$。因此，候选解的总数为 $2K^M$，其中 K是 QAOA 求解器保留的顶部比特串数量， M是子图数量。

最后，算法通过识别最优全局配置 B^*来更新最大割值。形式上，给定 M个子图 $\{G[V_1],\ G[V_2]$、……、$G[V_M]\}$及其对应的 QAOA 解集 $\{B_1,\ B_2$、……、$B_M\}$，合并阶段旨在求得：

$$B^* = b^{(1)} \| b^{(2)} \| \cdots \| b^{(M)}, \text{where } b^{(i)} \in B_i \text{ for each } i = 1, 2, \ldots, M,$$

以最大化全局割值：

$$\text{Cut}(B^*) = \sum_{i=1}^{M} \text{Cut}(b^{(i)}) + \sum_{(u,v)\in E_{\text{inter}}} \mathbb{I}\left[B^*[u] \neq B^*[v]\right],$$

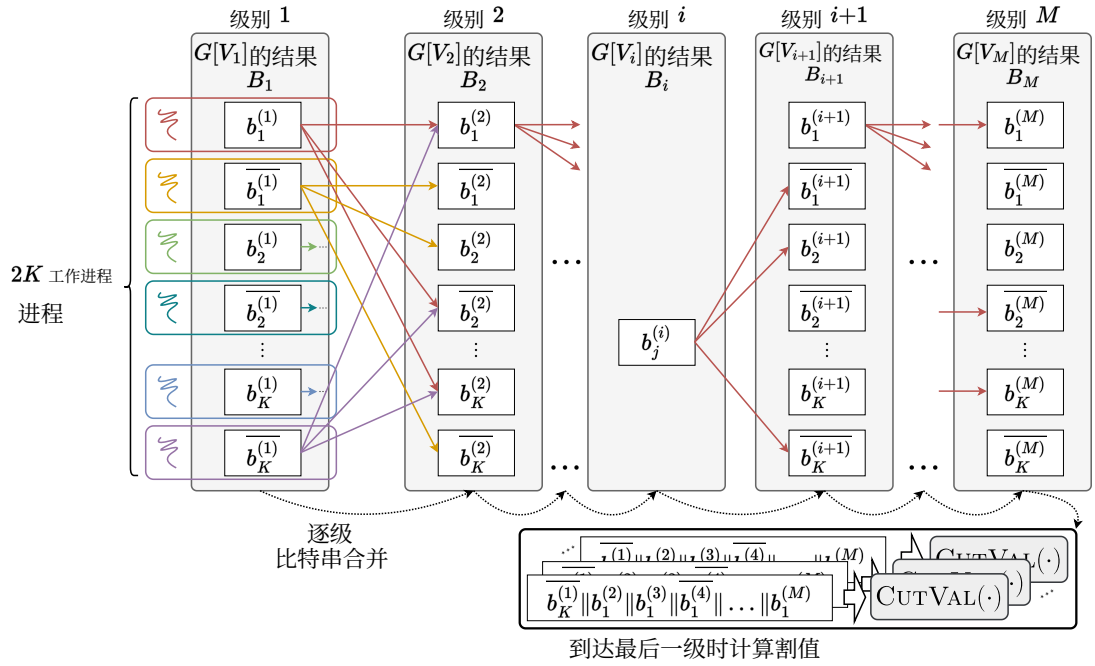


图 6. 并行深度优先遍历合并工作流程。多个工作进程从 $G[V_1]$ 的不同比特串结果出发，遍历由子图解形成的笛卡尔积空间，以探索解空间。

其中 E_{inter} 表示分区过程中舍弃的分区间边集合，$\mathbb{I}[\cdot]$ 是指示函数。指示函数 $\mathbb{I}[\cdot]$ 是一种数学函数：若括号内的条件成立，则返回 1；否则返回 0。在此情境下，它会逐一检查每条分区间边（u, v），判断顶点 u 和 v 在重构后的全局解 B^*中是否被分配到不同的分区（0 或 1）。如果 $B^*[u] \neq B^*[v]$，该边就计入割中，函数返回 1；否则返回 0。此机制确保总割值既计入各个 QAOA 求解器计算的子图内割，也计入分区阶段舍弃、但必须在最终全局解中考虑的分区间边。

3.4.2 层级感知方案。 为高效探索由子图结果的笛卡尔积形成的解空间，我们采用层级感知合并策略。此方法允许用户设置合并阶段的起始级别，从而在搜索解空间时实现更细粒度的并行性。图 7 展示了从第 2 级开始的层级感知合并过程。与图 6 中从第一级开始合并的情况相比，这种方式提高了并行性。该设计通过将工作进程数量从 $2K$ 增加到 $2K^2$，提升了硬件利用率，并在有更多计算资源可用时增强了可扩展性。一般而言，工作进程数量为 $2K^{\mathrm{L}}$，其中 L 是合并阶段的起始级别。这使得用户可以根据可用硬件资源和自定义参数灵活地进行并行处理。

算法2 详细说明了所提出的层级感知并行合并方法。该算法通过逐级遍历解空间，递归构建有效的全局配置，同时维护一条代表部分构建全局解的当前路径。算法采用上述并行且具备层级感知能力的合并技术，最后返回最大割值及相应的全局配置。算法分为以下几个阶段：

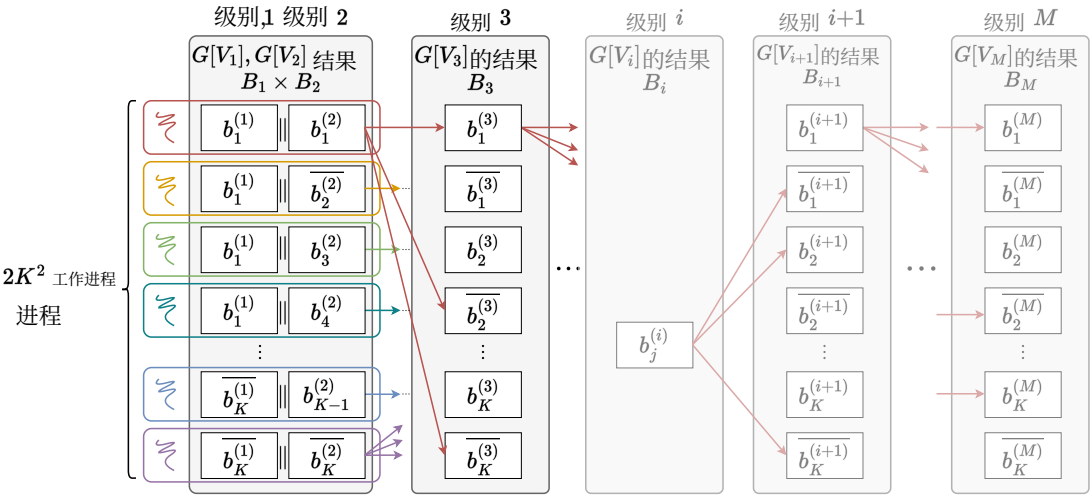


图 7. 层级感知并行深度优先遍历合并。从级别 2 开始，相比从级别 1 (图 6) 开始，通过先合并 $G[V_1]$ 和 $G[V_2]$ 的结果，提升了并行性。右侧内容变暗，表示后续进程与图 6所示相同。

