应用于二次指派问题的量子优化

Andrew Freeland 和 Jingbo Wang*

量子信息、模拟与算法中心，

西澳大学，澳大利亚 *WA 6009* 珀斯

本文研究新兴的非变分基于量子游走的优化算法（NV-QWOA） [1–3] 在求解小规模二次指派问题（QAP）实例时的性能。我们将 NV-QWOA 与经典启发式算法进行基准比较，包括最大-最小蚁群系统（MMAS） [4] 和贪心局部搜索（GLS） [3, 5]，并与作为量子基线的格罗弗量子搜索算法 [6], 进行比较。我们采用两项指标评估性能：目标函数评估次数，以及在包含 $n = 5$ 至 $n = 10$ 个设施的 QAP 实例中稳定获得最优解或近优解所需的算法迭代次数。本研究的动机源于经典精确方法和当前量子算法的局限性。变分量子算法（VQAs），例如量子近似优化算法（QAOA）和变分量子本征求解器（VQE），尽管已得到广泛研究，却受到高成本参数调优和贫瘠高原的困扰，阻碍了算法收敛。本文采用非变分方法，探索一种可能更高效且可扩展的量子策略，以解决组合优化问题。研究结果对经典与量子框架进行了直接比较，刻画了 NV-QWOA 在平均情况下的性能。我们的发现凸显了量子游走在复杂组合问题中的实用价值，并为未来的量子优化算法奠定了基础。

arXiv:2601.01104v1 [quant-ph] 3 Jan 2026

一、引言

大多数行业都面临棘手的组合优化问题（COP），如何制定能够大规模求解这些问题的策略，仍是一项重大挑战。在这类组合优化问题中，最难的一类是在计算复杂性理论中被归为非确定性多项式时间困难（NP困难）的问题。在$P \neq NP$这一假设下，人们普遍认为经典算法难以解决这类问题。这是因为，解空间会随问题规模呈指数级增长，而利用问题结构加速求解的机会有限。

一个特别值得关注的组合优化问题是二次指派问题（*QAP*），它被认为是最难的 NP 困难问题之一。即使是找到一个较小的 ϵ-近似解也是 NP 困难的，因为多项式时间近似方案将意味着 $P = NP$ [7]。QAP 用途广泛，既可用于有约束情形，也可用于无约束情形。它最常见的表述是设施选址指派问题，并且在医院管理 [8],中有大量文献记载的应用。

* jingbo.wang@uwa.edu.au

排班 [9], 和配电板布线 [10]。这些应用因其与现实世界的相关性，经常在 QAP 研究中被引用。QAPLIB 数据库 [11] 收录了基准实例，为经典算法的开发提供了指导，并且至今仍是评估算法性能的标准。

本研究基准评估了新兴的非变分量子游走优化算法（*NV-QWOA*）求解 QAP 实例的性能。我们将 NV-QWOA 与成熟的经典启发式算法——最大-最小蚁群系统（MMAS） [4]和贪心局部搜索（GLS） [3, 5], ——以及作为量子基准的格罗弗量子搜索算法 [12], 进行比较。与经典启发式算法比较时，主要指标是目标函数评估次数；对于 NV-QWOA 与 GLS 的比较，我们还考察测量结果和解的概率。对于格罗弗算法，我们比较了不同问题规模下所需的量子迭代次数。

A. 经典算法

用于求解 QAP 和其他组合优化问题（COP）的经典算法通常分为两类：精确算法和启发式算法。精确算法穷举解空间，系统地剔除低质量解，以保证找到全局最优解。相比之下，启发式算法会持续搜索解空间，直到无法进一步改进。尽管启发式算法不保证找到最优解，且往往会收敛到局部最优，但其效率远高于精确算法。关于精确算法和启发式算法的更多细节，尤其是用于求解 QAP 的最大-最小蚁群系统（MMAS）和贪心局部搜索（GLS），请参见第 II节。

在定义二次指派问题（QAP）的奠基性论文发表后不久，Gilmore 和 Lawler [14, 15] 便各自独立地发现，可以将 QAP 化简为线性指派问题（LAP），从而得到 QAP 的一个下界；此外，LAP 的置换矩阵 L 必须是 QAP 置换矩阵的克罗内克平方 X： $L = X^{\otimes 2}$。这个下界称为 Gilmore–Lawler下界（GLB），其时间复杂度为 $O(n^3)$，但随着 n 增大，其性能会下降。GLB 推动了众多松弛和化简方法的发展，从而加速分支定界算法 [16], 、切平面法 [17], 和多面体方法 [18]。半定松弛—矩阵分裂（SDRMS）下界等更先进的下界 [19], 则利用松弛和重构，在包括 Taillard基准在内的大型困难实例上取得更小的最优性间隙 $n > 50$ [11, 20, 21]。虽然这些下界在间隙大小方面优于 GLB，但通常计算成本更高，因此需要在运行时间和解的质量之间权衡。

目前已知最好的 -二次指派问题精确算法是分支 -and-分支定界（B&B）方法。这些算法

可以求解规模约达 $n = 30$ 的实例，但需要大量的并行计算时间 [22]。精确方法最适用于小型到中等规模的问题，在这些问题中，它们能够以具有竞争力的运行时间保证获得最优解。它们还在为 QAPLIB 等基准数据集生成最优解方面发挥着关键作用。启发式算法通常分为局部搜索和基于种群的两类。示例包括贪心局部搜索（GLS） [23], 迭代局部搜索（ILS） [24], 模拟退火 [25], 禁忌搜索 [26], 蚁群优化（ACO） [27], 以及进化算法或遗传算法 [28, 29]。基于种群的算法通常会结合局部搜索子程序，以快速找到局部极小值，并且通常能够以较小的差距返回近优解，尽管其计算强度更高 [4]。

B. 量子算法

量子组合优化是一个新兴研究领域，其发展动力源于解决二次指派问题（QAP）等难题的需求。早期方法采用量子绝热算法（QAA） [30, 31], ，逐渐将初始哈密顿量演化为最终哈密顿量，其基态编码了最优解。在演化速度足够慢的情况下，系统会保持在基态，从而确保测量以较高概率得到最优解。随后，Farhi 等人 [32] 提出了量子近似优化算法（QAOA），这是一种有别于绝热方法的基于门的方法。QAOA 使用一个参数化量子电路，在两个哈密顿量之间交替作用，并通过 $2p$ 经典反馈环优化参数。其性能高度依赖参数初始化和电路深度：更深的电路虽然能提升表达能力，但也会增加计算成本 [33]。QAOA 还容易受到贫瘠高原的影响——代价景观中梯度消失的区域会阻碍收敛 [34]。

Marsh 和 Wang [35–37] 将 QAOA 推广为基于量子游走的优化算法（*QWOA*）。QWOA 以连续时间量子游走（CTQW）取代横场哈密顿量， [38, 39], 该游走仅连接搜索空间中的可行解。这种设计可防止振幅流向无效状态，与基于惩罚的方法相比，改进了约束处理 [40–43]。在 QWOA 的基础上，Bennett 等人 [1–3] 提出了非变分基于量子游走的优化算法（*NV-QWOA*），将基于 CTQW 的混合算子与 QAOA 原理相结合，以利用问题结构。与需要大量参数调优的传统 QAOA/QWOA 算法不同，NV-QWOA 只需三个超参数来控制演化时间和游走动力学。这一精简消除了成本高昂的经典优化循环，提升了可扩展性，并缓解了贫瘠高原现象 [1–3]。因此，NV-QWOA 为资源高效的量子优化提供了一个前景广阔的方向，同时降低了对训练病态现象的敏感性。

II. 理论背景

本节详细介绍二次指派问题的数学描述，以及所比较的经典和量子启发式算法背后的算法流程。纳入考虑的经典算法是最大-最小蚁群系统和贪心局部搜索。纳入考虑的量子算法是 NV-QWOA 和格罗弗量子搜索算法；与 QAOA、VQE 和 QWOA 等变分量子算法需要大规模参数调优且计算开销高昂相比，两者都提供了一种更“非变分”的组合优化问题求解方法。鉴于 QAOA 和 QWOA 与 NV-QWOA 的发展密切相关，文中也介绍了它们的背景。

