arXiv:2603.03613v1 [stat.ML] 4 Mar 2026

基于置换的优化中无免费午餐违背的实证评估

Grzegorz Sroka

热舒夫科技大学非线性分析系，波兰热舒夫 *Powstańców Warszawy 12*，邮编 *35-959* 电子邮箱：*gsroka@prz.edu.pl*

摘要。无免费午餐（*NFL*）定理保证：只有在函数空间上进行均匀采样，且该函数空间在排列下封闭（*c.u.p.*）时，平均性能才相等。我们探讨这种平均结果何时不再能反映基准测试的实际报告结果。我们研究一种采用无放回抽样的迭代搜索设置，其中算法之间的差异仅在于评估顺序。对于二元目标函数，在可完全枚举的情况下可以进行穷举评估，而效率则定义为首次达到全局最小值所需的时间。随后，我们通过对相同基准函数进行加法和差运算的代数重组，构造另外两个基准测试。我们通过相关结构、层次聚类、差值热图和主成分分析，考察函数-算法关系。单因素方差分析与 *Tukey* 对比证实，代数重构会引起具有统计学意义的性能模式变化。均匀抽样的基线仍与全局 *NFL* 对称性一致。相比之下，经代数变换的基准测试会产生稳定的排名变化，并形成由函数和抽样策略构成的连贯簇。由较简单分量构建的复合目标也可能呈现非加性的搜索开销。蒙特卡洛实验表明，在更大的空间中，顺序效应依然存在，且取决于函数类。综上，结果表明，目标函数重构和基准设计能够产生结构化的局部偏离，使实际情况不同于 *NFL* 直觉。这些发现提示，算法选择应同时考虑问题类和目标表示。这一结论既适用于进化计算，也适用于基于重新标记、重抽样和置换检验的统计程序。

1 引言

实际中的优化是在有限离散机器上进行的。在这些条件下，NFL定理可能并不能提供有效的指导。性能可能取决于受限问题类中的结构，而非对所有函数的完整空间求平均。

是否存在一种通用优化算法，能够在所有可能的问题上表现同样出色？这个问题是优化领域最引人入胜的结果之一——NFL定理的核心。NFL定理于1995年提出，它指出，若在所有可能问题的整个空间上进行评估，那么所有优化算法的平均表现都相同。这一颇具争议的主张引发了广泛的辩论与批评。

在他的著作中， [30], Goldberg认为，传统优化方法只对一小类问题有效，而进化算法等稳健技术则能在更广泛的问题范围内取得良好结果。然而，一份具有开创性的1995年报告的作者[88] 对此提出反驳，指出对所有可能的问题求平均时，没有任何算法会优于其他算法。NFL定理也被用于更广泛的争论。例如，Dembski [20] 在有关智能设计的论证中援引了NFL框架。

正如 Clerc 所观察到的， [16], 计算是有限且离散的。这一观点直接启发了我们的研究。因此，研究受限的函数类别（包括那些未在排列下封闭的函数（not-c.u.p.））是顺理成章的：

> 在计算机上，一切都是离散有限的，因此我们更仔细地考察非 c.u.p. 函数集。看来，除了非常特殊的情况，无免费午餐定理不仅不成立，平均而言也不仅存在最优算法，甚至还确实存在一种“终极”算法。

这一视角促使我们关注结构化子集：在这些子集中，NFL式平均可能失效，算法之间的性能差异也可能持续存在。

本文旨在重新审视NFL定理，重点关注该定理不成立的特定条件。通过分析在排列和算术运算下封闭的二元函数子集，我们证明某些算法组的效率存在差异，从而对NFL定理的普适性提出质疑。

本研究聚焦于基于排列的优化算法，因为这类算法广泛适用于旅行商问题、调度、指派问题和路径规划等组合优化问题。这些算法利用解空间中的对称性，广泛应用于进化计算和启发式优化。由于排列空间是有限且离散的，因此它为研究贴近实践情境下NFL定理的理论影响提供了明确的框架。

为了在二元目标函数的完整空间中对算法性能进行受控且穷尽的研究，我们将分析限定在 $n = 4$的情形，此时可以完整枚举所有$4! = 24$ 排列及其对应的 2^{24} 二元函数。这样的维度允许我们在没有抽样误差的情况下精确测量算法效率，并准确验证排列封闭等结构性质。对于更大的维度，由于函数空间迅速增长，穷尽分析将变得不可行。对于 $n = 5$，空间规模从 2^{24} 增长到 2^{120}, ，即增长了 $2^{96} \approx 10^{28}$倍。对于$n = 10$，排列数量为$10! = 3628800$，完整函数空间包含$2^{10!} = 2^{3628800}$ 个二元函数，即使是高性能计算系统也难以处理。

应当指出，即使对于 $n = 4$ ，计算也可能十分繁琐。对于大得多的定义域，例如 $2^{30}, 2^{50},$ 或 $2^{100},$ ，人工检验实际上是不可能的，而且需要大量计算资源，包括处理能力和内存。在这种情况下，更有效的方法是采用统计方法，并分析具有特定特征的二元样本。例如，可以选择输出向量高度不平衡的函数，例如至少 91% 的位均为“1”或“0”。也可以选择二进制字符串中初始位置有 91% 为“1”的序列。这些采样规则有助于研究高维情况，并提高分析效率。

因此， $n = 4$ 在可处理性与分析严谨性之间取得了平衡，适合用于揭示无免费午餐定理的假设可能不再成立的条件。未来研究计划采用统计抽样扩展至更高维度，这超出了本文的研究范围。

本研究的方法选择旨在确保结果清晰、严谨且易于直接解读。将陪域限制为 $\{0, 1\}$ 是一种有意采用的分析简化方法，使我们能够完整枚举函数空间（针对较小的问题规模），并分离出与NFL定理相关的结构性质。这遵循了NFL文献中广为确立的做法（例如，Ho和Pepyne [36]），即将二值函数用作最精简而又具有代表性的模型。重要的是，研究结论也适用于更广泛的陪域，因为观察到的效应取决于问题结构，而非取值范围。同样，无放回抽样的假设反映了NFL的原始框架（Wolpert和Macready [88]）。这确保每个评估步骤都能提供新信息，并允许公平比较信息高效策略。尽管这一假设经过理想化处理，但它与许多实际元启发式算法中的机制一致（例如，禁忌搜索、进化算法中的重复项消除），这些机制会避免或尽量减少重复访问解。采用这些设计决策后，我们排除了与问题结构无关的干扰，并聚焦于影响算法性能的关键因素。研究结果证实，当问题类

偏离完全随机的情况时，即使在信息效率最高的条件下，结构规律性也能系统性地使某些算法更具优势。

方法论。 本研究分析了24种基于排列的算法在一组排列封闭（c.u.p.）的16个二元函数上的效率。研究采用相关矩阵、层次聚类和主成分分析等统计分析方法，评估函数的代数变换对优化的影响。研究确定了无免费午餐定理不成立的条件，并强调函数结构在优化算法效果差异中的作用。

2 相关工作

我们简要回顾了排列优化领域的近期进展，这些研究利用排列空间的几何、代数和概率结构。

近期关于排列问题的研究强调，必须根据这一搜索空间的特定结构定制算子和模型。Moraglio 和 Poli [51]提出了用于排列表示的几何交叉概念，该方法基于解空间中定义的度量，并将现有重组算子统一到一个共同的几何框架中。Santucci 等人 [69],[70]提出了一种代数方法，将排列集合视为具有群结构的集合，从而通过定义排列上的相应操作，将连续算法（例如差分进化和粒子群优化）迁移到离散定义域。从概率视角出发，Ceberio 等人 [12] 分析了在估计分布算法中的Mallows模型和广义Mallows模型里，排列距离度量（例如 Kendall、Cayley、Ulam）的选择所产生的影响；结果表明，恰当的距离度量会显著影响算法性能，并使适应度景观更加平滑。Santucci 和 Ceberio [68]提出了一种用于估计分布算法的双随机矩阵（DSM）模型，以求解二次指派问题（QAP）；该模型能更有效地捕捉解结构，且优于早期概率模型。Ceberio 和 Santucci [11]则提出了一种基于模型的梯度搜索方法，可降低计算开销并提高排列问题的可扩展性。Doerr 等人 [24] 对排列空间中的进化算法进行了正式的运行时间分析，将经典基准问题（如前导一和 跳跃）改编到这一领域，并证明变异难度不仅取决于汉明距离，还取决于给定排列的循环结构。这些研究涵盖广义度量空间、概率模型、几何算子和代数形式体系，与本研究采用的方法一致，或可视为其自然延伸，表明所提出的方法符合排列优化算法领域的当前研究方向。

总体而言，这些研究为我们针对结构化基准测试类别开展的排列算法受控评估奠定了基础。

3 符号说明

设 $\mathcal{X}$ 是一个离散状态集， $\mathcal{Y}$ 是一个可通过映射 $f\colon \mathcal{X} \to \mathcal{Y}$ 赋给这些状态的离散值集。我们假设这两个集合都是有限的，并且在优化语境下， f 是代价函数，也称为目标函数、能量函数等。特定函数 f 对应一个特定的优化问题，而集合 $\mathcal{F} = \mathcal{Y}^{\mathcal{X}}$ 是所有可能优化问题构成的空间。特别是，当 $\mathcal{Y} = \{0,1\}$ 时，集合 $\mathcal{F}$ 由 $2^{|\mathcal{X}|}$ 个二元函数组成。