多个不同阶段。首先，它初始化最大割值及相应的配置，用于跟踪遍历过程中找到的最佳解。随后，算法合并前 L 个子图的比特串结果，生成起始点，其中 L 是用户定义的合并阶段起始级别。此步骤会创建一组将在并行处理中扩展的初始路径。其次，算法启动多个并行进程，从每条起始路径开始，在级别 $L+1$ 执行深度优先遍历，并逐步推进，直至到达最终级别 M。每个进程都会将当前子图结果集中的兼容比特串追加到当前路径，以此探索解空间。当遍历到达最终级别 M 时，算法会计算割值；如果找到更优解，则更新最大割值。随后，算法等待所有进程完成。最后，返回最大割值 C_{max} 及其对应的全局配置 V_{max}。

3.5 性能评估指标

为了系统地评估不同方法在最大割问题上的整体性能，本文提出性能效率指数 作为统一指标，用于量化解的质量与计算效率之间的性能权衡。受能量延迟积 [26]，（最初由 Horowitz 提出 [19]，用于评估数字电路设计中的权衡）启发，性能效率指数针对量子优化领域进行了调整。在该领域，解的质量提升往往以执行时间增加为代价。性能效率指数的形式化定义如下。

$$\mathrm{PEI} = \mathrm{AR} \times \mathrm{EF} \times 100$$

该指数整合了两个关键组成部分：近似比（AR），用于衡量最大割解相对于最优值或基准值的质量；以及效率因子（EF），用于量化相对于基准运行时间的计算效率。通过结合这两个维度，PEI 提供了一种标准化且易于理解的指标

算法 2： 层级感知并行合并

输入： $\{B_1, B_2, \ldots\ldots, B_M\}$，子图结果列表，
L，层级感知合并的合并阶段起始级别
输出： (V_{max}, C_{max})，最大割值及其对应的割配置

初始化 $V_{max} \leftarrow 0$，$C_{max} \leftarrow$ 空值；
起始_路径 $\leftarrow$ 合并 $B_1, \ldots\ldots, B_L$； ▷ 使用 L 合并比特串并创建起始点
/* 从级别 L 开始的并行深度优先遍历 */
对每个起始_路径 **执行**
  生成进程：深度优先遍历($L+1$，路径);
功能 深度优先遍历(级别, 当前_路径)**：**
  如果级别 $= M$ **则**
    $V_{cut} \leftarrow$割值(current_path);
    if $V_{cut} > V_{max}$ **then**
      $V_{max} \leftarrow V_{cut}$； $C_{max} \leftarrow$ 当前_路径；
    返回;
  对每个 兼容的比特串 b 在 B_{level} **执行**
    深度优先遍历(级别 $+1$, $current\_path \parallel b$); ▷ 比特串拼接
等待所有进程完成；**返回** (V_{max}, C_{max});

用于评估不同参数设置和基线方法下的算法性能，从而通过单一分数实现多目标评估。

$$\text{Approximation Ratio: AR} = \frac{\text{CutVal}_{\text{ALG}}}{\text{CutVal}_{\text{OPT}}},$$

其中，$\text{CutVal}_{\text{ALG}}$ 表示待评估算法获得的割值，而 $\text{CutVal}_{\text{OPT}}$ 表示原始图的最优割值或现有方法得到的已知最佳值。近似比用于量化所得解与最优解的接近程度，取值范围为 0 到 1，其中 AR $= 1$ 表示最优解。在本研究中，对于包含 100 至 400 个顶点的问题，最佳割值取自 GW方法 [16],，该方法保证在多项式时间内获得至少为 0.878 的近似比。对于更大规模的实例，则采用最先进经典算法得到的已知最佳解作为基线，以评估解的质量。

$$\text{Efficiency Factor: EF} = \frac{1}{1 + e^{\alpha \cdot (T_{\text{ALG}} - T_{\text{Base}})}},$$

其中 T_{ALG} 是待评估方法的执行时间，T_{Base} 是基线执行时间，而 α 是一个用于控制效率比对时间差异敏感度的缩放参数 e。效率比采用 S形函数，在不同性能区间之间实现平滑过渡，并使输出值保持有界。如图8所示，当 $T_{\text{ALG}} = T_{\text{Base}}$时，效率比为 0.5，表示性能持平。趋近于 1.0 的值表示计算加速显著（当 $T_{\text{ALG}} \ll T_{\text{Base}}$时），而趋近于 0.0 的值表示性能严重退化（当 $T_{\text{ALG}} \gg T_{\text{Base}}$）。

与线性归一化或对数归一化相比，Sigmoid 形式具有多项优势：它能够自然地处理极端时序变化，且不会引发数值不稳定性；其有界范围也便于直观理解，

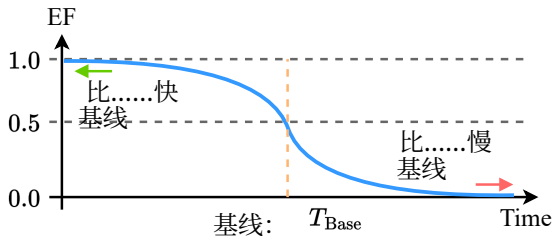


图 8. 效率因子取值范围的可视化。

并且无论时序差异的幅度如何，都能确保其对整体 PEI 得分的贡献保持均衡。在本研究中，我们将缩放参数设为较小的值，例如 $\alpha = 0.001$，以确保平滑过渡，并防止极端时序变化主导效率分量。

4 评估

本节介绍所提出的 ParaQAOA 框架的实验评估。实验设置和计算基础设施将在第 4.1 节中介绍。我们将在第 4.2 节中通过配置框架参数，展示该框架能够在解的质量与计算效率之间实现权衡。第 4.3 节比较 ParaQAOA 与当前最先进的基于 QAOA 的最大割求解器在中小规模图（少于 400 个顶点）上的性能，重点关注近似比和执行时间。第 4.4 节介绍大规模实例（超过 1,000 个顶点）上的可扩展性结果。最后，第 4.5 节使用所提出的性能效率指数评估性能效率，该指数刻画了解的质量与计算成本之间的权衡。