A. 二次指派问题

二次指派问题（QAP）最早由 Koopmans 和 Beckmann 于 1957年提出，作为一种与不可分割的经济活动相关的数学模型，用于在收益最大化的背景下将生产工厂分配到各个位置 [13]。人们很快认识到该问题在计算上的难度，并制定了相应方法来确定成本景观的下界，并为经典算法设计提供参考 [16, 17]。

二次指派问题的定义是：将一组设施分配到一组位置，使指派总成本最小。总指派成本由设施之间的流量乘积之和构成 $f_{i,\ j}$ 再乘以它们所分配位置之间的距离 $d_{\pi(i),\ \pi(j)}$。图 1展示了一个二次指派问题示例，以及相应的流量矩阵和距离矩阵 F，它们见下文公式 D 1 和2 ：

$$F = \begin{bmatrix} 0 & 10 & 11 & 12 \\ 10 & 0 & 13 & 8 \\ 11 & 13 & 0 & 15 \\ 12 & 8 & 15 & 0 \end{bmatrix}, \tag{1}$$

$$D = \begin{bmatrix} 0 & 16 & 10 & 32 \\ 16 & 0 & 9 & 15 \\ 10 & 9 & 0 & 18 \\ 32 & 15 & 18 & 0 \end{bmatrix}. \tag{2}$$

最小化总指派成本的基本原则是：流量较大的设施应分配到彼此距离较近的位置，而流量较小的设施则可以安排得相距较远。流量矩阵和距离矩阵通常都是对称矩阵，且对角线元素为零；不过，非对称实例也已有相关研究 [11, 44]。本文仅考虑对称矩阵。

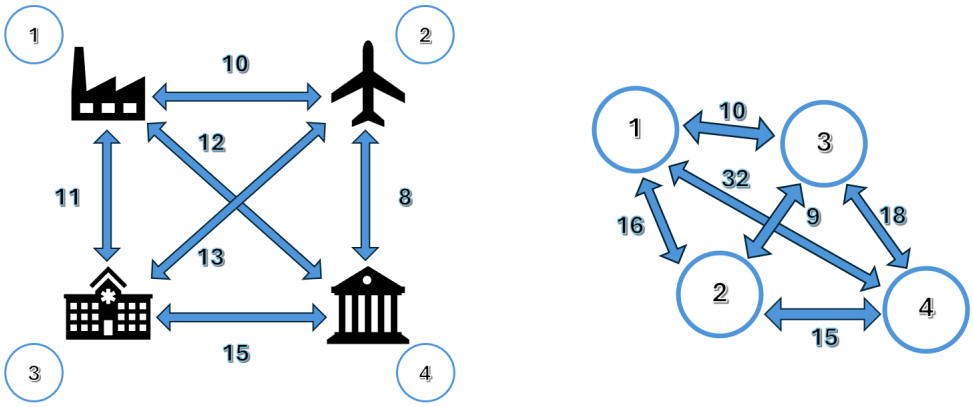

图 1. 示例 $n = 4$ 二次指派问题中的设施流量和位置距离

对称二次指派问题的目标函数如下，目标是实现最小化：

$$f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{i,j} d_{\pi(i),\pi(j)}, \tag{3}$$

其中，$f_{i,\ j}$ 是设施 i 之间的流量，而 j, $d_{\pi(i),\ \pi(j)}$ 是位置 $\pi(i)$ 和 $\pi(j)$ 之间的距离，π：$\{1, 2, ..., n\} \to \{1, 2, ..., n\}$ 表示将设施映射到位置的排列。给定二次指派问题实例的可能解数量，对应于所有可能排列集合 $S_n = \{\pi_i\}_{i=1}^{N}$ 中将设施指派到位置的排列数量。给定n 个设施和位置，二次指派问题的规模定义为 n，共有 $N = n!$ 个可能解；解的数量呈阶乘增长，使二次指派问题具有 NP 难度。

为了能够根据成本提取解，需要对排列进行适当的索引，因为解空间的阶乘增长使得跟踪和存储解变得困难，而且索引会将解与成本一一对应。本文所用的索引算法见算法 1；反索引算法可根据给定索引返回对应的排列，对于找到与最优成本对应的解很有用，见算法 2。

B. 最大-最小蚁群系统

最大-最小蚁群系统（MMAS）是基于种群的蚁群优化（ACO）启发式算法的一种变体 [4]。在 ACO 算法中，会生成许多称为蚂蚁的智能体，通过从初始为空的解中添加解组件（局部设施-位置指派）来为给定的 QAP 实例构建新解，并将信息素编码在一个 $n \times n$矩阵 $\tau_{i,\ j}$中，以此向下一代蚂蚁指示最优和非最优解区域；较优的解组件会获得更多信息素，从而鼓励下一代蚂蚁

算法1 排列索引

输入： 问题规模 n，排列 $\pi_i \in S_n$

输出： 索引 i 为 π_i

初始化 $i \leftarrow 0$

生成元素列表 $\phi \leftarrow \{0, 1, ..., n-1\}$

生成 $factorials = \{0!, 1! ..., (n-1)!\}$

初始化 $depth \leftarrow n-1$

对于每个 $k \in \pi_i$，**执行**

    获取 ϕ 中 k 的索引 l

    从 ϕ 中移除 k

    $i \leftarrow i + l \cdot factorials[depth]$

    $depth \leftarrow depth - 1$

结束循环

Algorithm2 排列取消索引

输入： 问题规模 n，索引 i

输出： 排列 π_i

初始化 $\pi_i = []$

生成元素列表 $\phi \leftarrow \{0, 1, ..., n-1\}$

生成 $factorials = \{0!, 1! ....., (n-1)!\}$

初始化 $depth \leftarrow n-1$

当 ϕ != [] **时**

    $block \leftarrow factorials[depth]$

    弹出 ϕ 中第 $i//block$ 个元素，并追加到 π_i

    $i \leftarrow i\%block$

    $depth \leftarrow depth - 1$

结束 while 循环

使用这些解组件，尝试找到最优解。在 MMAS 中，每次迭代只选择一只蚂蚁来沉积新的信息素，而该解可能是也可能不是最优解。为避免搜索陷入停滞，所有信息素轨迹都会按一个因子ρ 递减，该因子称为信息素蒸发率。蚂蚁还会受到依赖于问题的启发式信息的引导，该信息用于指示特定设施与位置指派配对的预期效果，并编码在一个 n×n矩阵$\beta_{i,\ j}$中。对于 QAP，这种信息可能基于完成部分解的有效下界，通常是 GLB [45], ，但 [4] 中的研究发现，由于启发式信息对解质量的影响微乎其微，因此并不需要使用它。

MMAS 参数包括：蚂蚁数量 m、信息素重要性 α、信息素矩阵$\tau_{i,\ j}$、初始信息素水平 τ_0、信息素蒸发率 ρ、信息素更新常数 Q、启发式信息重要性 β、启发式信息矩阵 $\beta_{i,\ j}$以及最大迭代次数 T。一般

MMAS 的示意图见算法3 ，其推导自 [4]:

算法3MMAS 算法

输入：问题规模 n、流量矩阵和距离矩阵 F 及 D、最大最小蚁群系统参数

输出：优化后的解成本 x

$I \leftarrow 1$

当 $I < T$ **时**

  对每只蚂蚁 k **执行**

    逐步构建完整分配 π_k ，经过 $p_{i,\ j}^{k} = \frac{(\tau_{i,j})^{\alpha}(\beta_{i,j})^{\beta}}{\sum_{l \in Available}(\tau_{i,l})^{\alpha}(\beta_{i,l})^{\beta}}$

    $x \leftarrow f(\pi_k)$

    $\tau_{i,j} \leftarrow (1-\rho)\tau_{i,j}$

    $\tau_{i,j} \leftarrow \tau_{i,j} + \Delta\tau_{i\ j}^{best},$，其中 $\Delta\tau_{i}^{best}{}_{,j} = \begin{cases} \frac{Q}{f(\pi_{best})} & \text{如果 } \pi_{best}(i) = j \\ 0 & \text{否则} \end{cases}$