示例 1 令集合 $\mathcal{X} = \{1, 2, 3, 4\}$ 和 $\mathcal{Y} = \{0, 1\}$ 表示值集。那么 $|\mathcal{X}| = 4$. $|\mathcal{Y}| = 2$，我们有 $|\mathcal{F}| = m^n = 2^4 = 16$ 个可能的函数: $\mathcal{X} \to \mathcal{Y}$，它们由其在集合 $\mathcal{X}$ 上的取值定义（见表 *1*）。

在本例中，我们固定 $n = 4$ 以表示输入维度，即定义域 $\mathcal{X}$中元素的个数，并用 $m = 2$ 表示陪域 $\mathcal{Y}$的基数。这些函数完全由其在 $\mathcal{X}$的每个元素上的输出值所确定，而表 **1**中的输入字符串表示长度为 $n = 4$的所有二进制组合。

在 $\mathcal{X}$上的值 / $f \in \mathcal{F}$	$f1$	$f2$	$f3$	$f4$	$f5$	$f6$	$f7$	$f8$	$f9$	$f10$	$f11$	$f12$	$f13$	$f14$	$f15$	$f16$
x_1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
x_2	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

表 1：16个准备函数。

上表中使用的字符串应按如下方式理解：对于每个函数$f \in \mathcal{F}$，我们有：函数 fi 的值是数 $i - 1$的二进制展开，按逆位序书写（即最低有效位在前）。例如， $f4$**(1)** $= 1$， $f4$ **(2)** $= 1$， $f4$**(3)** $= 0$， $f4$**(4)** $= 0$。二元函数的广泛应用已在多篇论文中得到讨论，其中包括： [36], [67], [93], [94]。

我们通过求解问题 f 所需的目标函数评估次数来衡量算法的效率，该次数对应达到预定解质量所需的评估次数。我们假设会评估空间中的不同 点 $\mathcal{X}$。搜索结果是序列

$$T_m = \{< x_1, f(x_1) >, < x_2, f(x_2) >, \ldots, < x_m, f(x_m) >\}$$

其中 $1 \leq x_i \leq m$，对于 $m \in \{1, ..., |\mathcal{X}|\}$（$|\mathcal{X}|$ 此处表示集合 $\mathcal{X}$的大小）， $x_i \in \mathcal{X}$和f（x_i）$= y_i \in \mathcal{Y}$ [30]， [77]。

为了定义优化算法的概念，我们将介绍有关采样原则的定义。

在优化函数 f 时： $\mathcal{X} \to \mathcal{Y}$ 可以假设集合 $\mathcal{X}$ 和 $\mathcal{Y}$ 已知，但只知道少数特定值 $y = f$（x）。那么，关于 f 的部分信息可以用函数 d： $\mathcal{X} \to \mathcal{Y} \cup \{?\}$ 来描述，并称为数据。符号“?”表示 f（x）的未知值。对于所有已知值 $y = f$（x）的 x，我们得到d（x）$= f$（x），对于其余的 x，我们得到 d（x）$= ?$

令 $\mathcal{D}$为函数在 $\mathcal{F}$中的所有可能“数据”的集合。因此 $|\mathcal{D}| = (m+1)^n$，且对于每个 $x \in \mathcal{X}$， $d(x) = ?$ 这表示“没有数据”，即我们对代价函数一无所知的情形。

定义 1（采样策略） *[43]* 采样策略 s 是一个函数 s： $\mathcal{D} \to \mathcal{X}$，其中 $\mathcal{D}$ 是所有可能数据的集合（如上所定义）。

定义 2（非重复策略） *[43]* 不重复采样策略 s 是一个函数 s： $\mathcal{D} \to \mathcal{X}$，满足：若 $s(d)$ $= x$，则或者 $d(x) = ?$，或者 $|d| = m$。

定义 3（优化算法） *[43]* 基于采样策略 s 的优化算法 A 是对该采样策略的迭代使用，并更新数据：

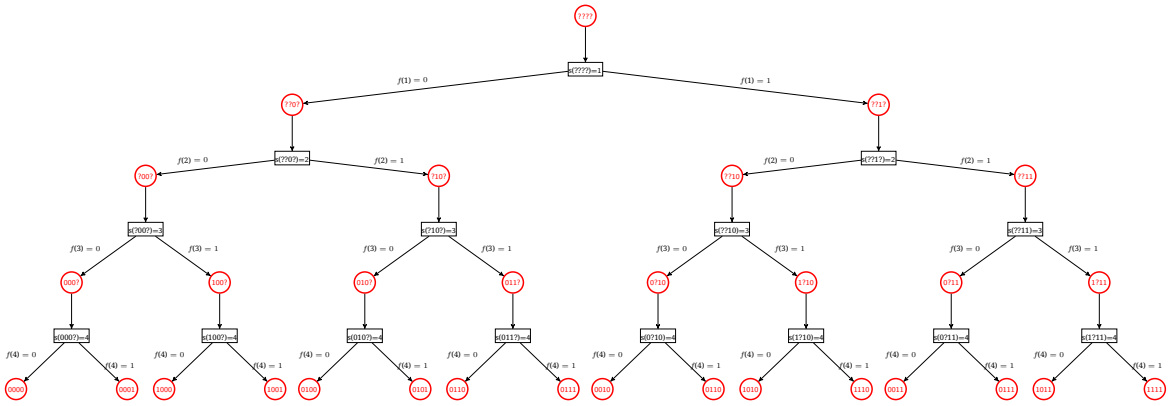

图1：以树表示策略的示例：针对函数 f 的策略： $\{1,2,3,4\} \to \{0,1\}$。

(a) 将 d 设为初始数据；*(b)* 如果不存在未采样的 $x \in \mathcal{X}$，则终止；*(c)* 在 $x = s(d)$ 处采样，并将结果添加到数据 d 中，即设置 $d(x) = f(x)$；*(d)* 转到步骤 *(b)*。

我们使用图1所示的有向图，展示一种示例性优化算法。在该图中，节点表示数据。边表示根据优化器的采样原则对问题函数进行采样所产生的所有潜在数据转换。图1的灵感源自 [43]。

本文组织如下。第4节介绍排列迭代搜索框架和基准测试协议，包括效率度量。第5—8节分析算法-函数依赖关系，以及代数结构和变换的影响。第9节讨论对统计学和进化计算的影响，并总结全文。

4 迭代搜索算法

我们将迭代搜索算法定义为一组不重复的采样点序列（如果我们是从业者，可能会把算法视为一个过程。只需想象，这里给出的定义正是这一过程的结果）。这些是位置的排列（在[39]中：函数 ϕ: $\mathcal{X} \to \mathcal{X}$ 当 ϕ 是双射时，我们称其为排列），因此共有 $4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24$ 种可能的算法，例如表2中列出的那些。

根据上述对点采样问题的描述，我们可以列出一个算法表：

算法	$a1$	$a2$	$a3$	$a4$	$a5$	$a6$	$a7$	$a8$	$a9$	$a10$	$a11$
序列	(1,2,3,4)	(1,2,4,3)	(1,3,2,4)	(1,3,4,2)	(1,4,2,3)	(1,4,3,2)	(2,1,3,4)	(2,1,4,3)	(2,3,1,4)	(2,3,4,1)	(2,4,1,3)
算法	$a12$	$a13$	$a14$	$a15$	$a16$	$a17$	$a18$	$a19$	$a20$	$a21$	$a22$
序列	(2,4,3,1)	(3,1,2,4)	(3,1,4,2)	(3,2,1,4)	(3,2,4,1)	(3,4,1,2)	(3,4,2,1)	(4,1,2,3)	(4,1,3,2)	(4,2,3,1)	(4,2,1,3)
算法	$a23$	$a24$									
序列	(4,3,1,2)	(4,3,2,1)									

表2：24个预备序列。

4.1 24种基于排列的算法的特征

表2中列出的全部24种算法都采用相同的解空间探索策略，唯一的区别在于四个预定义输入点的排列顺序。每种启发式算法的运行机制都完全相同：算法按固定的确定性序列进行四次评估，不进行自适应参数调优，也不使用随机性或内存。实际上，这意味着算法之间的唯一区别，就是评估这相同四个点的顺序。尽管没有学习或记忆，各算法在搜索过程的不同阶段仍会遇到不同的结果——访问顺序决定了哪些信息（即目标函数值）会被“较早”看到，哪些会被“延后”看到。概念上，每种算法都是针对相同四个输入配置定义的确定性过程，不涉及任何基于代码的实现、随机性、内存或自适应机制。

采样顺序直接影响探索与开发之间的平衡。初始评估点决定了首先检查输入空间中的哪些区域。首先对彼此相近的点（在输入空间中相邻的点）进行采样的算法，会立即将评估集中在有限区域内（侧重开发）；而从距离较远或位于极端位置的点开始采样的算法，则会先进行更广泛、分布更均匀的探索（侧重探索）。例如，按照 (1，2、3、4) 的顺序评估点的算法，会迅速发现目标函数的局部结构；而像 (1，4、2、3) 这样的排列，起初会考察两个边界区域，从而更全面地了解空间，但代价是向有希望的区域收敛得更慢。