4.1 实验设置

所有实验均在本地高性能计算系统上进行，具体规格见表 1。需要注意的是，尽管主板支持 PCIe 5.0，但由于 NVIDIA RTX 4090 GPU 的硬件限制，其运行速度为 PCIe 4.0。此外，为确保公平比较，经典的 Goemans-Williamson 近似算法（GW） [16]（由 [31]实现）、Coupling QAOA（CQ） [31][1] 和 QAOA-in-QAOA（QAOA^2） [46] [2] 均在相同的硬件平台上下载并进行评估。

我们在各种 Erdős-Rényi 随机图 [12]配置下评估性能，涵盖小规模（20–26 个顶点）、中规模（100–400 个顶点）和大规模（1,000–16,000 个顶点）实例，边概率设为 0.1、0.3、0.5 和 0.8。使用 NetworkX 库中的 gen_erdos_renyi_graph 功能生成 10 个测试图 [18]。，每个图使用不同的随机种子[3] ，以确保所有实验中的随机化一致且可复现。对于中规模和大规模实例，使用单个固定种子来控制实验的总体运行时间。由于固有限制，并非所有框架都适用于所有图规模。在小规模评估中，我们报告 GW、CQ 和量子近似优化算法^{2}的结果。对于中规模图（最多 400 个顶点），仅评估量子近似优化算法^2 和 ParaQAOA，并以 GW 作为基线。不包括 CQ。在大规模场景中（最多 16,000 个顶点），

[1] https://github.com/LucidaLu/QAOA-with-fewer-qubits/tree/30a6a3fe24fe664281e17a3723573d1abf0b06df。 [2] https://github.com/ZeddTheGoat/QAOA_中的_QAQA/tree/7704cfd2c2cbfac58a11fbf4f8beebb1efb9c04c。 [3]对于所有图生成配置，均使用从 0 到 9 的整数种子。

表 1. 实验计算基础设施的具体 配置。

组件	规格
CPU	AMD Ryzen Threadripper 7960X（24 核）
GPU	2× 张 NVIDIA RTX 4090，每张配备 24 GB GDDR6X 显存
系统内存	256 GB（8× 32 GB DDR5 4800 MHz）
主板	TRX50 AERO D，配备两个 PCIe 5.0 x16 插槽
操作系统	Ubuntu 20.04 LTS
软件	CUDA 12.5，Python 3.12
Python 包	NetworkX 3.4.2、Numpy 2.0.2、Numba 0.60.0

我们比较量子近似优化算法[2] 和 ParaQAOA；然而，量子近似优化算法[2] 求解包含 4,000 个顶点的图需要超过 9 小时，因此更大规模实例的结果是外推得出的。

4.2 用于管理质量-效率权衡的参数配置

我们的框架涉及若干关键参数：QAOA 求解器的数量（N_s）、每个求解器的量子比特数（N）、分区子图的数量（M）、QAOA 求解轮次（T）、候选解保留参数（K），以及合并过程中的起始级别（L）。这些参数共同影响解的质量与计算效率之间的权衡。本节评估它们对解的质量和执行时间的影响。我们将参数分为三类：依赖硬件的参数、依赖输入的参数和可调参数。我们说明这些参数的配置，并展示实验结果，揭示可调参数如何影响求解最大割问题的性能。为评估可调参数的有效性，我们分析了在不同边概率下，求解包含 200 个和 600 个顶点的图时，可调参数对所得割值和执行时间的影响。

依赖硬件的参数。依赖硬件的参数包括 N_s 和 N，它们由可用计算资源决定。根据硬件规格（表1），我们在每个 GPU 上最多部署 12 个并发 QAOA 求解器实例，总计部署 $N_s = 24$ 个求解器。此配置可最大限度地提高物理 CPU 核心利用率，同时避免系统过载。每个求解器最多分配 $N = 26$ 个量子比特，使其能够处理最多包含 26 个顶点的子图，同时占用不到 1 GB 的 GPU 内存。其余内存则留作执行期间存储中间数据。

依赖输入的参数。依赖输入的参数包括分区子图的数量（M）和量子近似优化算法求解轮次（T）。M 的值由输入图规模（$|V|$）和分配给每个求解器的量子比特数量（N）决定。具体而言，M 设为 $|V|/(N-1)$，其中减去的部分用于计入相邻子图之间的一个共享节点。求解轮数 T 设为 M/N_s，以确保每个量子近似优化算法求解器承担均衡的工作负载，并且所有求解器在整个执行过程中都得到充分利用。

可调参数。可调参数 K 和 L 可进一步控制解的质量与计算效率之间的权衡。参数 K 指定在并行 QAOA 执行阶段保留的高概率比特串数量（前 K 个），从而影响候选解的多样性。参数 L 定义合并过程的起始级别，影响子图解开始合并的早晚。可根据特定应用需求和硬件能力调整这些参数，使框架的行为与之相适应。

参数 K 是我们框架中可调节权衡的控制参数。用户可以根据特定应用需求，在解的质量与计算效率之间进行平衡。为评估这种权衡，我们分析了K 对割值和执行时间的影响。图9 展示了不同 K 取值下的结果；该参数决定并行 QAOA 执行期间保留的高概率比特串数量（前K 个）。横轴表示 K，左纵轴显示执行时间，右纵轴表示 ParaQAOA 达到的割值。结果表明，增大 K 可提高解的质量，但会增加执行时间，体现了准确性与效率之间的权衡。在大多数情况下，ParaQAOA 达到的割值与 QAOA 相当[2]，且 $K = 1$ 或 $K = 2$执行时间显著更短，展现出其通过参数调优平衡解的质量与效率的能力。