    $\tau_{i,j} \leftarrow \min(\tau_{max}, \max(\tau_{min}, \tau_{i,j}))$

  结束循环

  $I \leftarrow I + 1$

结束循环

针对二次指派问题设计的 MMAS 算法，是对针对旅行商问题设计的算法的扩展：它仅修改了解的构造步骤，使其与设施分配公式而非路径相对应；此外，由于启发式信息$\beta_{i,\ j}$ 对解的劣化影响微乎其微，因此不使用启发式信息。本文中，我们设定启发式信息重要性 $\beta \leftarrow 0$。

C. 贪心局部搜索

贪心局部搜索是一种局部搜索算法的变体，它从一个随机初始解出发，对排列中元素之间所有可能的换位交换进行尝试，以找到性能最佳的交换。找到最佳交换后，会持续执行寻找最佳交换的过程，直到找不到能够改进解的交换为止。这一终止条件确保 GLS 能找到一个最小值，通常是局部最小值，因为局部最小值的数量可能会随着问题规模 n 大幅增加，正如 [3] 中的最大割问题所示。GLS 能高效地找到至少一个局部最小值，并在每次迭代中执行 $O(n^2)$，目标函数的评估次数为 $N_{f(x)} = iterations \cdot {}^{n}{}_{(n-1)\ 2}$ ，其中第二项表示从初始排列出发可能进行的换位次数。

所使用的 GLS 算法见算法 4。用于比较的特定 GLS 并未通过随机化来跳出局部最优并探索新的景观区域，而是保持为纯粹的爬山式局部搜索贪心算法，以突出其与 NV-QWOA 的对比。NV-QWOA 利用所考虑问题的结构，精准地放大最优解和近优解。

算法4 贪心局部搜索

输入： 问题规模 n、流量矩阵和距离矩阵 F 和 D

输出： 优化后的解成本 x

生成随机排列 π
$x \leftarrow f(\pi)$
$improved \leftarrow True$
当 $improved = True$ **时循环执行**
  $\Delta_{best} = 0$
  Δ_{best} 索引 $\Delta_i = -1$ 和 $\Delta_j = -1$
  $\Delta_{best} \leftarrow \Delta_{swap}(\pi, i, j)$ 对于 $i = 0, 1, ..., n-2$ 以及 $j = i+1, i+2, ..., n-1$
  更新 Δ_i 和 Δ_j
  如果 $\Delta_{best} < 0$ **则**
    $\pi[\Delta_i, \Delta_j] \leftarrow \pi[\Delta_j, \Delta_i]$
    $x \leftarrow x + \Delta_{best}$
  结束如果
  若 $\Delta_{best} \geq 0$ **则**
    $improved \leftarrow False$
  结束 if
结束 while

D. 量子近似优化算法

量子近似优化算法（QAOA） [32] 最初是为求解最大割问题而开发的；最大割是量子组合优化领域中另一类备受关注的组合优化问题。给定问题的成本函数 $C(x)$ 被编码为问题哈密顿量 $H_p = C$，其基态对应于最优解。解 x 表示为 k 位二进制字符串 $b_0 b_1 ... b_{k-1}$，算法则制备一个参数化量子态，并通过最小化或最大化期望值 $\langle \beta, \gamma | C | \beta, \gamma \rangle$，其中 $\beta = [\beta_1, \beta_2, ... \beta_p]$ 和 $\beta = [\gamma_1, \gamma_2, ... \gamma_p]$ 是由 p参数角组成的向量。

量子交替算子拟设（QAOAz）框架 [46] 将 QAOA 中使用的运算族加以推广，其涉及将初始哈密顿量的基态（通常为均匀叠加态）映射到给定问题哈密顿量的基态。此映射通过依次应用问题酉算子实现，该算子由 H_p 和 γ 组成，以影响成本质量；此外，还应用由混合器哈密顿量 $H_m = \sum_{j=1}^{N} \sigma_j^x$ 组成的混合器酉算子，促使解的二进制字符串转化为量子自旋，从而得到参数化量子态：

$$|\beta, \gamma\rangle = \prod_{l=1}^{p} e^{-i\beta_l H_m} e^{-i\gamma_l H_c} |s\rangle^{\otimes k}, \tag{4}$$

其中 $|s\rangle$ 是覆盖 k 个量子比特、用于编码解的叠加态。

E. 基于量子游走的优化算法

基于量子游走的优化算法（QWOA） [36]是在 QAOA 和 QAOAz 的发展基础上提出的，但它将混合器酉算子（公式8）定义为混合图上的连续时间量子游走（CTQW），并将问题酉算子定义为相移幺正算子（公式7），该算子根据解的目标函数值（也称为解的质量或成本），以成比例的相位区分各个解状态。与 QAOA 中的问题哈密顿量和混合哈密顿量类似，这些酉算子会依次作用于初始均匀叠加态，该状态编码了问题的所有解，并重复应用 p 次。混合图最初是一个超立方体，其连通性可变，从而能够逐步找到最优解。

QWOA 还需要高效的索引算法和反索引算法，将解映射到混合图节点，从而执行量子操作来放大状态，提高测得最优解的概率。经过 p 次交替酉算子应用后，放大态表示为：

$$|\gamma, t\rangle = U_M(t_p)U_Q(\gamma_p)...U_M(t_2)U_Q(\gamma_2)U_M(t_1)U_Q(\gamma_1)\,|s\rangle\,. \tag{5}$$

F. 非变分 QWOA

作为量子近似优化算法和 QWOA 的后继者，NV-QWOA 通过交替应用两个酉算符 p 次，将所有解状态的初始均匀叠加态演化为最终状态；在该状态中，最优解和近优解得到显著放大，而低质量解则被显著抑制 [1–3]。随后，在测量次数预算内进行少量重复态制备和测量，即可测得最优解。
NV-QWOA 与 QWOA 及其他变分量子算法的区别在于：它能够为所关注的问题选择相关的混合器（混合图），并通过解状态相位的组合，使近优解产生相长干涉、次优解产生相消干涉；与之相对，优化参数仅有 3 个，而非 $2p$。

在这一框架中，通过独热编码方案将二次指派问题的解嵌入输入寄存器的计算基态中：为每个 n 变量x_j分配一个子寄存器，并确定每个子寄存器的 n 计算基态，这些基态直接编码变量值 $x_j \in \{0, 1, ..., n-1\}$，因此总共需要 n^2 个量子比特。由于排列的不重复约束（所有元素均不相同），各子寄存器的状态相互依赖；在 [2]中提出了一种复杂度为 $O(n^3)$的均匀叠加态制备方案。将所有形式为 $|\mathbf{x}\rangle = \prod_{j=0}^{N-1} |x_j\rangle$ 的解状态编码为均匀叠加态的初始量子态表示为

by:

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{\mathbf{x} \in S_n} |\mathbf{x}\rangle \tag{6}$$

NV-QWOA包含 p 层交替酉算子应用，分别为相移幺正算子U_Q 和混合幺正算子 U_M。对于每一层，U_Q 向每个解状态施加一个相位 $e^{-i\gamma Q}$ ，其大小与质量矩阵 Q中包含的目标函数值及参数 γ成正比，表示如下：

$$U_Q(\gamma) = e^{-i\gamma Q}. \tag{7}$$

随后，混合幺正算子会在相邻解之间分散振幅概率，持续一段游走时间 t，如下所示：

$$U_M(t) = e^{-itA}, \tag{8}$$

其中， A是邻接矩阵，用于定义解状态作为图顶点时的连通关系。转置混合图将仅相差一次转置交换（交换 2 个排列元素）的解定义为最近邻，这等价于两个解之间的汉明距离为 2。随后，通过从初始顶点/排列 A迭代构造相邻顶点来建立。

整个 $U_M(t)U_Q(\gamma)$ 在整个电路深度 p 上的作用由下方的公式 9 表示：

$$|\gamma, t, \beta\rangle = \left[\prod_{i=0}^{p-1} U_M(t_i) U_Q\left(\frac{-\gamma_i}{\sigma}\right)\right] |s\rangle \tag{9}$$

每个 γ_i 和 t_i 值都由线性渐变定义，与 p 无关、取决于 β；其中， γ_i在定义域 $[\beta\gamma, \gamma]$内线性递增，而 t_i 在定义域 $[\beta t, t]$内递减：

$$\begin{aligned} \gamma_i &= \left(\beta + (1-\beta)\frac{i}{p-1}\right)\gamma, \\ t_i &= \left(1 - (1-\beta)\frac{i}{p-1}\right)t. \end{aligned} \tag{10}$$