这些策略的主要影响可总结如下：

- 早期区域发现：采样顺序决定了首先发现目标函数的哪些区域。从相邻点开始的算法往往聚焦于局部区域（开发性更强），而从相距较远的点开始的算法则覆盖更广的区域（探索性更强）。因此，不同的排列会导致发现有价值区域的速度各不相同。

- 无自适应性——多样化的观测序列：表2中的所有算法都是确定性的，且不具备自适应性：固定的先验评估顺序会以不同顺序揭示各个区域，但绝不会利用已观测到的函数值来选择下一个点。顺序不同意味着每种排列都会生成独特的函数值观测序列（即成功与失败模式各异）。因此，观测值（迄今最优值）的轨迹也各不相同，进而导致效果不同。在相同的评估预算下，结果仍取决于所选顺序。

- 模式相似性与聚类：采用相似初始方案的算法（例如，从同一点出发或最初评估相邻元素的算法）往往呈现相似的行为和性能。密切相关的排列会产生相似的评估轨迹，这在相关性分析和层次聚类分析中表现为簇。采样顺序上的这些结构化差异，清楚地解释了所观察到的算法行为分组。

尽管所有算法评估的都是相同的四个点，但采样顺序的差异会形成不同的信息轨迹，从而导致效果各异。所给出的集合提供了一个精确控制的实验框架，可用于分析启发式结构如何影响优化性能。

4.2 基准函数评估

为了通过实证考察24种基于排列的算法之间的差异如何影响其性能，我们现在转向受控的基准测试设置。我们考虑测试函数 $B = \{f1, ..., f16\}$。与许多人工构造的测试函数集的情况一样，我们假设已知所有测试函数问题的最小值，此处该值为零。对于每一对(a_i, f_j)，我们记录算法 a_i 找到函数 f_j最小值所需的最少试验次数$s_{i,\ j}$。为便于比较不同算法和函数，我们计算一个称为效率 [16]的归一化指标:

$$E_{a_i,f_j} = \frac{|\mathcal{X}| - s_{i,j}}{|\mathcal{X}| - 1} \tag{1}$$

其中， $|\mathcal{X}|$ 表示定义域中的点数。例如，在我们的主要设置中， $n = 4$，因此 $\mathcal{X} = \{1, 2, 3, 4\}$，进而 $|\mathcal{X}| = 4$。效率得分介于 1（在第一次试验中找到最优值）和 0（在最后一次可能的试验中找到最优值）之间。

定义 4 (函数排列**)** *[43]*：设 ϕ 为一个排列 $\phi: \mathcal{X} \to \mathcal{X}$，且 $f: \mathcal{X} \to \mathcal{Y}$ 为任意函数。我们称 f_ϕ 为函数排列，其中定义$f_\phi(x) = f(\phi(x))$。

定义 5 (排列封闭**)** *[38], [43], [50]*：设 $\mathcal{F}$ 为一个集合函数： $\mathcal{X} \to \mathcal{Y}$。我们称 $\mathcal{F}$ 在排列下封闭，或称排列封闭，当且仅当对于任意排列 ϕ： $\mathcal{X} \to \mathcal{X}$， $f \in \mathcal{F} \Longrightarrow f_\phi \in \mathcal{F}$.

例 *2*（非排列封闭）令 $f := f_2$ 来自表 *1*，即 *($f(1)$ $f(2)$ $f(3)$ $f(4)$) =(1, 0, 0, 0)*。令 ϕ *=(1 2)* 为对换。于是 *($f_\phi(1)$ $f_\phi(2)$ $f_\phi(3)$ $f_\phi(4)$) =(0, 1, 0, 0)* $\neq$ *(1, 0, 0, 0)*，因此 $\mathcal{F} = \{f\}$ 不是排列封闭的。

定义 6 （排列闭包） *[39], [43]*：令 $\mathcal{F}$ 为函数映射的集合：$\mathcal{X} \to \mathcal{Y}$。我们将 $\mathcal{F}_{c.u.p}$ 定义为包含 $\mathcal{F}$ 且在排列下封闭的最小集合。

不久前，Maurice Clerc 在 [16] 证明，对于某些连续函数类，或者由于元素排列而形成的某些非闭集（非排列封闭），NFL定理 [1], [4], [25], [31], [36], [38], [39], [43], [44], [50], [56], [62], [65], [74], [77], [83], [90], [91]并不成立。可以证明函数在排列封闭子集上的某些性质。

为了直观展示 24 种采样算法在整个测试函数集上的表现，我们提供了两张汇总表。

表3显示了算法 a_i 找到函数 f_j 最小值所需的步数 $s_{i,\ j}$ （即首次达到最优值前的评估次数）。

表4报告了相应的效率值 $E_{a_i,\ f_j}$，这些值根据公式（1）定义，并归一化到范围 $[0, 1]$。这些分数表示在采样序列中发现最优值的早晚程度：1 表示在第一步就成功，0 表示在最后一个可能的步骤才成功。

每张表的最后一行报告了所有测试函数中每种算法的总步数或平均效率。

作为比较，我们还考察了较小的维度 $n = 2$ 和 $n = 3$，它们分别需要7次和15次试验次数才能完成完整的函数识别。与此一致，算法的平均效率随 n增加： $n = 2$为0.5，$n = 3$约为0.625， $n = 4$为0.71。这一趋势支持如下一般性观察：更大的输入维度会提高算法性能指标的判别能力。

函数 \ 算法	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}	a_{20}	a_{21}	a_{22}	a_{23}	a_{24}
f_1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
f_2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
f_3	1	1	1	1	1	1	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1
f_4	3	3	2	2	2	2	3	3	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1
f_5	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	1	1	1	1	1	1
f_6	2	2	3	3	2	2	1	1	1	1	1	1	3	3	2	2	2	2	1	1	1	1	1	1
f_7	1	1	1	1	1	1	2	2	3	3	2	2	2	2	3	3	2	2	1	1	1	1	1	1
f_8	4	3	4	3	2	2	4	3	4	3	2	2	4	3	4	3	2	2	1	1	1	1	1	1
f_9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2
f_{10}	2	2	2	2	3	3	1	1	1	1	1	1	1	1	1	1	1	1	3	3	2	2	2	2
f_{11}	1	1	1	1	1	1	2	2	2	2	3	3	1	1	1	1	1	1	2	2	3	3	2	2
f_{12}	3	4	2	2	4	3	3	4	2	2	4	3	1	1	1	1	1	1	4	3	3	4	2	2
f_{13}	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	3	3	2	2	2	2	3	3
f_{14}	2	2	3	4	3	4	1	1	1	1	1	1	3	4	2	2	4	3	3	4	2	2	4	3
f_{15}	1	1	1	1	1	1	2	2	3	4	3	4	2	2	3	4	3	4	2	2	4	3	3	4
f_{16}	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
总计	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**

表3：n=4所需采样点数量表。

函数 \ 算法	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}	a_{20}	a_{21}	a_{22}	a_{23}	a_{24}
f_1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
f_2	2/3	2/3	2/3	2/3	2/3	2/3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
f_3	1	1	1	1	1	1	2/3	2/3	2/3	2/3	2/3	2/3	1	1	1	1	1	1	1	1	1	1	1	1
f_4	1/3	1/3	2/3	2/3	2/3	2/3	1/3	1/3	2/3	2/3	2/3	2/3	1	1	1	1	1	1	1	1	1	1	1	1
f_5	1	1	1	1	1	1	1	1	1	1	1	1	2/3	2/3	2/3	2/3	2/3	2/3	1	1	1	1	1	1
f_6	2/3	2/3	1/3	1/3	2/3	2/3	1	1	1	1	1	1	1/3	1/3	2/3	2/3	2/3	2/3	1	1	1	1	1	1
f_7	1	1	1	1	1	1	2/3	2/3	1/3	1/3	2/3	2/3	2/3	2/3	2/3	2/3	2/3	2/3	1	1	1	1	1	1
f_8	0	1/3	0	1/3	2/3	2/3	0	1/3	0	1/3	2/3	2/3	0	1/3	0	1/3	2/3	2/3	1	1	1	1	1	1
f_9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2/3	2/3	2/3	2/3	2/3	2/3
f_{10}	2/3	2/3	2/3	2/3	1/3	1/3	1	1	1	1	1	1	1	1	1	1	1	1	1/3	1/3	2/3	2/3	2/3	2/3
f_{11}	1	1	1	1	1	1	2/3	2/3	2/3	2/3	1/3	1/3	1	1	1	1	1	1	2/3	2/3	1/3	1/3	2/3	2/3
f_{12}	1/3	0	2/3	2/3	0	1/3	1/3	0	2/3	2/3	0	1/3	1	1	1	1	1	1	0	1/3	1/3	0	2/3	2/3
f_{13}	1	1	1	1	1	1	1	1	1	1	1	1	2/3	2/3	2/3	2/3	1/3	1/3	2/3	2/3	2/3	2/3	1/3	1/3
f_{14}	2/3	2/3	1/3	0	1/3	0	1	1	1	1	1	1	1/3	0	2/3	2/3	0	1/3	1/3	0	2/3	2/3	0	1/3
f_{15}	1	1	1	1	1	1	2/3	2/3	1/3	0	1/3	0	2/3	2/3	1/3	0	1/3	0	2/3	2/3	0	1/3	1/3	0
f_{16}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Mean	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**	**0.71**