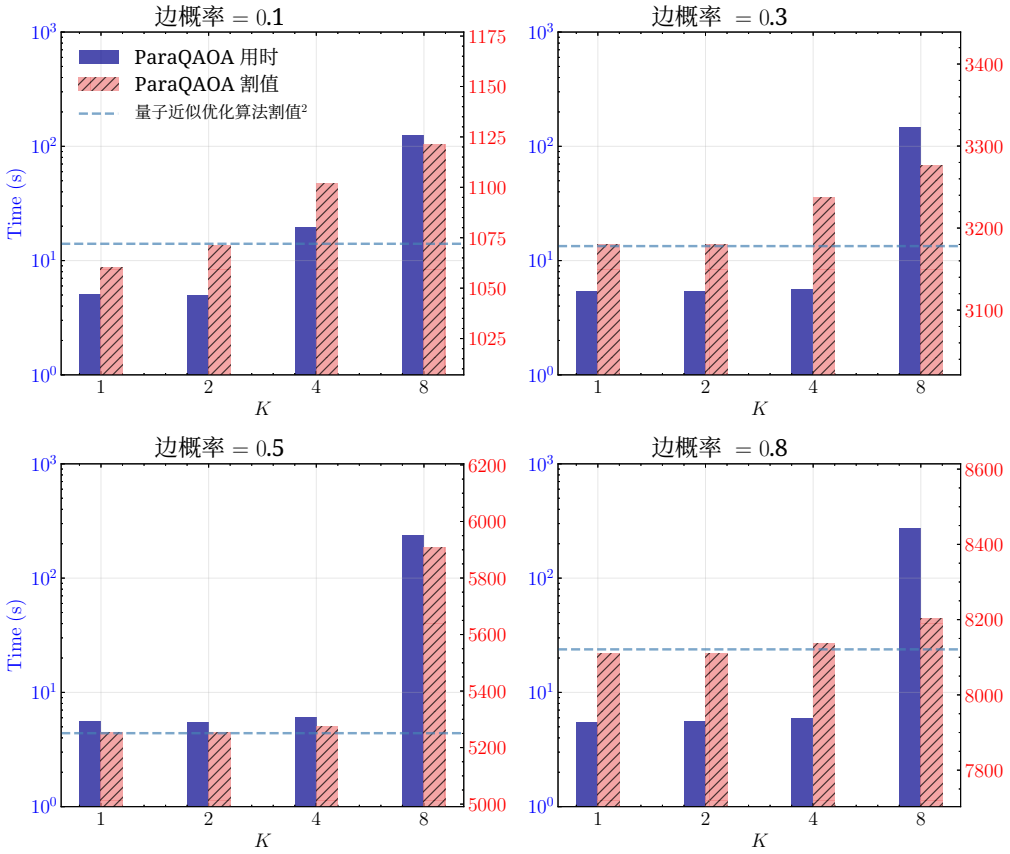


图 9. 在不同边概率下，ParaQAOA 在包含 200 个顶点的 Erdős-Rényi图上使用不同筛选参数 K 时的割值和执行时间。

参数 L 控制合并过程中的起始级别，并直接影响后处理期间的并行性。合理选择 L 可使并行合并操作与可用的 CPU 核心相匹配，从而提高计算效率。建议将 $2K^L$ 设置为接近物理 CPU 核心的数量，以确保高效利用硬件，同时避免产生系统开销。图 10 展示了 L 从 1 到 3 变化时 ParaQAOA 的性能。参数 L 决定并行执行的进程数量。例如，

当 $L = 1$时，会生成 4 个进程；当 $L = 2$时，会生成 8 个进程；而当 $L = 3$时，会生成 16 个进程。结果表明，将进程数量加倍可使运行时间缩短约一半。这凸显了 L对执行效率的影响，并强调了该框架在现代多核和众核系统上的可扩展性潜力 [20]。

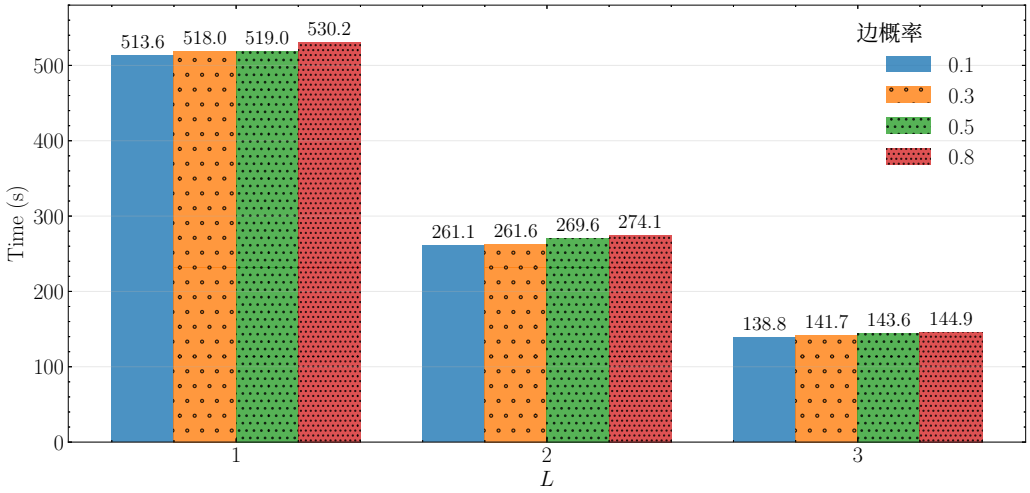


图 10. ParaQAOA 的执行时间随起始点 L 变化，在 Erdős-Rényi 图上针对不同边e 概率 with $K = 2$。

4.3 中小规模图的性能评估

我们首先评估小规模最大割实例，具体而言，是包含 20 至 30 个顶点的 Erdős-Rényi 随机图。结果汇总于表 2，其中比较了 ParaQAOA 与 CQ 和量子近似优化算法的近似比和执行时间[2]。结果表明，ParaQAOA 在实现具有竞争力的近似比的同时，在执行时间方面显著优于其他方法。例如，对于一个边概率为 0.1、含 20 个顶点的图，ParaQAOA 仅用 0.85 秒便实现了 82.5% 的近似比，而量子近似优化算法[2]需要 2.46 秒，CQ 则需要 106.87 秒才能达到相近的结果。随着图规模和边概率的增加，ParaQAOA 仍展现出优异的性能，在保持较高近似比的同时，将执行时间控制在较低水平。结果表明，ParaQAOA 在实现具有竞争力的近似比的同时，在执行时间方面显著优于其他方法。

对于中等规模的问题，我们将 ParaQAOA 框架的性能与量子近似优化算法进行比较[2]，，后者是一种先进的基于QAOA的最大割求解器。由于 CQ 的运行时间过长（处理一个含 30 个顶点的图需要超过 8 小时）、仅适用于二分图，且无法处理更大规模的实例，因此本次评估不包括 CQ。GW 算法作为基线，因为在此规模下蛮力方法并不可行。表3 和图11 总结了 ParaQAOA 和量子近似优化算法的性能结果[2]。

表3 报告了执行时间结果，并展示了 ParaQAOA 的计算效率。在所有配置下，我们的框架始终优于量子近似优化算法[2]，且加速比随图的边概率增加而提高。例如，在边概率为 0.1 的 100 个顶点图上，ParaQAOA 实现了 112.1× 的加速比；在 1652.2×