根据统计推断，每个原始 γ 值都会按样本实验 [2]得出的标准差进行缩放。

以下给出了适用于量子硬件的 NV-QWOA 完整算法：

对于二次指派问题，适当的混合图是换位图，因为最近邻解彼此互为换位；图 2 给出了适用于 $n = 4$ QAP 实例的换位混合图示例。混合图通过邻接矩阵2构建

$$A = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} SWAP_{i,j}, \tag{11}$$

算法5 非变分 QWOA 流程

1: 初始化可行解的均匀叠加态 $|s\rangle$，其中每个 $|x\rangle$ 分配一个包含 n 个量子比特的子寄存器，作为 in 6

2: 交替重复应用 7 和 8 p 次，使用 $\{\gamma_i, t_i\}$ 得到放大态 $|\psi\rangle$，它来自 9

3: 测量 $|\psi\rangle$ 以返回近似解 x^*，其成本/质量可通过 $f(x)$快速确定

4:重复步骤 1-3，执行所需的测量次数（态制备和测量），以进行近似 $\langle\gamma, t, \beta|Q|\gamma, t, \beta\rangle$

5: 最优解定义为所遇到的成本/期望值最低的解，理想情况下即全局最小值

其中，$SWAP$ 算符是一个置换矩阵，它交换 i^{th} 和 j^{th} 寄存器的状态，并对其余寄存器执行恒等操作。A 本身不是酉算子，但混合操作本身 $U_M(t) = e^{-itA}$ 是酉的。

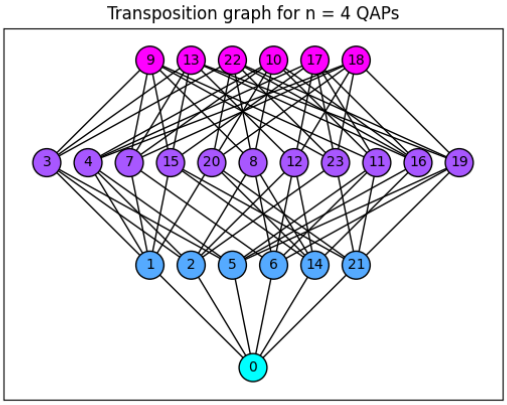

图2. $n = 4$ *QAP* 实例的换位图，以解 $0 = [0, 1, 2, 3]$ 为中心。节点代表各个解，边连接最近邻解，即仅相差一次对换的解。

III. 方法

本节详细介绍了用于执行 NV-QWOA 模拟、生成所有启发式算法的基准指标以及创建 QAP 实例的方法。由于在基准测试和比较过程中，模拟 NV-QWOA 运行受到并行计算限制，因此选择了较小的问题规模（$n = 5$，6、7、8、9和10）。关于模拟复杂度和运行时间的详细信息见第 IV 节。选择经典启发式算法进行比较，而非精确算法，尽管精确方法可用于较小的问题规模并保证找到最优解。这样选择是为了与采用类似非穷举方法的算法进行基准测试。与精确算法不同，启发式算法不会系统地探索整个解空间，也不会不断排除候选解，直至找到全局最优解。相反，它们通常会收敛到局部极小值，而这些局部极小值可能对应次优解。比较 NV-QWOA 与经典启发式算法的主要指标是

目标函数评估次数以及测量概率/求解概率。对于经典启发式算法，评估次数与执行时间直接相关，因为评估次数越多，所需处理也越多。NV-QWOA 的其他基准测试还包括分析所需电路深度 p 如何随 n 变化，以达到平均 10% 的最优解概率（OSP），并与 Grover 搜索进行比较；此外还考察不同问题规模下预期的平均节点间距离。

A. MMAS 配置

对于 MMAS 算法而言，其本身设计用于以启发式方式解决中等规模的实例，其效率高于最先进的精确算法，也能够处理远大于精确算法所能达到上限的问题规模。因此，已知将 MMAS 应用于较小的问题规模时效率较高。此外，MMAS 算法通常以主执行循环达到一定迭代次数或达到时间限制作为终止条件。由于时间限制与迭代次数和目标函数评估次数相关，因此需要考虑终止搜索所需的迭代次数；并且由于 MMAS 在为本文所考虑的小问题规模寻找实例解时效率更高，因此决定让该算法针对每个实例运行，直至找到最优解为止。

不计局部搜索时，目标函数评估次数等于找到最优解所需的 MMAS 迭代次数与算法所使用的蚂蚁数量之积： $N_{f(x)} = m * I$。为了更全面地了解 $N_{f(x)}$在不同问题规模下的统计分布，针对 30 个 QAP 实例分别运行算法 50 次，以计算平均值 $N_{f(x)}$ ，结果见图3。

在计算 MMAS 算法的目标函数评估次数时，每个问题实例的 50 次运行最初均按顺序计算，导致计算时间超出必要时长；随后采用 OpenMPI，让专用于较小运行子集的节点并行执行，从而显著缩短了计算时间。

B. NV-QWOA 配置

为了对比 NV-QWOA 与经典启发式算法在求解二次指派问题时的性能，需要了解拟设层数或电路深度 p如何随问题规模 n变化。为了实现高效的硬件实现， p应在 n中具有多项式上界。为确定合适的最优 p 值，以适当放大测得最优解的概率，我们采用 10% 的阈值，因为在进行 1000 次电路采样后未能测得最优解的概率约为 10^{-46}。1000 次独立的态制备和

算法执行后对放大的量子态进行测量 p次，其数量处于 NISQ 时代硬件量子电路评估的典型范围内 [3]。随后，最优的 p 值可用于确定目标函数评估次数，即 $s*p$，从而可直接与经典启发式算法进行比较。与 NV-QWOA 求解最大割问题的基准测试一致 [3]，，我们仅考虑 4 次测量，因此将 NV-QWOA 执行的目标函数评估次数定义为 $4p$。

$2p$ 变分参数 $\{\gamma_1, \gamma_2, ..., \gamma_p, t_1, t_2, ..., t_p\}$ 定义变分量子算法的参数被替换为 NV-QWOA 的 3 个可调参数 $\{\gamma, t, \beta\}$。这些参数定义了初始参数，它们作用于交替酉算符作用的第一层 $U_M(t)U_Q(\gamma)$，并通过公式10所示的线性渐变，在不进行微调的情况下构建完整的参数集。每一层都会跟踪这些参数值，并可观察 p 递增时各问题实例的演化情况，结果列于表I。

C. 解壳层结构

对于给定的问题规模 n，解壳层 k 包含与任意选定的参考解相距置换距离 k 的所有解，由包含 $n-k$ 个不相交循环的排列组成。这些不相交循环表示参考排列与目标排列之间元素映射构成的链，并在映射返回起始元素时终止。因此，l循环定义为在置换映射下经过 l 步返回初始元素的循环。

例如，以固定顶点 0123 为起点，考虑排列 2310，其不相交循环是一个 4-循环。我们从元素 0 开始执行映射 $\pi(0) \to 3, \pi(3) \to 1, \pi(1) \to 2, \pi(2) \to 0$。接着对元素 1,2, 3 重复这一过程，以确定该排列包含的不相交循环数量，从而确定其所在的解壳层。与固定顶点仅相差一次对换的解包含 $n-1$ 个不相交循环。此外，子壳层不仅由不相交循环的数量定义，也由所含循环的类型定义。不过，本文仅考虑规则壳层。

一个值得关注的问题是：在执行 NV-QWOA 前后，解壳层的概率如何变化。解壳层是根据与最优解的距离以及对换图连通性定义的。观察这种变化有助于全面了解随着问题规模增大，最优解和近优解如何得到放大，以及较差解如何受到抑制。

所有 30 个 QAP 实例均使用 Python 的 NumPy 包，通过均匀分布随机生成。先创建二维位置网格，再根据点之间的欧几里得距离初始化距离矩阵

。流量矩阵条目在随机初始化后乘以 20。为避免在 U_Q 中因构造性干涉和相消干涉而产生过度的相位旋转，会将编码后的成本缩放至 (0， 2π) 范围内，这是量子相位角的自然取值范围。实例由 Pawsey 超级计算研究中心的 Setonix 超级计算机系统生成；对于较大的问题规模（即 $n = 10$），使用 OpenMPI 并行化并运行少量任务，实例则保存为 .npy 文件。