表4：效率算法表。

5 二元函数优化算法的统计与可视化分析

二元函数的优化是数值计算和人工智能中的一个基本问题。选择合适的优化算法会显著影响在不同搜索空间中求解问题的效率。本分析基于图2和图3中给出的两个数据集，它们深入揭示了各种算法的性能差及其与不同测试二元函数的相互作用。

图2中的相关图对表2所列的24种基于排列的算法之间的关系进行了详细的定量评估，揭示了这些算法在表1所列的各种测试二元函数上的性能内在结构。该矩阵以颜色编码，相关系数范围为 -1 至 1，突出显示了所评估算法之间相似与差异的关键模式。深红色表示强正相关，表明算法对的性能高度一致，意味着它们的搜索机制以相似的方式探索和开发解空间。相反，蓝色表示负相关，意味着算法产生了根本不同的优化行为，这可能是由于它们在探索与开发之间采用了不同的平衡策略。

具有高相关值的算法聚类表明，存在性能上高度一致的主导启发式结构。特定算法对$a1$ 和 $a2$展现出近乎完美的相关性，意味着几乎相同的搜索动态

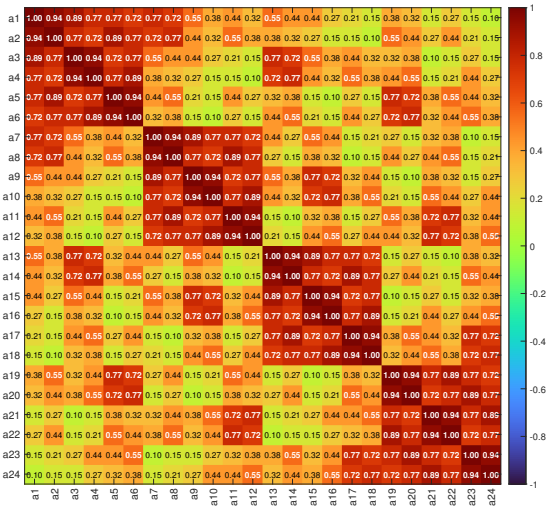

图2：展示24种排列顺序（采样策略） $a1$ - $a24$ （表2）性能特征之间成对相关性的相关图，即仅采样顺序不同的同一非重复采样算法的不同变体。坐标轴表示策略；颜色深浅表示相关系数。

，这很可能是由于共享的启发式设计原则。同样，$a3$ 与 $a7$ 之间的关系表明，这些算法采用互补策略，对测试函数景观的响应相似，从而得到稳定且可预测的结果。 $a12$ 与 $a16$ 之间的强相关性进一步强化了这一趋势，表明这些算法在不同函数集上保持平衡的方法，从而导致相似的收敛行为。

考察中等相关性的分布可以发现，某些算法的搜索行为部分一致，但在特定函数类中仍保留各自的特点。例如， $a5$ 与 $a9$之间的相关性表明二者的性能趋势有一定共性，但在某些函数上的效率仍存在明显差异。这种差异可能源于局部搜索强化方面的不同，或源于它们在约束各异的复杂解空间中导航的能力不同。同样，算法 $a10$ 与 $a14$的相关值处于中等水平，表明二者的优化轨迹有一定重叠，同时仍保留各自的自适应机制。

对于相关图中算法 $a19$、 $a21$ 和 $a23$所呈现的明显差异，应谨慎解读。尽管它们在相关矩阵中的行与原始列顺序下算法 $a3$ 和 $a4$ 等的行不同，但按升序排列相关系数后，得到的特征完全相同。这表明这些算法的相关值分布（多重集）相同，差别仅在于这些值对应到特定算法对的方式不同。因此，不应将它们视为整体相关强度方面的离群值。相反，它们的差异体现在关系结构上，源于成对相似性的排列不同。这可能影响它们在层次聚类中的位置，但并不意味着其全局相关性概况存在根本差异。

该相关图还揭示了算法群体之间错综复杂的结构依赖关系，其中，强相关方法形成的簇表现为功能等效的启发式方法。这些分组表明，某些算法在性能方面可以互换，从而进一步支持了这样的假设：某些启发式设计会趋于

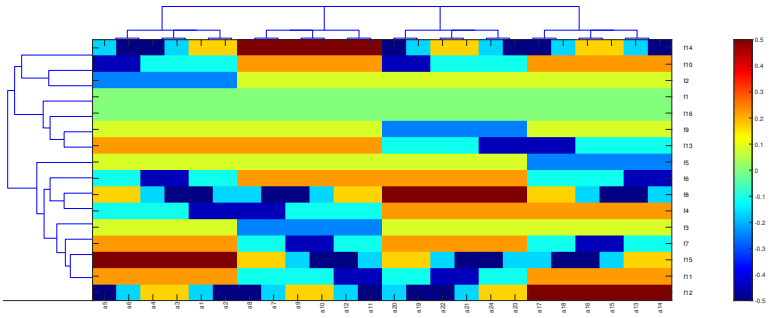

图3：二值函数优化中基于排列的算法的层次聚类。横轴：按照性能相似性聚类的基于排列的算法。纵轴：二值函数，表示算法在不同优化景观中的行为。右侧的色标表示行均值中心化性能值（相对于函数均值的偏差）：红色和橙色表示对于给定函数，性能高于平均水平；蓝色色调表示性能低于平均水平。绿色和黄色对应接近函数均值的数值，表示偏差较小。

常见优化模式。这些关系的层次性表明，在受限的函数集中，某些算法始终优于其他算法，从而对所有可能问题中的算法有效性均一的假设提出了质疑。这一结构性洞见有助于更深入地理解算法行为，并据此对启发式方法进行分类，评估它们在受控基准测试环境中的预期效率。

通过相关图对相关结构进行可视化，是评估算法效率和冗余性的关键分析工具。识别出高度相关的算法对，有助于了解启发式方法之间的功能重叠；而相关性较弱的区域则为探索互补策略提供了机会，这些策略可能提升性能多样性。逐渐显现的聚类模式表明，某些算法的行为几乎完全相同，而其他算法则存在显著差异，这反映了启发式机制以及对不同优化景观的适应性各不相同。这些观察结果为深入研究算法选择策略奠定了基础，尤其是在设计能够利用互补搜索行为的集成或混合方法时。文献中，Surjanovic 和 Bingham [80] 指出，启发式算法的有效性与优化景观的特征密切相关，这一点也得到了所得结果的证实。

图3展示了行均值中心化性能值的层次聚类热图。它揭示了算法与函数之间的结构化关系，从而能够对所研究空间内的结构依赖关系进行详细评估。该可视化呈现了算法与函数的层次分组。这使得可以根据测试函数的特征识别算法性能模式。算法的聚类表明，某些方法表现出高度的性能一致性。这可能意味着相似的探索与利用策略。特别是，由 $a1$， $a3$， $a7$， $a10$ 组成的算法组在多个函数上表现出稳定性以及一致的相对性能模式。这表-

明这些算法对测试函数结构变化的敏感性较低，可被视为所分析基准测试中相对通用的策略。

相反，仅凭热图，不应将算法 $a5$、 $a8$和 $a12$解读为整体数值分布特征存在根本差异。按行排序比较（例如， $a12$ 与 $a3$）表明，排序后的分布特征可能完全相同。在这种情况下，差异源于数值被分配给了哪些具体函数。因此，这种区别是函数特定的（关系性的），而非整体数值分布的差异。这表明存在依赖于函数的选择性（即相对专门化），而不是全局上截然不同的数值模式。

测试函数的聚类分析显示，函数 $f1$、 $f2$、 $f5$ 构成一个具有相似相对性能模式的同质组。这与它们相似的代数性质和结构性质一致。这也表明，在其中一个函数上表现出色的算法，往往在同组其他函数上也表现出相似的相对优势。另一方面，函数 $f3$、 $f4$、 $f6$的结果会因所用算法不同而表现出更大的变异性。这可能意味着，它们的结构需要特定的探索策略。函数 $f8$、 $f9$、 $f13$也呈现出显著的性能差异。这些函数的按行均值中心化数值具有明显的局部变化。这可能表明其函数景观较为复杂，需要采用更具自适应性的优化方法。

值得特别注意的是，函数 $f12$、 $f14$、 $f15$ 的按行均值中心化数值呈现出强烈的对比。对于某些算法，尤其是 $a17$、 $a22$、 $a24$，这些函数之间的差异十分显著。这可能表明，这些算法采用的搜索机制与其他方法有所不同。此外，某些算法 $a9$、 $a11$、 $a20$ 未在层次结构中形成明显的簇，这可能意味着它们的效果更加多样。也可能说明它们的效果取决于测试二元函数的特定特征。

树状图还突显出算法中彼此紧密聚集的子群。集合 $a2$、 $a4$、 $a6$ 尤其明显：它们在大多数函数上的相对性能剖面几乎完全相同。这表明这些方法采用了非常相似的搜索空间探索机制。相比之下，群组 $a19$、 $a21$、 $a23$ 的结果差异大得多。这说明这些算法在测试函数的某些特定子集上相对更有效，但在整个问题空间中并非同样高效。