表 2. 不同图规模（$|V|$）和边概率（P）下，小规模最大割实例的运行时间和近似比比较。近似比数值通过将获得的割值除以暴力法确定的最优割值计算得出。

P	$\|V\|$	运行时间（秒）			近似比 (%)		
		量子近似优化算法[2]	CQ	ParaQAOA	量子近似优化算法[2]	CQ	ParaQAOA
0.1	20	2.46	106.87	0.85	84.6	94.8	82.5
	22	2.56	181.51	0.86	87.0	95.6	85.9
	24	2.93	280.82	0.87	83.7	96.1	90.7
	26	2.92	1044.23	0.89	84.7	96.8	81.4
0.3	20	8.18	151.88	0.85	93.8	97.7	89.4
	22	8.89	246.90	0.85	92.1	97.9	89.5
	24	8.71	488.65	0.87	91.5	98.4	90.9
	26	9.55	1531.19	0.91	90.9	98.4	86.9
0.5	20	16.18	179.31	0.86	95.6	98.4	92.0
	22	17.16	290.58	0.88	95.1	98.6	93.3
	24	18.92	501.91	0.90	94.5	98.7	93.9
	26	19.38	1590.81	0.91	94.1	98.8	93.0
0.8	20	33.29	219.46	0.87	96.9	98.5	94.6
	22	35.39	367.41	0.89	95.5	99.1	96.1
	24	34.23	586.28	0.90	96.4	99.1	97.2
	26	37.49	1592.60	0.91	95.4	99.1	95.6

边概率为 0.8 的 400 个顶点图上，加速比达到这一水平。这一性能趋势凸显了算法上的关键差异。量子近似优化算法[2] 由于采用穷举式子解枚举，计算时间会随着图的边概率增加而呈指数增长。相比之下，ParaQAOA 的运行时间主要取决于各个 QAOA 求解器的性能，因此对图复杂度的敏感度显著较低。

表 3. 在图规模 ($|V|$) 和边概率 (P) 不同的中等规模最大割实例上，比较 QAOA[2] 与 ParaQAOA 的执行时间。

P	$\|V\|$	**QAOA[2] 运行时间（秒）**	**ParaQAOA 运行时间（秒）**	**加速比**
0.1	100	668.0	6.0	112.1×
	200	1128.6	7.3	155.2×
	400	2158.8	8.7	247.0×
0.3	100	888.8	6.4	138.0×
	200	1754.1	7.7	227.2×
	400	3320.7	9.9	334.4×
0.5	100	1753.5	6.5	268.1×
	200	2937.7	7.9	372.8×
	400	6943.8	10.2	679.4×
0.8	100	4659.4	6.6	706.0×
	200	8591.3	8.2	1051.6×
	400	17001.0	10.3	1652.2×

图 11 展示了 ParaQAOA 与量子近似优化算法的近似比对比[2]。从结果中可以得出两个关键观察。首先，由于单条边对割值的影响增大，两种方法在边概率较低的图上都呈现出较低的近似比。由于两种框架都采用随机划分并忽略子图间边，稀疏图更容易出现近似性能下降。相比之下，在单条边贡献不那么显著的较稠密图上，两种方法的近似比都接近 GW 算法。其次，与量子近似优化算法相比，我们框架在最坏情况下的近似比下降幅度在 2% 以内，而典型差异约为 1%。在若干种配置下，ParaQAOA 的解质量甚至优于量子近似优化算法[2] 。

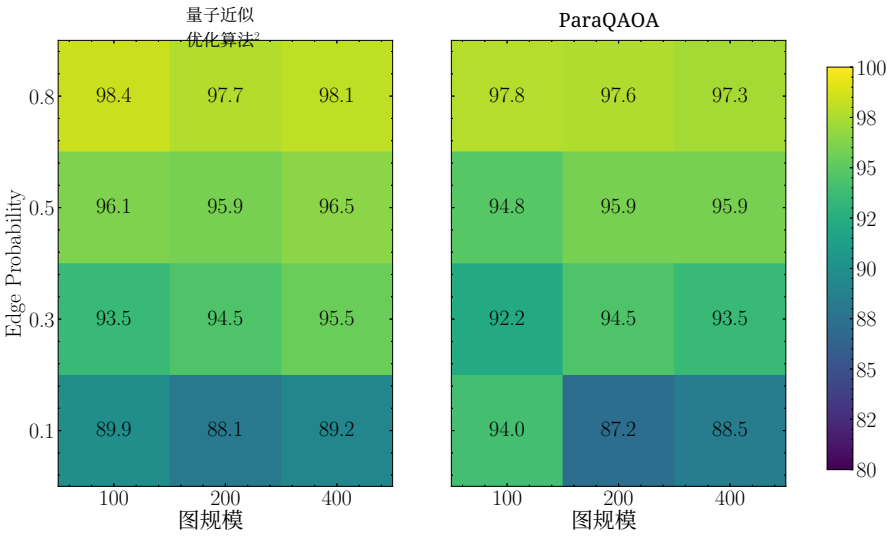


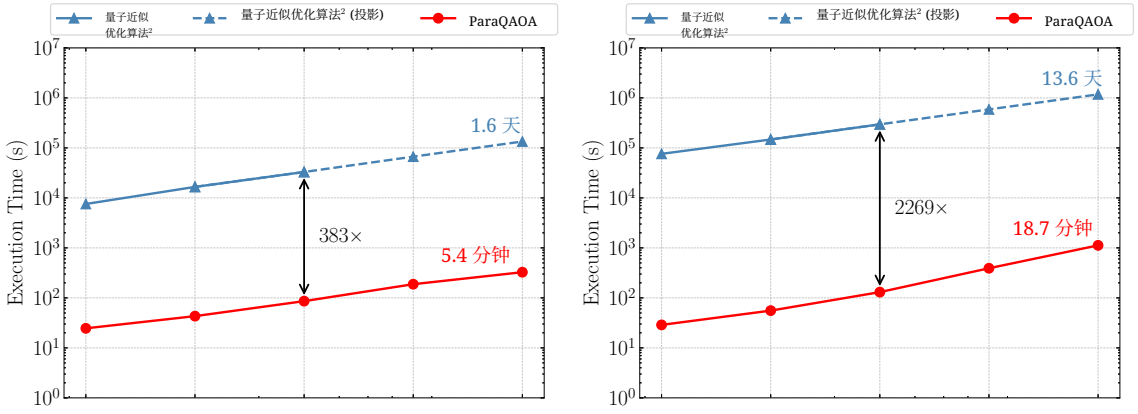
图 11。 使用 GW 的割值计算的近似比热图，对比量子近似优化算法[2] 和 ParaQAOA .