IV. 结果与讨论

本节将考察通过运行 NV-QWOA 模拟以及应用经典启发式算法求解二次指派问题（QAP）所获得的结果。比较启发式算法时，一个重要的考量是，与精确算法不同，启发式算法无法保证找到全局最优解。因此，在实际应用中求解未见过的 QAP 实例时，平均情况和最坏情况性能指标能够体现这些启发式算法的质量，尤其是在力求以较高效率获得最优解或最优性间隙较小的近优解时。本文将先详细介绍模拟结果，再呈现经典和量子启发式算法的主要基准测试结果。

用于优化参数向量的优化器是 SciPy 中 minimize 函数所使用的 Nelder-Mead 算法 [47]。最初使用的是 COBYLA 算法，但它的收敛时间要长得多，而目标函数期望值最小化的效果却微乎其微，甚至没有改善。该 Nelder-Mead 优化器记录了每一层的目标函数评估次数，直至电路深度 p。

并行计算作业通过 SLURM 调度器 [48] 提交。使用 $-array$ 指令，对各问题规模中的每个实例执行模拟，并生成结果，直到达到合适的 p，以确保 NV-QWOA 在每个问题规模的各个实例上均返回平均 10% 的 OSP。每个问题规模的结果按顺序生成，所有优化器结果（包括 $\{\gamma, t, \beta\}$ 参数值和优化器目标函数评估次数）、迭代次数 p、目标函数值和模拟时间，均针对各问题规模中的每个问题实例存储在 CSV 文件中。

A. 平均模拟时间

在不同问题规模下，对 30 个随机生成的二次指派问题实例运行 NV-QWOA 所需的平均模拟时间 $n \in \{5, 6, 7, 8, 9, 10\}$以及优化后的参数值 $\{\gamma, t, \beta\}$ 及其对应的标准差汇总于表I。报告的标准差量化了模拟时间和参数估计的变化程度。

如表I所示，较小实例的模拟时间大致呈线性增长，但随着 n 增大，模拟时间会急剧上升，尽管分配的 MPI 任务有所增加。这一现象表明，经典模拟的计算成本会随问题规模呈指数级增长，凸显了在经典硬件上模拟量子算法的固有局限。此外，较大 n对应的模拟时间范围更宽（如标准差所示），进一步印证了这一观察结果，并凸显了资源限制对可扩展性的影响日益增大。

			模拟时间（秒）		参数值					
n	p	MPI	$\bar{t_s}$	σ_{t_s}	$\bar{\gamma}$	σ_γ	$\bar{\beta}$	σ_β	$\bar{t}$	σ_t
5	3	16	1.23	0.10	1.5197	0.9714	0.4533	0.0990	0.3153	0.0423
6	6	16	4.05	0.29	1.8081	1.9879	0.3623	0.0808	0.2155	0.0258
7	10	16	15.52	0.61	1.5178	0.1942	0.3339	0.0659	0.1649	0.0131
8	17	16	164.68	6.95	1.4777	0.2040	0.3027	0.0656	0.1367	0.0216
9	28	32	2878.51	233.34	1.3902	0.2785	0.3127	0.0916	0.1075	0.0270
10	49	64	96659.14	7519.34	1.3976	0.2591	0.2630	0.0939	0.0948	0.0200

表 I。*NV-QWOA* 运行在问题规模 5、6、7、8、9、10时的平均模拟时间和带标准差的平均参数值

每种问题规模对应的平均优化参数值也记录在表 I 中。 γ 在 $n = 6$ 实例中的较大均值和标准差归因于某个单独实例，该实例所识别区域中的 γ 比预期的 $\gamma \approx 1$大了一个数量级，因此导致 β 和 t的值较小。然而， β 和 $\bar{t}$ 会随着 n 按比例减小，符合预期；此外， $\bar{\gamma}$ 仍处于 $\gamma \approx 1$的数量级，与 [2] 中的发现一致。各实例的目标函数评估次数总和的平均值（向上取整）分别为 645、1199、1902、3134、5030 和 8681，对应于每个n；这些值与问题规模增长时 p 的增长呈线性正比关系。如第IVB节所示，目标函数评估次数反映了p随 n呈现的多项式尺度增长。

B. 目标函数评估次数

MMAS 的目标函数评估次数平均值 $N_{f(x)} = m \cdot I$随问题规模 n 变化的情况如图 3所示。平均情况和最坏情况下的评估次数显然都随问题规模呈指数增长；每当 n线性增加时，评估次数便大约增加一个数量级。鉴于解空间呈阶乘增长，这种表现是意料之中的。随着 n 增加，固定数量的蚂蚁 m 与 $n!$ 个可能解之比急剧下降；因此，需要大幅增加 I 来弥补。由于所需的 I 增幅远远超过恒定的 m， $N_{f(x)}$ 会急剧上升。

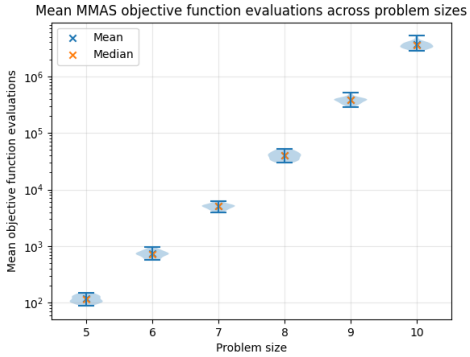

图 3。 *MMAS* 算法在问题规模5、6、7、8、9、10下的目标函数评估次数平均值

在所有问题规模下，GLS 的目标函数评估次数平均值都显著高于 NV-QWOA，如图 4所示。GLS 的设计是从邻近解中选取改进效果最佳的交换；然而，它既不考虑全局成本景观，也不会提前结束与邻近解的比较。这些特点导致目标函数评估次数较高。可通过改进其搜索过程来缓解这一问题，例如引入随机化，偶尔接受较差的解，以避免收敛到局部极小值。

然而，此类修改通过引入旨在逃离局部极小值并提升搜索质量的复杂机制，带来了额外的复杂性。这些增强功能的实现通常会导致更多的目标函数评估次数，这一点在突破局部搜索[49] 和禁忌搜索 [26] 中已有体现。因此，尽管标准 GLS 更有可能在局部最小值处终止，但与其更复杂的变体相比，在评估总次数方面仍然更加高效。

随着问题规模增大，GLS 与 NV-QWOA 之间的评估次数差异也随之增加。这表明，该量子启发式算法的评估次数增长率较慢，说明在所研究的较小问题规模范围内呈现亚指数级扩展。总而言之，与 MMAS 和 GLS 启发式算法相比，NV-QWOA 在目标函数评估次数方面实现了更优的扩展性。

C. 求解与测量概率

图中展示了不同问题规模下 GLS 和 NV-QWOA 的求解概率与测量概率4。该图包含一个代表理想情况的基准线：在准备并测量放大态四次时，NV-QWOA 对每个实例都保持一致的 10% 成功概率（OSP）

。误差线表明，与 GLS 相比，NV-QWOA 结果的方差显著更大，这说明 GLS 在相同问题规模的不同实例中的性能更加稳定。

对于问题规模为 $n \in \{5, 6\}$ 的情况，GLS 的求解概率最初显著高于 NV-QWOA。然而，正如预期，NV-QWOA 在所有问题规模下始终保持 10% 的 OSP，而 GLS 的求解概率则随着 n 的增加呈指数级衰减。这种衰减是局部搜索启发式的典型特征；随着局部极小值的数量随问题规模呈指数级增长，贪心搜索收敛到全局最优解的概率也相应降低。此外，小型 n 下较高的初始求解概率可归因于受限的解空间：在这种情况下，随机初始排列只需经过几次交换就能到达全局最小值的可能性更高。