这些观察结果与算法分类以及针对特定问题的优化方法的选择高度相关。具有相似性质的函数呈现出清晰的聚类，这表明可以针对特定问题类型定制一组算法来有效开展优化。这优于在所有情况下依赖单一方法。结果还表明，在所研究的空间内，存在更具挑战性的函数，其算法性能的差更为显著。这使它们成为进一步研究优化方法及其对特定问题类别的适配的关键参考点。

Hansen 等人也有类似的观察。 [33], 他们提出了一种针对黑箱优化的随时性能评估方法，该方法基于达到预定义目标所需的函数评估次数。这个框架能够在不同问题类别和评估预算之间进行定量性能比较。这一关系与 Surjanovic 和 Bingham 汇编的基准函数描述一致。 [80]他们表明，复杂景观（例如窄谷或许多局部极小值）会增加优化难度，并阻碍收敛到最小值。

6 与实现无关的可加性度量

优化算法在解决科学与工程学科广泛领域中的复杂问题时发挥着关键作用。准确且一致地评估其性能，对于理论分析和实际应用都至关重要。然而，传统的评估指标往往依赖于硬件、软件环境和实现细节。在黑箱优化中，通常以目标函数评估次数而非实际运行时间报告性能，因为依赖于实现的计算开销可能占据主导地位，而且在不同方法和实现之间可能存在显著差异。 [61]这种依赖使得人们难以对算法效率和行为得出普遍结论。为克服这些局限，研究人员日益关注**与实现无关的度量**的开发。与实现无关的度量不受硬件和软件依赖因素的影响，而是专注于算法和目标函数的内在属性。这类度量对于实现算法的公平比较至关重要，无论所使用的计算环境或编程语言如何。它们能够分离出算法对性能的贡献，并加深对优化原理的理解。在此背景下，与实现无关的度量最引人关注的性质之一 [33],[96] 便是其**可加性**。可加性指的是将复合函数分解为更简单的分量、分别进行分析，并重新组合其结果以预测复合函数上的性能的能力。这一性质不仅在理论上优雅，而且具有重要的实际意义，因为它能帮助我们洞察算法行为，并为复杂优化任务设计更高效的策略。与实现无关的度量是否具有可加性，这个问题绝非显而易见。可加性意味着，算法在复合函数上的性能 $fk = fi + fj$可以直接由其在各个单独函数上的性能 fi 和 fj推导得出。这在优化领域尤其值得关注，因为它使我们能够：1. 性能预测：根据较简单的分量推断算法在复杂函数上的性能，可以降低计算成本并加快评估过程。2. 算法设计：理解可加性有助于指导算法设计，

explo它 e struct成 f 组合 ite functions for i改进 d effici性。
3. 跨领域泛化：可加测度在从机器学习、博弈论到风险分析和经济学等各个领域都有应用，为研究多样化系统提供了统一框架。

此外，**许多实际优化问题的变量分辨率有限（例如，受物理限制或数值精度的影响），这实际上使搜索空间离散化，并促使人们研究离散和二元目标函数 [82]**。尽管这一问题十分重要，文献对与实现无关的度量的可加性关注有限，尤其是在二元目标函数的背景下。本文提出的结果旨在通过系统分析可加性及其对优化算法的影响来填补这一空白。与实现无关的度量的理论基础可追溯至合作博弈论，其中引入了**夏普利值** [9],[14],[48]，用于量化各个玩家对博弈总收益的贡献。形式上，设 v 是定义在特征集合 $\mathbb{N} = \{1, 2, \ldots, n\}$ 上的特征函数，其中$v(S)$ 表示联盟 $S \subseteq \mathbb{N}$ 的价值。夏普利值 $\phi_i(v)$ 由公式（2）定义：

$$\phi_i(v) = \sum_{S \subseteq \mathbb{N} \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} \left[v(S \cup \{i\}) - v(S) \right] \tag{2}$$

它根据玩家的边际贡献，在玩家之间公平分配总价值。其性质包括效率、对称性、虚设玩家和可加性，使其成为分析具有可加特征的系统的有力工具。在优化领域，这些性质体现为性能度量：它们在不同实现之间保持不变，并能准确反映算法的内在能力。尤其是可加性使复合函数能够分解，为研究其行为提供了系统方法。本文中的“分解”是指定义在同一定义域上的基准函数（例如，$f5 + f10 = f14$）之间逐点成立的代数关系，而非$f(x, y) = g(x) + h(y)$ 对定义域内所有 (x, y) 成立这类经典的变量可分离性。本研究通过实证分析，探讨与实现无关的度量的可加性；分析涵盖应用于 16 个二元目标函数的 24 种优化算法。具体目标如下：1. 建立一个严格的框架，用于评估与实现无关的度量的可加性

t

措施。
2. 对算法在二元函数的求和与差上的表现进行实证分析，并研究可加性成立的条件。

3. 确定可加性在改进算法设计、降低计算成本以及理解特征交互方面的实际影响。

本研究的结果揭示了若干非平凡且极其重要的发现：

- 增强算法效率：某些算法在复合函数上的表现始终更佳，能够以更少的试验次数达到最优值。例如，算法 $a4$在三次试验中即可为 $f5 + f10 = f14$达到最优值，而单独处理 $f14$则需要四次试验。

- 性能中的系统性模式：通过根据算法的数值序列对其进行分组，并分析它们在复合函数上的行为，我们发现了一些模式，表明函数复合与算法效率之间存在更深层的结构关系。

- 广泛适用性：研究结果并不局限于二元目标函数，也可能对机器学习、经济学和风险评估等广泛领域产生影响 [9],[14],[48],[66]。

这些结果展现了可加性分析为优化带来新见解的潜力，并促使我们进一步探索这一性质在更复杂情境中的表现。

本节为与实现无关的度量提供了严谨的理论基础，并特别强调可加性这一性质。通过定义这些度量并探究其关键性质，我们建立了一个框架，以理解它们在分析优化算法中的作用。

一种**与实现无关的（性能）度量** [33],[96]，是用于评估优化算法性能的定量度量，不受计算环境具体条件的影响。与传统度量（如运行时间或计算复杂度）不同，这些度量仅依赖于算法和目标函数的内在属性。

令 $\mathcal{F} = \{f1, f2, \ldots, fn\}$ 表示一组目标函数，令 $\mathcal{A} = \{a1, a2, \ldots, am\}$表示一组算法。

定义 7 令 $M(ai、\ fj)$ 表示算法 ai 在定义域 $\mathcal{X}$ 上发现目标函数 fj 的全局最小值所需的评估次数（查询次数）。这等价于第*4.2*节中定义的值$s_{i,\ j}$ 。

如果满足以下标准，则认为该度量与实现无关：1. 不变性： $M(ai, fj)$ 不会因硬件、软件或编程语言的变化而改变。2. 一致性：该度量反映了算法探索解空间的基本效率。3. 可复现性：使用 $M(ai, fj)$ 得到的结果，在不同的实验设置下均可复现。与实现无关的度量具备若干理想属性，可确保其在分析和比较算法时发挥作用：(a) 效率：所有应用于函数 fj 的算法所需的函数评估次数总计，应与确定全局最优值所需的理论最小值一致。该属性表示为： $\sum_{i=1}^{m} M(ai, fj) \geq M^*(fj)$，其中 $M^*(fj)$ 是评估次数的理论最小值。(b) 对称性：如果两个函数 fi 和 fj 结构等价，则对于任何算法，该度量都应给出相同的结果： $M(ak, fi) = M(ak, fj) \quad \forall ak \in \mathcal{A}$.(c) 单调性：对于可分解为两个较简单子函数$fj = fp + fq$的函数 fj ，该度量应满足： $M(ak, fj) \geq \max\{M(ak, fp)\ M(ak, fq)\}$。(d) 可加性：可加性是一项至关重要的属性，它确保复合函数的度量可以表示为其各分量度量之和。对于$fj = fp + fq$，该属性可形式化为： $M(ak, fj) = M(ak, fp) + M(ak, fq)$。为更准确地阐明可加性的概念，我们将运算 $fk = fi + fj$形式化为定义域 $\mathcal{X}$ 上的逐点和。也就是说，对于每个 $x \in \mathcal{X}$，我们定义： $fk(x) = fi(x) + fj(x)$，其中 fi、fj、fk： $\mathcal{X} \to \{0, 1\}$，加法指整数加法。为确保 fk 仍为二值函数，我们只考虑满足以下条件的情况：对于所有 x， $fi(x) + fj(x) \in \{0, 1\}$。例如，函数 $f14 =$(1, 0, 1, 1) 是 $f5 =$(0, 0, 1, 0) 与 $f10 =$(1, 0, 0, 1) 的逐点和。然而，即使

f

$f14(x) = f5(x) + f10(x)$ 对所有 x 均成立，但性能度量 M 可能不满足可加性。在我们的实验中，我们观察到 $M(a4,\ f14) \neq M(a4,\ f5) + M(a4,\ f10)$。这表明，函数的结构可加性并不能保证可加性能；可加性能可能取决于组合方式如何影响搜索空间的景观。添加函数可能会降低目标景观中的对称性，从而增强对搜索候选解的区分能力，并有助于提升算法性能。

7 基于代数结构的基准测试中的算法层级：关于无免费午餐定理的新视角

现代优化算法分析不仅需要了解其运行方式，还需要准确把握其所测试问题的结构。在无免费午餐定理的背景下，一个尤为重要的方面是基准测试的组织方式，以及它们如何影响启发式算法的性能。文献中一个被忽视的关键问题是：在保持排列封闭性质的集合中，对测试函数进行代数运算所产生的影响。本章将详细分析函数加法和函数减法如何影响基于排列的算法在基于排列封闭结构的基准测试中的性能。具体而言，我们将采用表 2 所列的 24种基于排列的算法，并在基准测试上对其进行评估；这些基准测试

由表 1 中的排列封闭函数构建。这些基准测试基于表 1 中的函数，通过排列封闭结构内的求和与求差运算构造，不包括函数 $f1$。经过这些代数变换后，所得基准测试仍保留排列封闭性质（图5中的纵轴表示一个排列封闭基准测试，它由排列封闭集合中除 $f1$ 外的所有可能函数加法构造而成。同样，图 7中的纵轴表示一个排列封闭基准测试，它由排列封闭集合中除 $f1$ 外的所有可能函数相减生成）。这类基准测试用于研究启发式方法对测试函数结构变化的稳健性，以及构建具有可预测优化性质的问题空间 [9],[45],[89],[90]。通过利用算法性能相关图，我们旨在回答以下关键研究问题：

- 函数之间的代数关系会影响算法的分类吗?
- 尽管在全局范围内符合无免费午餐定理，加法和减法运算是否会导致性能上的局部不等式?
- 无论测试二元函数经历何种代数变换，某些算法策略是否仍能保持稳定?