4.4 大规模图上的可扩展性结果

为评估可扩展性，我们将实验扩展到包含 1,000 至 16,000 个顶点的大规模图实例。在这一规模下，只有量子近似优化算法[2] 和我们的 ParaQAOA 框架能够求解这些问题。图 12 展示了边概率较低（0.1）和较高（0.8）的代表性配置下的执行时间趋势；中间边概率的情况也呈现出类似的扩展模式。由于量子近似优化算法[2] 在大规模图上的计算成本很高，因此其执行时间仅针对最多包含 4,000 个顶点的图进行了测量。对于更大的实例，我们根据观测到的顶点数量与量子近似优化算法[2]运行时间之间的关系，采用线性回归外推执行时间（标记为“预测”）。

我们的分析得出了两个关键观察结果。首先，随着问题规模的增大，QAOA[2] 的性能会受到边密度的显著影响。具体而言，对于顶点数量相同的图，将边概率从 0.1 提高到 0.8，会使执行时间增加约 10 倍。相比之下，我们的框架对边密度表现出更强的鲁棒性：对于包含 1,000 至 4,000 个顶点的图，在相同边概率范围内，执行时间最多只增加 1.5× 。其次，随着问题复杂度的增加，我们的框架在计算效率方面始终优于 QAOA[2] ，加速比介于 300× 和 2,000× 之间。除了比较相对性能外，我们还给出了绝对执行时间，以强调我们的方法在实际应用中的可行性

框架。对于包含 16,000 个顶点的图，我们的方法只需几分钟即可完成，而 QAOA^2 则需要数天，因此在这一规模下不适用于实际应用。



(a) 边概率 = 0.1 (b) 边概率 = 0.8

图 12。 比较量子近似优化算法^2 与 ParaQAOA 在大规模最大割问题上的执行时间的可扩展性分析 .

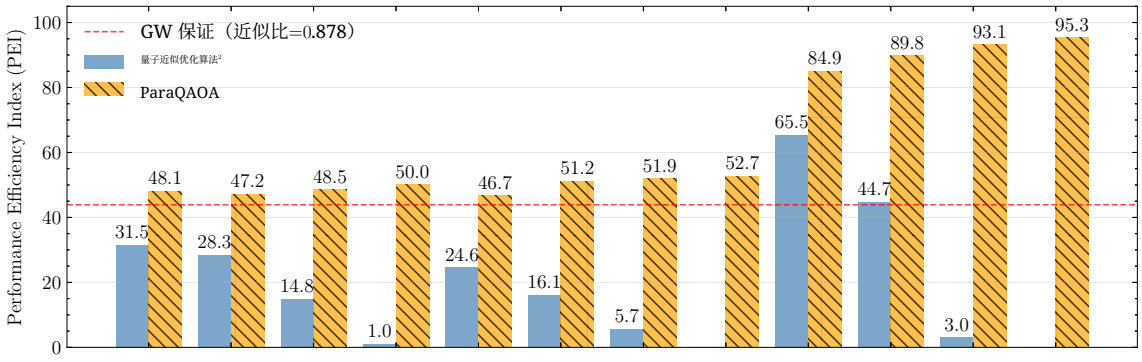
4.5 性能效率指数评估

如第 3.5 节所述，性能效率指数（PEI）是一种新颖的指标，它将近似比和执行时间合并为一个指数，用于评估不同解在最大割问题上的性能。图 13 展示了不同中等规模图配置下的 PEI 值，涵盖 100、200 和 400 个顶点，以及从 0.1 到 0.8 的边概率。在所提出的 PEI 指标中，GW 算法作为计算近似比和效率因子的基线，其中 $\alpha = 0.001$用于确保运行时间数据平滑缩放。我们提出的框架在所有测试配置中的表现均优于量子近似优化算法^2。值得注意的是，随着图复杂度增加（无论是顶点数增加还是边密度提高），这一优势都愈发明显。此外，我们的方法超过了 GW 算法的近似性能保证（以红色水平虚线表示），凸显了其在解的质量与计算效率之间实现更佳权衡的能力。

此外，我们还将 PEI 评估扩展到大规模最大割实例，如图 14 所示。由于这些图的规模以及现有实现的局限性，GW 算法不再适用。因此，我们使用量子近似优化算法^2 作为基线，并将其最佳割值作为参考。此外，参数 α 设为 0。0001，以确保运行时间数据平滑缩放，并反映这一问题规模下观察到的运行时间差距。结果表明，ParaQAOA 在所有测试的大规模图上仍能实现出色的性能效率。

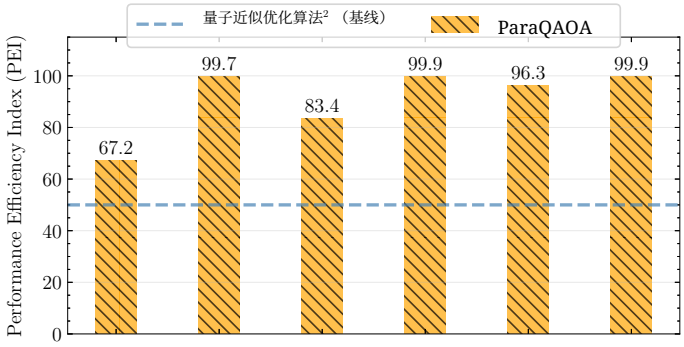
5 结论

本文提出了 ParaQAOA 框架，这是一种新颖的量子-经典混合方法，成功解决了大规模组合优化中解的质量与计算效率之间的根本性权衡。通过系统的算法创新和全面的实验验证，我们证明，实用的



图（规模-边概率）

图13。以 GW 算法为割值和运行时间的基线，对比 QAOA² 与 ParaQAOA 在不同图规模和边概率下的性能效率指数。



图（规模-边概率）

以 QAOA² 为割值和运行时间的基线，展示 ParaQAOA 在不同图规模和边概率下的性能效率指数。

量子启发式优化能够兼具可扩展性和适用于实际部署的性能。性能效率指数的引入提供了一个通用框架，用于评估不同优化算法在解的质量与计算效率之间的权衡。

尽管具有这些优势，仍有若干局限性为未来研究指明了方向。目前的随机划分在结构化图上的表现可能较差，因此值得探索自适应划分技术。虽然本方法针对最大割进行了评估，但将其扩展到其他 QUBO 问题（例如旅行商问题、图着色）可展现更广泛的适用性。硬件特定优化和噪声感知电路设计也是值得探索的方向。

致谢

本研究部分获得台湾国家科学及技术委员会资助，项目编号为 113-2119-M-002 -024 和 114-2221-E-006 -165 -MY3。感谢国家高速网络与计算中心（NCHC）、国立台湾大学高性能与科学计算中心，以及英业达提供计算与存储资源。感谢教育部高等教育深耕计划框架下的特色领域研究中心计划（114L900903）提供经费支持。