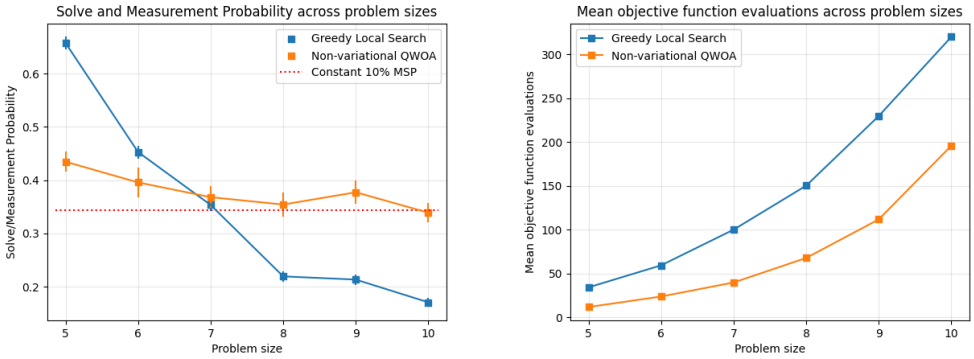

图 4. *GLS* 与 *NV-QWOA* 的直接比较。左：*GLS* 和 *NV-QWOA* 在问题规模为 5、6、7、8、9、10 时的求解概率和测量概率。右：*GLS* 和 *NV-QWOA* 在问题规模为 5、6、7、8、9、10 时的目标函数评估次数均值

D. 问题规模所需的迭代次数

图中以小提琴图展示了每种问题规模下生成的 30 个随机 QAP 实例，在 p 交错幺正算子深度不同的情况下，其 OSP 分布5，并给出了各离散 p 值对应分布的均值和中位数。图中还显示了 OSP 为 10% 的阈值线，以便找出在给定问题规模下平均情况下使平均 OSP 达到 p 的最佳整数 $> 10\%$ 值。

对于 $n = 5$ 最多 $n = 9$个实例，随着 p 值增大，分布按照算法预期平滑地向上增长；对于较大的问题规模，观测到的最大 OSP 出现了一定程度的趋平，其中 $n = 9$ 实例最为明显。对于 $n = 10$ 实例，最大 OSP 在 $p = 25$ 附近的趋平现象更加明显，随后在 $p = 40$处观察到急剧增加，并且随着 p 增大，该最大 OSP 继续增加。观察这一转变点附近的实例，

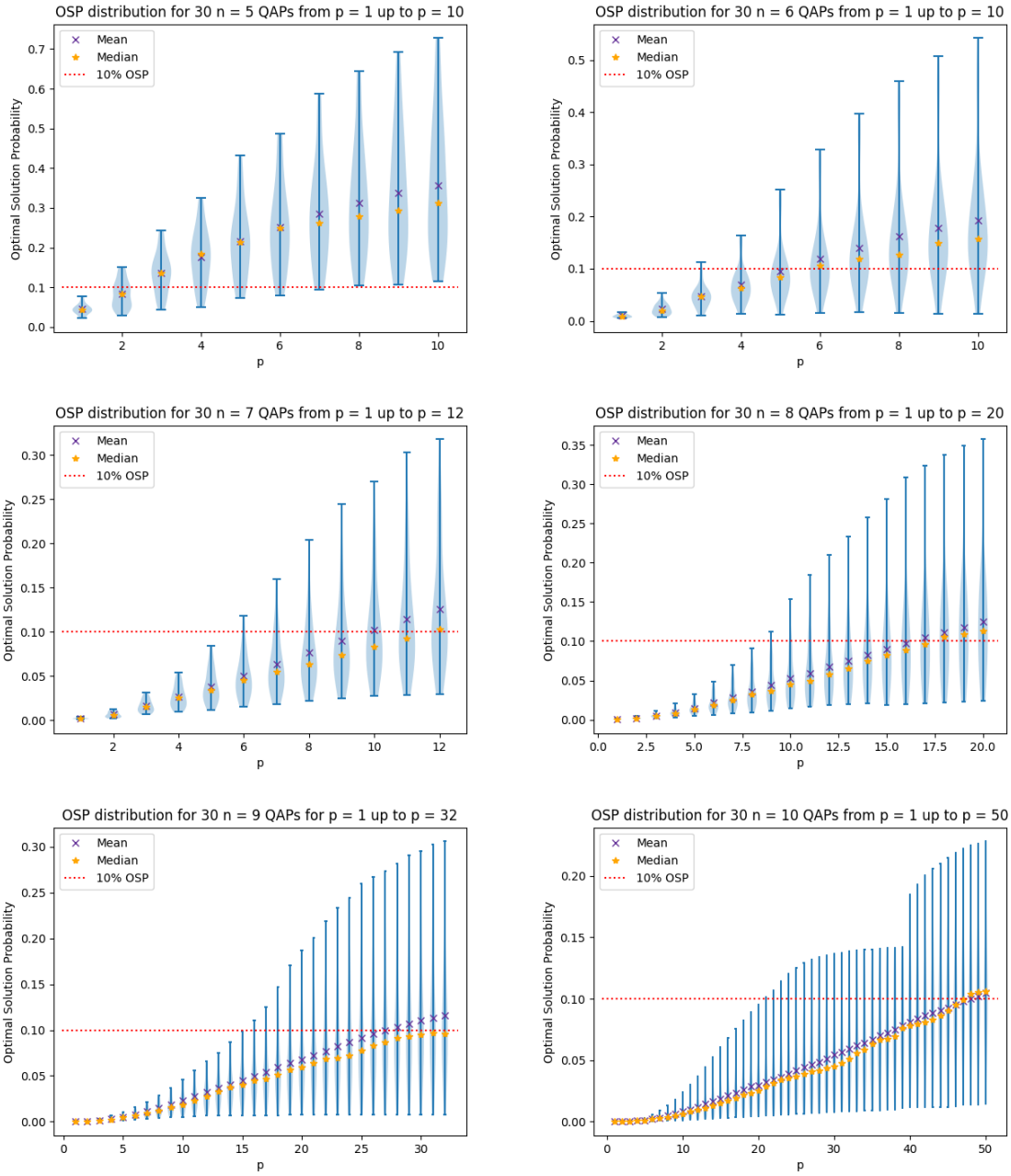

图 5. 不同 p 值范围内的最优解概率小提琴图

可以发现，最大 OSP 并不对应于单个实例，而是对应于多个实例。这些实例的参数值发生了显著变化，表明优化器在景观中找到了成本更低的邻近区域。

随后，我们将 NV-QWOA 所需的交替酉算子应用次数，与平均情况下将 OSP 放大至 10% 所需的 Grover 搜索迭代次数进行比较，如图 6 所示。对于较小的 $n = 5$，6，两种启发式算法最初的迭代次数相近，但在 $n = 7$ 之后，所需旋转次数便开始远远超过 NV-QWOA 所需的次数，且差距呈指数级增长。由于 Grover迭代 的次数由 $p = \sqrt{N}_{\arcsin(\sqrt{0.1})_2}{}^{-1}$ 决定，因此达到 10% OSP 所需的迭代次数约为 $O(\sqrt{n!})$，这也解释了这种指数级增长。我们

观察到，NV-QWOA 的最优整数迭代次数 p = 3 分别为 6、10、17、28、49。快速曲线拟合表明，这一最优迭代次数随 $p = 0.0049n^4$ 呈四次方增长，其与问题规模的关系为 $O(n^4)$，这符合我们对 NV-QWOA 在 QAP 实例上实现多项式尺度增长的期望。

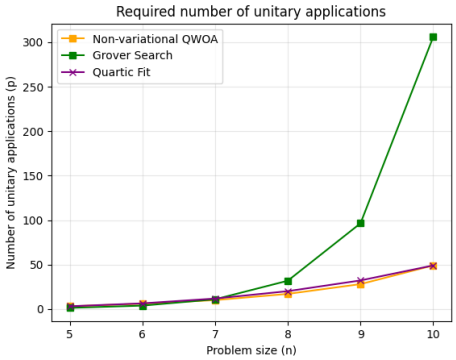

图 6。 *NV-QWOA* 所需的迭代次数 p 与 *Grover* 搜索进行比较，问题规模为 $n = 5$、6、7、8、9、10

E. 解壳层概率和节点间距离

现在，我们将考察在执行 NV-QWOA 后，对于不同问题实例，在最优解中给定固定顶点的情况下，QAP 解在各个壳层中的分布。解壳层概率定义为给定壳层中所有解的概率之和，即与最优解相距某个距离的解的概率之和；该距离由对换次数（排列元素交换次数）表示。节点间距离是指一个解与最优解之间的距离。