当代数运算以有利于特定搜索策略的方式改变函数景观时，对函数进行加法或减法会导致算法性能出现局部不等式。实验结果表明，算法效率的差异主要出现在函数减法的情况下：一些算法能够更快地达到最优值，这表明搜索空间结构的变化会带来局部偏好。尽管无免费午餐定理在全局范围内仍然成立，但在受限的函数集合上，根据目标函数的变换方式，所需评估次数可能会出现系统性偏差。

该分析涵盖两个基本方面：

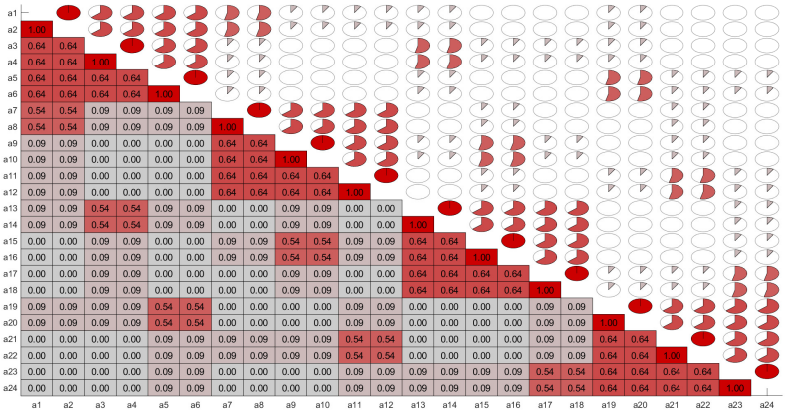

图 4：基于求和的 c.u.p. 基准测试中算法性能的相关矩阵。图例：红色/橙色色调——算法性能之间的高度正相关。蓝色色调——强负相关，表示相互对立的性能趋势。绿色/黄色色调——中等相关性水平，表示部分相似性。

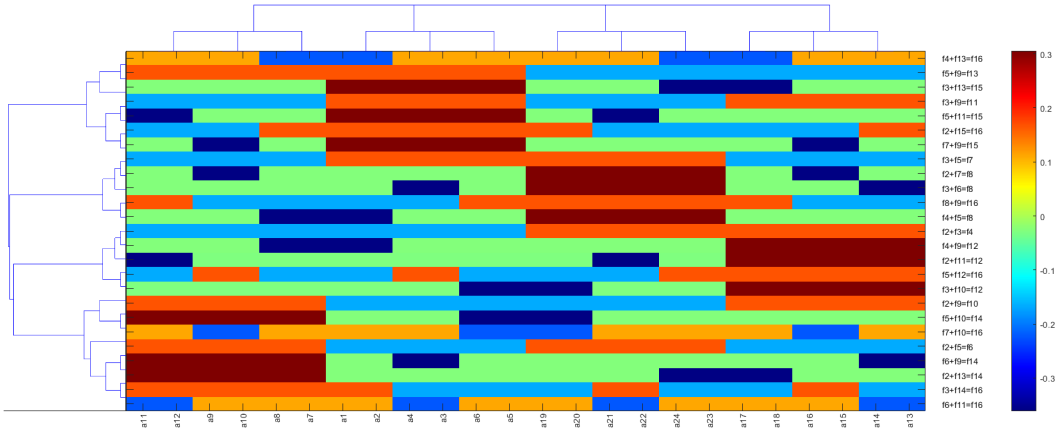

图5：基于排列的算法在 c.u.p. 基准测试中的函数加法上的层次聚类。横轴：按照性能相似性聚类的基于排列的算法。纵轴：函数加法（c.u.p. 基准测试中的复合函数），展示算法在代数结合的优化景观上的行为。右侧色标表示行均值中心化性能值（相对于函数均值的偏差）：红色和橙色表示在给定复合函数上的性能高于平均水平，蓝色色调则表示性能低于平均水平。绿色和黄色表示数值接近函数均值，偏差较小。树状图展示了根据性能模式相似性对函数和算法进行的层次聚类。

7.1 函数加法

图 4 所示的高级相关图展示了 24种基于排列的算法在二元函数上的性能相关矩阵。水平轴和垂直轴均对应于标记为 $a1$ 至$a24$ 的各个算法，表示它们在优化过程中的相互性能相关性。矩阵结构有助于识别算法之间的关系，突出显示性能趋势中的强依赖和弱依赖。

颜色渐变和圆形视觉元素对算法对之间的相关系数进行编码。深红色区域表示高度正相关，即两种算法的性能密切相关，表明它们以相似的方式探索搜索空间。相比之下，由较浅色调表示的接近零的相关性表明算法之间的行为相互独立，意味着它们采用了不同的搜索策略。例如，算法 $a1$，$a2$，$a3$，$a4$，$a5$，$a6$ 表现出显著的正相关性，表明这些方法共享相似的采样机制，从而在经过测试的二元函数上形成可比的效率特征。同样，另一个由 $a13$，$a14$，$a15$，$a16$，$a17$，$a18$ 组成的子群形成了一个紧密相关算法的独立集群，反映出它们相似的性能模式。

相反，$a9$、$a10$、$a11$、$a12$ 等算法与大多数其他方法的相关性较弱或接近于零，表明它们的优化行为与强相关组中的算法存在显著差异。这种分化表明，这些方法采用了不同的探索与利用的平衡，从而导致性能变化——

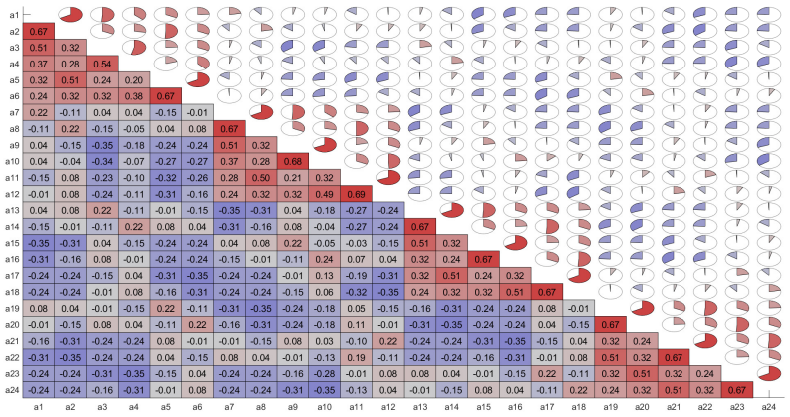

图6：c.u.p. 基准测试中函数相减的算法性能相关矩阵。深红色：高度正相关（接近 1）——算法在相同函数和上的行为几乎完全相同；浅红色：中度正相关——算法的性能趋势有一定相似性；灰色：接近零相关——算法在函数和上的表现不存在显著关系；圆圈：相关性的可视化表示——红色部分越大，正相关越强；X 轴和 Y 轴：在函数和上进行评估的 24种基于排列的算法；树状图：根据算法在函数和上的性能相似性进行层次聚类。

其表现难以根据其他算法的行为轻易预测。矩阵中特定区域存在孤立的相关性，这进一步表明，某些算法策略在某些二元函数子集上具有独特优势，但在其他子集上效率低下。

采用圆形元素的上三角矩阵表示，从另一个角度展现了相关强度。每个圆中红色填充部分的比例直观地凸显了算法之间的相似程度，让人能够直观理解性能依赖关系。最大的完整填充圆形，例如连接 $a1$与 $a2$、 $a3$ 与 $a4$ 或 $a5$ 与 $a6$的圆形，证实了它们的性能分布几乎相同；而零散或近乎空白的圆形，如涉及 $a9$或 $a12$的交互，则凸显了较弱的相互依赖性。

这些发现进一步证实，基于排列的优化方法在结构上可划分为不同的性能类别：某些方法表现出稳定的效率模式，而另一些则遵循独特的搜索轨迹。结果表明，函数景观在决定算法行为方面起着关键作用。尽管某些方法能够推广至多个函数类别，但另一些方法则展现出专门的性能优势，难以将其外推至特定函数组之外。这一可视化结果凸显了在二元优化任务中谨慎进行算法选择的必要性，因为性能趋势相似并不总能保证方法普遍适用于各种优化景观。