参考文献

[1] Rakesh Agrawal、Sridhar Rajagopalan、Ramakrishnan Srikant 和 Yirong Xu。2003。利用社会行为产生的网络挖掘新闻组数据。载于《第十二届万维网国际会议》（WWW）论文集。529–535。doi:10.1145/775152.775227[2] Frank Arute、Kunal Arya、Ryan Babbush、Dave Bacon、Joseph C. Bardin、Rami Barends、Rupak Biswas、Sergio Boixo、Fernando G. S. L. Brandao、David A. Buell、Brian Burkett、Yu Chen、Zijun Chen、Ben Chiaro、Roberto Collins、William Courtney、Andrew Dunsworth、Edward Farhi、Brooks Foxen、Austin Fowler、Craig Gidney、Marissa Giustina、Rob Graff、Keith Guerin、Steve Habegger、Matthew P. Harrigan、Michael J. Hartmann、Alan Ho、Markus Hoffmann、Trent Huang、Travis S. Humble、Sergei V. Isakov、Evan Jeffrey、Zhang Jiang、Dvir Kafri、Kostyantyn Kechedzhi、Julian Kelly、Paul V. Klimov、Sergey Knysh、Alexander Korotkov、Fedor Kostritsa、David Landhuis、Mike Lindmark、Erik Lucero、Dmitry Lyakh、Salvatore Mandrà、Jarrod R. McClean、Matthew McEwen、Anthony Megrant、Xiao Mi、Kristel Michielsen、Masoud Mohseni、Josh Mutus、Ofer Naaman、Matthew Neeley、Charles Neill、Murphy Yuezhen Niu、Eric Ostby、Andre Petukhov、John C. Platt、Chris Quintana、Eleanor G. Rieffel、Pedram Roushan、Nicholas C. Rubin、Daniel Sank、Kevin J. Satzinger、Vadim Smelyanskiy、Kevin J. Sung、Matthew D. Trevithick、Amit Vainsencher、Benjamin Villalonga、Theodore White、Z. Jamie Yao、Ping Yeh、Adam Zalcman、Hartmut Neven 和 John M. Martinis。2019。利用可编程超导处理器实现量子霸权。《自然》574，7779（2019），505–510。doi:10.1038/s41586-019-1666-5[3] Ryan Babbush、Dominic W. Berry、Robin Kothari、Rolando D. Somma 和 Nathan Wiebe。2023。模拟耦合经典振子的指数级量子加速。《物理评论 X》13（2023），041041。第4期。doi:10.1103/PhysRevX.13.041041[4] Francisco Barahona、Martin Grötschel、Michael Jünger 和 Gerhard Reinelt。1988。组合优化在统计物理和电路布局设计中的应用。《运筹学》36，3（1988），493–513。https://dl.acm.org/doi/10.5555/2804709.2804720[5] Harun Bayraktar、Ali Charara、David Clark、Saul Cohen、Timothy Costa、Yao-Lung L. Fang、Yang Gao、Jack Guan、John Gunnels、Azzam Haidar、Andreas Hehn、Markus Hohnerbach、Matthew Jones、Tom Lubowe、Dmitry Lyakh、Shinya Morino、Paul Springer、Sam Stanwyck、Igor Terentyev、Satya Varadhan、Jonathan Wong 和 Takuma Yamaguchi。2023。cuQuantum SDK：用于加速量子科学的高性能库。载于《IEEE量子计算与工程国际会议》（QCE）论文集。1050–1061。doi:10.1109/QCE57702.2023.00119[6] Kostas Blekos、Dean Brand、Andrea Ceschini、Chiao-Hui Chou、Rui-Hao Li、Komal Pandya 和 Alessandro Summer。2024。量子近似优化算法及其变体综述。《物理学报告》1068（2024），1–66。doi:10.1016/j.physrep.2024.03.002[7] M. Cerezo、Andrew Arrasmith、Ryan Babbush、Simon C. Benjamin、Suguru Endo、Keisuke Fujii、Jarrod R. McClean、Kosuke Mitarai、Xiao Yuan、Lukasz Cincio 和 Patrick J. Coles。2021。变分量子算法。《自然综述：物理学》3，9（2021），625–644。doi:10.1038/s42254-021-00348-9[8] Shin-Wei Chiu、Chuo-Min Yang、Shan-Jung Hou、Po-Hsuan Huang、Chuan-Chi Wang、Chia-Heng Tu 和 Shih-Hao Hung。2025。FOR-QAOA：用于求解最大割问题的完全优化资源高效QAOA电路模拟。载于《先进研究计算实践与经验2025：协作的力量》（PEARC）论文集。文章2，11页。doi:10.1145/3708035.3736006[9] 庄驰、Po-Hsuan Huang、Chia-Heng Tu 和 Shih-Hao Hung。2024。充分发挥QAOA潜力：通过经典并行搜索为时间敏感型应用高效求解最大割问题。载于《台湾消费电子国际会议》（ICCE-Taiwan）论文集。157–158。doi:10.1109/ICCE-Taiwan62264.2024.10674517[10] Vedran Dunjko、Yimin Ge 和 J. Ignacio Cirac。2018。利用小型量子设备实现计算加速。《物理评论快报》121（2018），250501。第25期。doi:10.1103/PhysRevLett.121.250501[11] Maxime Dupont、Bhuvanesh Sundar、Bram Evert、David E. Bernal Neira、Zedong Peng、Stephen Jeffrey 和 Mark J. Hodson。2025。在超导量子计算机上对最大割问题的量子优化进行基准测试。Physical Review Applied 23（2025），014045。第1期。doi:10.1103/PhysRevApplied.23.014045[12] Paul Erdös 和 Alfréd Rényi。1959。随机图论 I。《德布勒森数学出版物》6，290–297（1959），18。https://snap.stanford.edu/class/cs224w-readings/erdos59random.pdf[13] Edward Farhi、Jeffrey Goldstone 和 Sam Gutmann。2014。一种量子近似优化算法。arXiv:1411.4028 [quant-ph][14] Richard P. Feynman。1982。用计算机模拟物理。《国际理论物理学杂志》21，6（1982），467–488。doi:10.1007/ BF02650179[15] Yimin Ge 和 Vedran Dunjko。2020。面向小型量子计算机的混合算法框架及其在寻找哈密顿回路中的应用。《数学物理杂志》61，1（2020），012201。doi:10.1063/1.5119235[16] Michel X. Goemans 和 David P. Williamson。1995。利用半定规划改进最大割和可满足性问题的近似算法。《ACM杂志》42，6（1995），1115–1145。doi:10.1145/227683.227684