在应用交错酉算子之前，所有解的初始概率均相等，这是因为初始量子态是所有可能解状态的等幅叠加，其中每个解状态按照方程 6进行编码。在考虑与最优解之间的距离以及每个壳层内的解的数量后，每个壳层的初始概率之和就是该壳层内的解的数量。壳层中的解是具有 $n - k$ 不相交循环的排列。图 7 给出了各壳层初始概率之和的绘图7。

显然，最优解概率（壳层距离 $= 0$）按照初始概率 $1/n!$，以 $1/n$ 的幅度衰减；无论问题实例如何，NV-QWOA 都需要大幅提升这些较早壳层的概率。对于 $n = 9$ 情况，提升幅度为 10^4 倍，足以将平均情况下的 OSP 提升到更易测量的范围。

我们预计，NV-QWOA 会显著增强最优解和近优解，同时降低各问题规模下低质量解的概率，但对于更大的

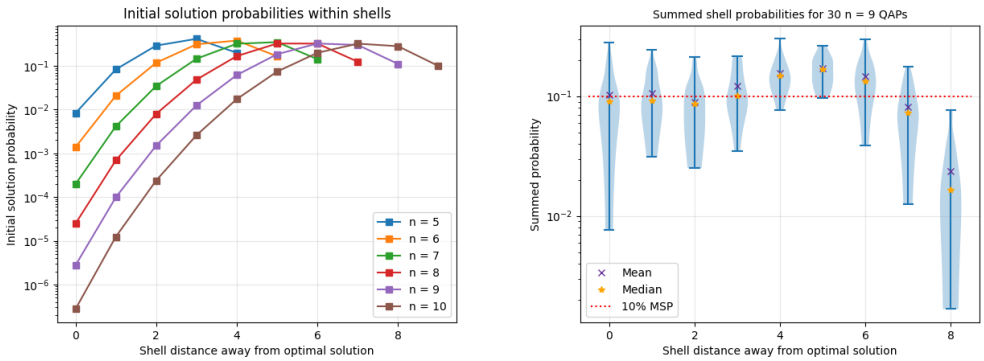

图 7. 左：在问题规模为5、6、7、8、9、10时，考虑与最优解的壳层距离后，初始概率之和。右：所有 $30\ n = 9$ *QAP* 实例的最终概率之和

问题规模，需要更强的效果。图8 展示了在所关注的问题规模下，30 个实例中与最优解之间节点间距离的预期均值。该预期均值反映了执行 NV-QWOA 并达到给定问题规模所需的 p 电路深度后，我们预期测得的解与最优解之间的距离。理想情况下，预期均值应为 0，且不受问题规模影响，从而保证在任何问题规模下仅通过一次测量就能找到最优解。然而，由于连续时间量子行走（CTQWs）引起的解状态间振幅概率转移和相位偏移具有随机性，并由此带来复杂的相位相互作用，NV-QWOA 本质上是一种启发式算法。每次迭代中，最优解和近优解并不会始终以相同比例得到增强；要使 OSP 稳定达到 10%，就需要在阶乘级缩放下实现更强的增强。所得预期均值显示其随问题规模线性增长；与问题规模增大时层数的增长相比，这一结果令人鼓舞。由于层数随问题规模线性增加，对于最差解，预期节点间距离最初要大得多。

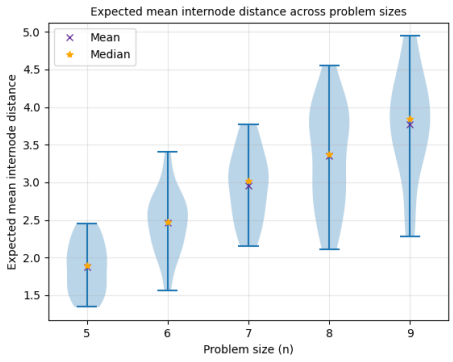

图 8. 问题规模为 $n = 5$、6、7、8、9时节点间距离的期望均值

F. 讨论

我们的研究结果表明，在目标函数评估次数和测量/求解概率方面，NV-QWOA 在 30 个随机生成的小规模二次指派问题实例中，其扩展性可与已知的高效经典启发式算法相媲美。如图6所示，交替酉算子应用的次数随问题规模呈二次增长，优于为在平均情况下将 OSP 放大至 10% 所需的 Grover 迭代次数的增长趋势，后者按 $O(\sqrt{n!})$ 增长。这一结果令人鼓舞，因为 p 随 NV-QWOA 的电路复杂度线性增长，因此对于小规模二次指派问题实例，整体电路复杂度随时间呈多项式增长。此外，鉴于 Grover 搜索相较于穷举搜索可实现二次加速，这至少表明，NV-QWOA 在求解小规模二次指派问题实例时可能实现超过二次的加速。

在比较量子启发式算法时，仅考虑了一项指标，且模型中未纳入量子噪声。由于缺乏噪声建模，模拟重点考察了容错量子设备上的理想量子算法性能；这是因为有适用于组合优化、且兼容在 Setonix 上并行计算的优化软件可供使用。对理想性能的关注与当前 NISQ 时代硬件的现状并不相符；当前硬件需要围绕噪声缓解进行算法设计，因此，这些结果对量子启发式算法在现实世界中的实际应用仅有间接影响。不过，模拟结果仍可作为重要的理论基准，为之后考虑退相干和含噪系统时的实际设计提供参考。

由于并行计算在对量子算法进行经典哈密顿量模拟时存在复杂性，模拟较大问题实例受到若干限制。较长的模拟时间使高效收集这些结果变得困难；预计随着问题规模增大，模拟时间将继续快速增长，因此需要对代码进行深入分析、更充分地利用混合图的并行性，甚至采用 GPU加速运行来探查计算瓶颈，并据此实现相应的扩展。

由于问题规模较小，人们不禁要问，NV-QWOA 得出的结果是否适用于更大的问题规模，尤其是在 $n > 30$ 精确算法的计算开销过高、因而在实际应用中难以处理，需要采用启发式方法的情形下。在针对 QAOA 参数优化初始化的相关研究中，几位研究者发现，可以将较小问题实例的参数外推并用于更大的问题 [50]，从而大幅缩短处理此类规模问题所需的计算时间，并为了解这一情形下启发式算法的性能提供了思路。

V. 结论与未来工作

本文探讨了 NV-QWOA 求解 QAP 的性能，并将其表现与两种经典启发式算法——最大-最小蚁群系统（MMAS）和贪心局部搜索（GLS）进行了比较；此外，还简要比较了 Grover 搜索算法，并考察了将最优解概率（OSP）放大至 10% 所需的算法迭代次数，以便通过多次测量近乎可靠地获得最优解。我们随机生成了 30 个 QAP 实例，规模为， 6、7、8、9、10，用于构成启发式算法的待解实例集。NV-QWOA 在该实例集上的表现显示，电路深度随较小问题规模呈多项式增长，且与这些启发式算法相比具有竞争力。

采用三项指标，对不同成对比较下启发式算法的性能进行定性和定量评估：MMAS 和 GLS 相较于 NV-QWOA 的目标函数评估次数、GLS 相较于 NV-QWOA 的测量/求解概率，以及 Grover 搜索相较于 NV-QWOA 所需的算法迭代次数。

未来工作中，值得研究这些趋势是否会在问题规模超过 n =30 的更大 QAP 实例中持续存在，以确定量子启发式算法能否更高效地求解 NP 困难问题。对于此类问题规模，存在硬件方面的顾虑：鉴于本文所研究的较小实例所需的计算时间，以及解空间按 n阶乘级增长，并行计算开销将十分巨大。不过，预计多项式尺度增长的结果在更大规模下仍将持续，因为 NV-QWOA 的干涉效应依赖于统计假设，而这些假设在此类规模下会越来越成立 [2], ；同时，人们也越来越关注如何开发可靠的参数迁移方案，以便将从小型实例中获得的参数值用于经典计算难以处理的问题规模。