分析图5可见，函数和算法都呈现出明显的聚类和相对性能模式。这表明，对函数进行代数运算会产生系统性的依赖关系。该图展示了行均值中心化性能值及其层次结构，二者共同揭示了所分析基准测试中的结构化关系。

图表的配色方案反映了特定函数和对应的行均值中心化性能值。红色和橙色区域表示某个

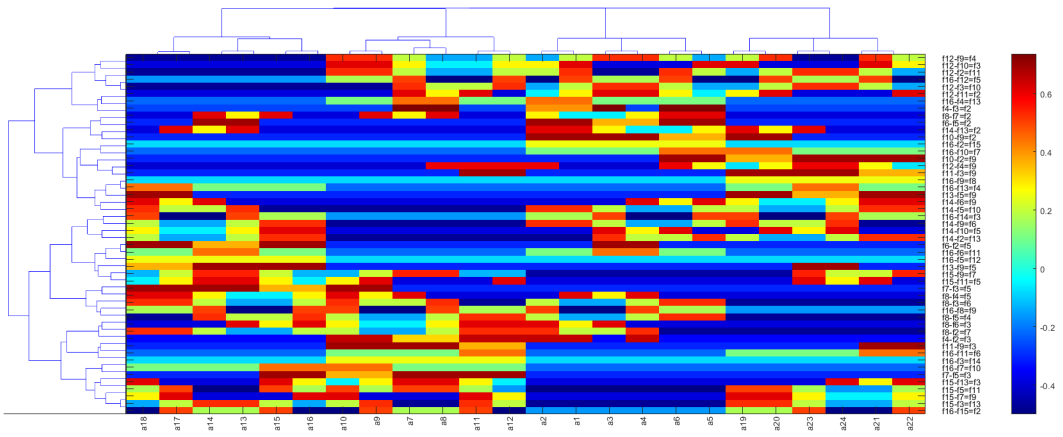

图 7：c.u.p. 基准测试中基于排列的算法在函数差上的层次聚类。横轴：按性能相似性聚类的基于排列的算法。纵轴：由 c.u.p. 基准测试得出的函数差，表示算法在基于差分的优化景观上的行为。右侧的色标表示行均值中心化性能值（相对于函数均值的偏差）：红色和橙色表示在给定函数差上的性能高于平均水平，而蓝色色调表示性能低于平均水平。绿色和黄色对应接近函数均值的数值，表明偏差较小。树状图显示了依据性能模式相似性对函数差和算法进行的层次聚类。各簇表示相对性能结构相似的分组。

给定函数和的性能高于平均水平（相对于该行的均值）。蓝色阴影区域表示该函数和的性能低于平均水平。绿色和黄色区域对应接近函数均值的数值，表示偏差较小。例如，函数 $f4 + f9 = f12$、 $f3 + f5 = f7$ 和 $f5 + f12 = f16$ 在热图的特定区域表现出明显的局部正偏差。这表明，它们的代数组合可能为算法的某些子集生成具有相似相对性能结构的优化景观。算法 $a3$、 $a7$ 和 $a12$ 在这些区域表现出相对一致的行为。这可能表明，它们在所分析的解空间中具有共同的响应模式。

相反，蓝色区域表示在特定函数和上的相对性能低于平均水平。例如，函数 $f6 + f11 = f16$ 的景观对于所分析的基准测试中的算法 $a15$ 和 $a19$ 尤其不利。与此同时，函数 $f2 + f5 = f6$ 和 $f2 + f3 = f4$ 在更广泛的一组算法上呈现出更接近函数均值的数值。这表明，它们对优化的影响较为均衡，在相对意义上不会明显偏向任何特定算法。

对该图应用层次聚类，凸显了函数的分组情况。这表明，某些测试函数的求和会形成相对性能模式相似的优化景观。函数簇 $f2 + f9 = f10$、 $f3 + f10 = f12$ 和 $f5 + f10 = f14$ 体现了可能影响算法行为的共同属性

，且这种影响方式彼此相关。对算法进行聚类进一步表明，特定优化方法具有相似的性能特征。具体而言，算法 $a7$、 $a8$ 和 $a2$ 似乎属于同一组，这表明它们对所分析的函数和具有相似的响应。相比之下，算法 $a11$ 和 $a22$ 则位于更加异质的簇中。这表明它们对测试函数结构的变化更为敏感。

从这项分析中得出的一个重要发现是，即使经过求和运算，c.u.p. 基准结构仍然保持有序。这表明，代数函数组合并未使相对性能模式完全随机化。相反，它们保留并揭示了鲜明的结构依赖关系。这一点对于NFL定理而言意义重大，因为它表明，在结构化基准测试中的特定测试函数类上，算法的性能可能存在系统性差异。

进一步分析表明，包含分量 $f2$、 $f3$、 $f5$ 和 $f9$ 的函数经常出现在与函数均值存在显著局部偏差的区域。这表明，这些函数的结构性质可能在塑造优化景观方面发挥关键作用。它们的代数组合似乎会以某种方式影响问题表述，从而使特定算法相对受益。这一发现与所分析函数类中偏离算法效率均匀随机分布的现象一致。

这些结果对特定优化问题的基准设计和算法选择具有重要影响。分析表明，与其将测试函数视为彼此独立的随机对象，不如考虑它们的代数结构。这些结构会影响算法对解空间的探索与利用。未来的研究应探讨将这一方法用于分析其他类型的代数运算。这有助于更准确地界定测试二元函数结构与优化启发式算法效率之间的依赖模式。

从实践角度来看，利用这些发现进行基准设计，对于更好地表征实际优化问题至关重要。对测试函数进行适当分组，有助于更准确地分类算法，并根据目标函数景观的特征制定优化策略。因此，未来研究可以聚焦于基于测试函数代数分析的自适应算法选择方法。这将为启发式优化领域开辟一个新方向。

7.2 函数减法

如图6和图 7所示，函数减法运算会使结果的变异性显著增大。

所展示的相关图详细分析了 24 种基于排列的优化算法之间的关系，重点考察了它们在经过系统变换的基准函数集中的协方差结构。该基准测试通过计算原始 c.u.p. 集合中函数之间所有可能的两两差值生成，其中排除了由 $f1$ 构成的函数，该函数完全由零组成。尽管排除了该函数，新形成的函数集仍在排列下封闭，保留了影响算法行为的关键结构性质。协方差矩阵的可视化揭示了算法性能中的显著模式，凸显了优化策略之间的相似之处与差异。

相关图通过双格式方法呈现算法关系：下三角矩阵中显示数值相关值，上三角部分则采用圆形分段可视化。配色方案直观地展现相关强度，红色色调表示正相关，即相应算法在修改后的函数集上表现相似，这意味着

这些算法在修改后的函数集上表现出相似的性能，而蓝色色调表示负相关，说明它们的优化倾向相反。颜色深浅反映相关性的幅度，颜色越深，对应的绝对值越高。该结构有助于快速识别具有共同搜索行为的算法簇，以及倾向于表现相反的算法对。

对相关结构的全面考察揭示了定义明确的算法分组：某些方法在组内表现出很强的一致性，同时又与其他方法保持明显区分。值得注意的是，标记为 $a1$至 $a5$的算法始终呈现较高的正相关，表明这些方法依赖相似的搜索启发式，并且受到函数变换的影响也相似。共同的性能趋势表明，这些方法具有确定性倾向，这可能源于结构化置换策略，而这类策略在变换后的基准测试中仍保持稳健。相比之下，算法 $a9$至 $a12$ 与这一初始组呈负相关，意味着它们的搜索动态存在根本差异。它们的性能分化表明，这些算法依赖更多随机性或自适应机制，可能采用概率重组或变异技术，从而能够以较少约束的方式探索解空间。

算法 $a15$，$a16$和 $a17$ 呈现中等相关值，表明它们采用了更具混合性的优化策略，融合了确定性方法和自适应方法的要素。这些算法在相关结构中的位置表明，当高度结构化的方法遇到困难时，它们能在相应情境下受益于函数变换，同时又与其他随机化方法保留一定程度的相似性。这种灵活性可能使它们特别适用于需要在探索与开发之间保持平衡的优化问题。

相关图的一个关键观察是，算法 $a19$ 至 $a24$之间存在强烈的正相关，表明这组方法的表现几乎完全相同。这种冗余表明，这些算法彼此之间可能并不具备独特优势，因而在许多优化情境中可以在功能上相互替换。该组内部持续存在的强相关性意味着它们共享某种底层搜索原理，对函数集的变化作出相似响应，这可能反映出针对变换后的基准测试某些结构性质的特定算法偏差。

与此同时，相关图还突出了相关性接近于零的算法对，表明其优化行为在很大程度上相互独立。这些独立关系在设计集成方法时尤为重要，因为将性能不相关的算法加以组合，可以提升优化任务的整体稳健性。此类独立行为的例子可见于算法 $a6$ 与 $a20$ 或 $a14$ 与 $a22$形成鲜明对比的性能特征，其相对独立性表明它们通过根本不同的方法探索函数空间。这一见解对于选择能够相互弥补弱点的互补算法很有价值，使它们非常适合混合优化框架。