[17] Gian Giacomo Guerreschi 和 Anne Y. Matsuura。2019。最大割的QAOA需要数百个量子比特才能实现量子加速。《科学报告》9，1（2019），6903。[18] Aric A. Hagberg、Daniel A. Schult 和 Pieter J. Swart。2008。使用NetworkX探索网络结构、动力学和功能。载于《第七届科学计算中的 Python 会议》（SciPy）论文集。11–15。doi:10.25080/TCWV9851[19] Mark Horowitz、Thomas I ndermaur 和 Ricardo Gonzalez。1994。低功耗数字设计。载于《IEEE 低功耗电子学研讨会》（LPE）论文集。8–11。doi:10.1109/LPE.1994.573184[20] Jim Jeffers、James Reinders 和 Avinash Sodani。2016。第1章——引言。载于《英特尔至强融核处理器高性能编程》（第二版）。Morgan Kaufmann，3–13。doi:10.1016/B978-0-12-809194-4.00001-6[21] Tyson Jones、Anna Brown、Ian Bush 和 Simon C. Benjamin。2019。QuEST与量子计算机的高性能模拟。《科学报告》9，1（2019），10736。doi:10.1038/s41598-019-47174-9[22] Richard M. Karp。1972。组合问题之间的可归约性。Springer US，85–103。doi:10.1007/978-1-4684-2001-2_9[23] Gary Kochenberger、Jin-Kao Hao、Fred Glover、Mark Lewis、Zhipeng Lü、Haibo Wang 和 Yang Wang。2014。无约束二次二进制规划问题：综述。《组合优化杂志》28，1（2014），58–81。doi:10.1007/s10878-014-9734-0[24] Naimisha Kolli 和 Balakrishnan Narayanaswamy。2019。在缺少网络结构信息时，利用复杂网络中的级联信息轨迹实现影响力最大化。《IEEE 计算社会系统汇刊》6（2019），1147–1155。doi:10.1109/TCSS.2019.2939841[25] Siu Kwan Lam、Antoine Pitrou 和 Stanley Seibert。2015。Numba：基于 LLVM 的 Python 即时编译器。载于《第二届高性能计算中的 LLVM 编译器基础设施研讨会》（LLVM）论文集。文章7，6页。doi:10.1145/2833157.2833162[26] James H. Laros III、Kevin Pedretti、Suzanne M. Kelly、Wei Shu、Kurt Ferreira、John Vandyke 和 Courtenay Vaughan。2013。能量延迟积。Springer London，51–55。doi:10.1007/978-1-4471-4492-2_8[27] Ang Li、Omer Subasi、Xiu Yang 和 Sriram Krishnamoorthy。2020。通过现代GPU集群上的BSP机器进行密度矩阵量子电路模拟。载于《高性能计算、网络、存储与分析国际会议》（SC）论文集。1–15。doi:10.1109/SC41405.2020.00017[28] Jianqiang Li。2025。寻路问题量子算法的指数级加速。《量子信息处理》24，3（2025），67。doi:10.1007/s11128-025-04689-7[29] Junde Li、Mahabubul Alam 和 Swaroop Ghosh。2023。通过分治法实现大规模量子近似优化。《IEEE 集成电路与系统计算机辅助设计汇刊》42（2023），1852–1860。doi:10.1109/TCAD.2022.3212196[30] Yu-Cheng Lin、Chuan-Chi Wang、Chia-Heng Tu 和 Shih-Hao Hung。2024。面向使用QAOA求解最大割问题的量子电路模拟优化。载于《第39届ACM/SIGAPP应用计算研讨会》（SAC）论文集（西班牙阿维拉）。1487–1494。doi:10.1145/3605098.3635897[31] Yiren Lu、Guojing Tian 和 Xiaoming Sun。2023。使用更少量子比特的QAOA：一种求解更大规模最大割问题的耦合框架。arXiv:2307.15260 [quant-ph][32] Jhon Alejandro Montañez-Barrera 和 Kristel Michielsen。2025。迈向线性递增 QAOA 协议：在求解某些组合优化问题时展现尺度优势的证据。《npj 量子信息》11，1（2025），131。doi:10.1038/s41534-025-01082-1[33] Roger Penrose 等。1971。负维张量的应用。《组合数学及其应用》1，221–244（1971），3。https://homepages.math.uic.edu/~kauffman/Penrose.pdf[34] John Preskill。2018。含噪中型量子时代及其之后的量子计算。《Quantum》2（2018），79。doi:10.22331/q-2018-08-06-79[35] Stefan H. Sack 和 Maksym Serbyn。2021。量子退火初始化量子近似优化算法。《Quantum》5（2021），491。doi:10.22331/q-2021-07-01-491[36] Peter W. Shor。1994。量子计算算法：离散对数与因数分解。载于《第35届计算机科学基础年会》（SFCS）论文集。124–134。doi:10.1109/SFCS.1994.365700[37] Daniel R. Simon。1994。论量子计算的能力。载于《第35届计算机科学基础年会》（SFCS）论文集。116–123。doi:10.1109/SFCS.1994.365701[38] Damian S. Steiger、Thomas Häner 和 Matthias Troyer。2018。ProjectQ：用于量子计算的开源软件框架。《Quantum》2（2018），49。doi:10.22331/q-2018-01-31-49[39] Michael Streif、Martin Leib、Filip Wudarski、Eleanor Rieffel 和 Zhihui Wang。2021。具有局域粒子数守恒的量子算法：噪声效应与纠错。《物理评论 A》103（2021），042412。第4期。doi:10.1103/PhysRevA.103.042412[40] Jules Tilly、Hongxiang Chen、Shuxiang Cao、Dario Picozzi、Kanav Setia、Ying Li、Edward Grant、Leonard Wossnig、Ivan Rungger、George H. Booth 和 Jonathan Tennyson。2022。变分量子本征求解器：方法与最佳实践综述。《物理学报告》986（2022），1–128。doi:10.1016/j.physrep.2022.08.003[41] Steven R. White。1993。用于量子重整化群的密度矩阵算法。《物理评论 B》48（1993），10345–10356。第14期。doi:10.1103/PhysRevB.48.10345[42] Yu-Tsung Wu、Po-Hsuan Huang、Kai-Chieh Chang、Chia-Heng Tu 和 Shih-Hao Hung。2025。QOPS：用于量子电路模拟加速并采用基于性能剖析优化的编译器框架。《超级计算杂志》81（2025）。doi:10.1007/s11227-025-07157-2[43] Ming Yang、Bo Liu、Wei Wang、Junzhou Luo 和 Xiaojun Shen。2014。基于最大割的802.11 无线网状网络最大容量重叠信道分配。《通用计算机科学杂志》20，13（2014），1855–1874。doi:10.3217/jucs-020-13-1855

[44] Kieran Young、Marcus Scese 和 Ali Ebnenasir。2023。经典机器上的量子计算模拟：一项综述。arXiv：2311。16505 [quant-ph][45] Leo Zhou、Sheng-Tao Wang、Soonwon Choi、Hannes Pichler 和 Mikhail D. Lukin。2020。量子近似优化算法：近期设备上的性能、机制与实现。《物理评论 *X*》10（2020），021067。第 2 期。doi:10.1103/PhysRevX.10.021067[46] Zeqiao Zhou、Yuxuan Du、Xinmei Tian 和 Dacheng Tao。2023。QAOA-in-QAOA：在小型量子机器上求解大规模最大割问题。*Physical Review Applied* 19（2023），024027。第 2 期。doi:10.1103/PhysRevApplied.19.024027