致谢

本项目由澳大利亚政府通过关键技术挑战计划（CTCP）和先进战略能力加速器（ASCA）提供支持。Pawsey 超级计算研究中心和 Acacia 对象存储提供了大量计算资源。作者感谢 Tavis Bennett 就量子优化方法进行宝贵的讨论，也感谢 Edric Matwiejew 使用 QuOp MPI 软件包协助实现模拟代码 [51, 52], 并就量子组合优化展开了富有成效的讨论。

-

[1] T. Bennett & L. Noakes & J. B. Wang. 非变分量子组合优化。载于 arXiv:2404.03167 和 2024 IEEE 量子计算与工程国际会议（QCE），第1卷，第31–41页。IEEE，2024年。[2] T. Bennett & L. Noakes & J. B. Wang. 非变分量子游走优化算法分析。arXiv:2408.06368，2024年。[3] T. Bennett & A. Smith & E. Matwiejew & J. B Wang. 量子启发式算法基准测试：用于加权最大割的非变分 QWOA。Physical Review A（即将发表），2026年。[4] T. St¨utzle & H. H. Hoos. 最大–最小蚁群系统。《未来一代计算机系统》，16(8):889–914，2000年。[5] J. Christiansen & K. Smith-Miles. 二次指派问题的实例空间分析。arXiv:2506.20172，2025年。[6] L. K. Grover. 量子力学助力在大海捞针中搜索。Physical Review Letters，79(2):325–328，1997年。[7] S. Sahni & T. Gonzalez. P完全近似问题。《美国计算机协会学报》，23(3):555–565，1976年。[8] A. N. Elshafei. 作为二次指派问题的医院布局。《运筹学季刊》，28(1):167–179，1977年。[9] A. M. Geoffrion & G. W. Graves. 考虑转换成本的并行生产线调度：二次指派／线性规划方法的实际应用。《运筹学》，24(4):595–610，1976年。[10] L. Steinberg. 背板布线问题：一种布局算法。《SIAM评论》，3(1):37–50，1961年。[11] QAPLIB 2012。[12] L. K. Grover. 一种用于数据库搜索的快速量子力学算法。载于第28届ACM计算理论年度研讨会论文集（STOC），第212–219页。ACM，1996年。[13] T. C. Koopmans & M. Beckmann. 指派问题与经济活动选址。Econometrica，25(1):53–76，1957年。[14] P. C. Gilmore. 二次指派问题的最优与次优算法。《工业与应用数学学会学报》，10:305–313，1962年。[15] E. L. Lawler. 二次指派问题。《管理科学》，9(4):586–599，1963年。[16] J. W. Gavett & N. V. Plyter. 通过分支定界实现设施与地点的最优指派。《运筹学》，14(2):210–232，1966年。[17] M. S. Bazaraa & H. D. Sherali. 将 Benders分解方案应用于二次指派问题的一种新形式。《海军研究后勤季刊》，27(1):29–41，1980年。[18] M. W. Padberg & G. Rinaldi. 一种用于求解大规模对称旅行商问题的分支割算法。《SIAM评论》，33:60–100，1991年。[19] J. Peng & H. Mittelmann & X. Li. 一种基于矩阵分裂的二次指派问题新松弛框架。《数学规划计算》，2:59–77，2010年。[20] E. D. Taillard. 用于二次指派问题的鲁棒禁忌搜索。《并行计算》，17:443–455，1991年。

[21] E. D. Taillard。《二次指派问题迭代搜索方法的比较》。Location Science，3：87–105，1995年。[22] K. M. Anstreicher 等。《在计算网格上求解大型二次指派问题》。Math. Program. Ser. B，91：563–588，2002年。[23] P. Merz 与 B. Freisleben。《无约束二元二次规划的贪婪启发式算法与局部搜索启发式》。Journal of Heuristics，8：197–213，2002年。[24] H. R. Louren¸co、O. Martin 与 T. St¨utzle。《迭代局部搜索：框架与应用》，第146卷，第363–397页。Kluwer Academic Publishers，2010年。[25]S. Kirkpatrick、C. D. Gelatt 与 M. P. Vecchi。《通过模拟退火进行优化》。Science，220(4598)：671–681，1983年。[26] F. Glover。《整数规划的未来路径及其与人工智能的联系》。Computers & Operations Research，13(5)：533–549，1986年。[27] M. Dorigo。《优化、学习与自然算法》。博士论文，米兰理工大学，意大利，1992年。[28] T. Bartz-Beielstein、J. Branke、J. Mehnen 与 O. Mersmann。《进化算法》。Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery，4，2014年。[29] P. Moscato。《论进化、搜索、优化、遗传算法与武术——迈向模因算法》。加州理工学院并发计算项目，2000年。[30] E. Farhi、J. Goldstone、S. Gutmann 与 M. Sipser。《通过绝热演化进行量子计算》。arXiv：quant-ph/0001106，2000年。[31] E. Farhi 等。《应用于随机NP完全问题实例的量子绝热演化算法》。Science，292(5516)：472–475，2001年。[32] E. Farhi、J. Goldstone 与 S. Gutmann。《量子近似优化算法》。arXiv：1411.4028，2014年。[33] L. Bittel 与 M. Kliesch。《训练变分量子算法是NP困难的》。Phys. Rev. Lett.，127：120502，2021年。[34] M. Larocca 等。《变分量子计算中的贫瘠高原综述》。arXiv：2405.00781，2024年。[35] S. Marsh 与 J. B. Wang。《一种用于有界NP优化问题的量子游走辅助近似算法》。Quantum Information Processing，18(3)：61，2019年。[36] S. Marsh 与 J. B. Wang。《通过高效量子游走进行组合优化》。Physical Review Research，2(2)，2020年。[37] S. Marsh 与 J. B. Wang。《使用交替量子游走的确定性空间搜索》。Physical Review A，104：022216，2021年。[38] K. Manouchehri 与 J. B. Wang。《量子游走的物理实现》。Springer，2014年。[39] T. Loke 与 J. B. Wang。《复合图上连续时间量子游走的高效量子电路》。Journal of Physics A: Mathematical and Theoretical，50(5)：055303，2017年。[40] T. Bennett、E. Matwiejew、S. Marsh 与 J. B. Wang。《基于量子游走的车辆路径优化》。Frontiers in Physics，9：730856，2021年。[41] N. Slate、E. Matwiejew、S. Marsh 与 J. B. Wang。《基于量子游走的投资组合优化》。Quantum，5：513，2021年。[42] E. Matwiejew、J. Pye 与 J. B. Wang。《通过结构化搜索对连续多变量函数进行量子优化》。Quantum Science and Technology，8(4)：045013，2023年。

[43] D. Qu 等。《量子游走投资组合优化的实验实现》。Quantum Sci. Technol.，9：025014，2024年。[44] E. M. Loiola 等。《二次指派问题综述》。《欧洲运筹学杂志》，176：657–690，2007年。[45] V. Maniezzo、M. Dorigo 与 A. Colorni。《应用于二次指派问题的蚁群系统》。技术报告，比利时布鲁塞尔大学，1994年。´S. Hadfield 等。《从量子近似优化算法到量子交替算子拟设》。《算法》，12(2)：34，2019年。[46] E. Jones、T. Oliphant 与 P. Peterson。《SciPy：用于 Python 的开源科学工具》，2001年。[47] M. A. Jette 与 T. Wickberg。《Slurm 工作负载管理器的架构》。载于 Klus[48] aˇcek、D.、Corbal´an、J.、Rodrigo、G.P.（编）：《并行处理作业调度策略》。JSSPP 2023。《计算机科学讲义》，第14283卷。Springer，Cham，2023年。´U. Benlic 与 J. Hao。《用于二次指派问题的突破局部搜索》。Applied Mathematics and Computation，219(9)：4800–4815，2013年。[49] A. Galda 等。《随机图之间最优QAOA参数的可迁移性》。arXiv：2106.07531，2021年。[50] E. Matwiejew 与 J. B. Wang。《Quop mpi：用于并行模拟量子变分算法的框架》。《计算科学杂志》，62：101711，2022年。[51] E. Matwiejew。《Quop mpi v.1.2.1——https://github.com/edric-matwiejew/quop mpi》，2025年。