这些发现的更广泛影响凸显了函数集修改对算法性能的影响。由于该基准测试是通过系统地计算原始 c.u.p. 集合内各函数之间的差异构建的，因此生成的优化景观保留了固有的结构依赖关系，同时也带来了新的计算挑战。相关图中观察到的协方差结构表明，某些算法对这些变换更为敏感，并形成了清晰的簇，反映出它们固有的优势与劣势。排除 $f1$ 可确保平凡的零差异情形不会扭曲分析，而剩余的函数集

则保持着紧密的相互依赖关系，这一点可从算法之间的相关模式看出。

分析证实，一些基于排列的算法天然适用于涉及函数差的优化任务，而另一些算法则会根据施加于目标函数的变换表现出不同程度的适应性。这进一步说明，基于排列的启发式方法会受到其函数景观结构性修改的显著影响，因此其性能特征取决于底层基准测试的属性。理解这些关系有助于根据协方差模式更明智地选择算法，尤其是在考虑混合方法或集成方法的场景中。

相关图最终从整体上呈现了变换函数条件下的算法行为，揭示了冗余的优化策略、对立的搜索方法以及功能相互独立的技术。正相关算法的聚类表明，函数变换会偏向某些策略，而负相关方法则提供了替代方案，这些方案在不同的优化场景中可能更具优势。这些洞见有助于更深入地理解排列优化中的算法适应，并为进一步研究基于协方差分析的自适应算法选择框架提供了潜在方向。

对图 7 中 24 种排列算法的层次聚类热图进行分析，在排列封闭函数减法的背景下，可以发现，算法会根据其对不同函数差的响应呈现出鲜明的聚类模式。应用减法运算后，基准结构依然保持有序。这表明，某些算法在基于减法的测试函数上表现出稳定的行为模式。

首先，算法 $a1$、$a2$、$a3$、$a4$聚为一组，彼此距离很近，其相似的色彩模式和聚类排列中的相邻位置都体现了这一点。它们彼此接近，说明在许多函数差上的行为非常相似。这表明它们可能采用了相近的搜索策略，或具有相似的启发式机制。同样，$a5$、$a6$、$a7$这一组也表现出密切的关联，其相对性能模式高度相似。这说明它们对基于减法的函数的变化有着相近的响应方式。

还值得注意的是，算法 $a10$、$a11$、$a12$虽然形成了自己的子组，但其相对性能模式往往与前两组形成对比。具体而言，对于 $a1$ -$a4$ 组相对性能往往高于平均水平的函数差，$a10$ - $a12$ 子组的相对性能可能低于平均水平，反之亦然。这种行为可能源于它们在探索和开发解空间时采用了不同的方法。

一个特别有趣的案例是组 $a15$、$a16$、$a17$。在热图的某些区域中，该组呈现出相似的相对性能模式，但在其他区域则表现出明显的内部差异。这表明，这些算法对特定类别的基于减法的函数更为敏感，其性能取决于所测试优化问题的特征。算法$a20$、$a21$、$a22$也呈现出类似的不规则模式。这些算法在不同函数差下的相对性能会明显地从高于平均水平转变为低于平均水平，这表明其运行机制可能会因所采用的基于减法的函数结构不同而产生不同响应。

此外，从热图的色彩结构可以明显看出，$f12 - f9$、$f14 - f10$和 $f16 - f13$等函数差对观测到的模式有着显著影响。它们出现在颜色鲜明的暖色区域中，表明这些

函数差会使特定算法组相对于函数均值产生较大的正偏差。这表明，它们是塑造相对性能结构时最具影响力的基于减法的函数之一。另一方面， $f7 - f3$ 和 $f11 - f9$ 等函数差主要分布在较深的蓝色区域，表明某些算法组的相对性能低于平均水平。这说明算法对它们的响应存在差异：一些方法的相对表现更好，而另一些方法的效率则有所下降。

树状图表明，算法自然地聚集成不同的群组。这支持了这样的结论：减法操作后，基准结构仍得以保留。这也使我们能够进一步对算法进行分类，不仅可以根据它们在经典 c.u.p. 设置中的相似性进行分类，还可以结合基于减法的函数差进行分类。

综合来看，图 7 的观察结果表明，算法 $a1$- $a4$ 和 $a5$- $a7$ 在相对性能模式上表现出很强的相似性，而算法$a10$ - $a12$ 和 $a15$ - $a17$ 则可能因基于减法的函数特征不同而呈现不同的行为模式。某些函数差对算法聚类的影响更为显著。这为优化分析中评估它们的有效性提供了有益的见解。

相关性与聚类分析有助于识别出对特定测试函数变换表现出鲁棒性或敏感性的算法群组。例如，算法 $a3$， $a7$和 $a12$ 在代数函数减法的情境下表现相似，这表明它们在解空间中的探索与利用模式相近。相比之下，算法 $a15$， $a19$和 $a22$ 会因所分析的函数不同而呈现截然不同的模式，这可能表明它们针对特定类型的目标函数景观进行了适应。一个关键观察是，即使经过函数减法，c.u.p. 基准测试的结构仍得以保留，这表明该问题空间具有内在的有序性，也说明与无免费午餐定理的假设相反，某些算法可能更适合特定的函数类。

8 测试函数的代数变换对优化算法性能的影响

8.1 差分热图与主成分分析的结论

在本分析中，我们对三个数据集进行了详细研究：Data1（原始数据）、Data2（和占主导）和 Data3（差占主导）。后两者通过对原始函数进行代数重组而得到，所用操作为对原始函数求和与求差。我们使用差分热图和主成分分析 (PCA) 分析了这些数据集，以探究代数运算对函数与算法之间关系的影响。本分析旨在了解在无免费午餐定理的背景下，对函数进行代数运算会如何影响数据结构及算法的行为。Data1 表示表 1 中属于 c.u.p. 集合的原始函数。该集合包括从 $f2$ 到 $f15$ 的函数，不包括 $f1$（完全由零构成）和 $f16$ （完全由一构成）。根据定义，这两个函数对于每种算法的性能值分别为 0 和 1。因此，Data1 包含 14 个函数 $f2 - f15$ 和 24 种算法 $a1 - a24$ 的性能值矩阵。Data1 中的每个函数都是独立的，代表函数与算法之间的原始关系。Data2 是通过将表 1 中的原始函数替换为来自同一 c.u.p. 集合的函数之和或差而创建的。Data2 中的每个函数都是对 Data1 中的函数进行代数运算后得到的结果

。例如：- Data2 中的 $f2$ 是通过运算 $f4 - f3$ 得到的，- Data2 中的 $f3$ 是通过运算 $f8 - f6$ 得到的，- Data2 中的 $f4$ 是通过运算 $f12 - f9$ 得到的，- Data2 中的 $f5$ 是通过运算 $f14 - f10$ 得到的，- Data2 中的 $f6$ 是通过运算 $f2 + f5$ 得到的，- Data2 中的 $f7$ 是通过运算 $f3 + f5$ 得到的，- Data2 中的 $f8$ 是通过运算 $f3 + f6$ 得到的，- Data2 中的 $f9$ 是通过运算 $f13 - f5$ 得到的，- Data2 中的 $f10$ 是通过运算 $f2 + f9$ 得到的，- Data2 中的 $f11$ 是通过运算 $f3 + f9$ 得到的，- Data2 中的 $f12$ 是通过运算 $f3 + f10$ 得到的，- Data2 中的 $f13$ 是通过运算 $f5 + f9$ 得到的，- Data2 中的 $f14$ 是通过运算 $f2 + f13$ 得到的，- Data2 中的 $f15$ 是通过运算 $f7 + f9$ 得到的。

这些运算引入了函数与算法之间的新关系，这可能导致算法性能发生变化。

Data3 是通过用来自同一 c.u.p. 集合的函数的差与求和替换表 1 中的原始函数而创建的。Data3 中的每个函数都是将 Data1 中的两个函数相减所得的结果。例如：

- $f2$在 Data3 中是运算 $f6 - f5$ 的结果，- $f3$在 Data3 中是运算 $f4 - f2$ 的结果，- $f4$在 Data3 中是运算 $f8 - f5$ 的结果，- $f5$在 Data3 中是运算 $f7 - f3$ 的结果，- $f6$在 Data3 中是运算 $f8 - f3$ 的结果，- $f7$在 Data3 中是运算 $f15 - f9$ 的结果，- $f8$ 在 Data3 中是运算 $f4 + f5$ 的结果，- $f9$在 Data3 中是运算 $f15 - f7$ 的结果，- $f10$ 在 Data3 中是运算 $f14 - f5$ 的结果，- $f11$在 Data3 中是运算 $f15 - f5$ 的结果，- $f12$在 Data3 中是运算 $f11 + f2$ 的结果，- $f13$在 Data3 中是运算 $f15 - f3$ 的结果，- $f14$在 Data3 中是运算 $f9 + f6$ 的结果，- $f15$在 Data3 中是运算 $f13 + f3.$ 的结果

这些操作在函数与算法之间的关系中引入了另一层变化。

图8使用两幅差值热图比较了基准函数经过代数修改后的函数-算法关系。面板（a）展示逐元素差值 Data2-Data1。面板（b）展示 Data3-Data1。颜色映射同时编码性能值变化的符号和幅度。红色表示正偏差（修改后的数据集中的数值较高）。深蓝色表示显著负偏差（数值较低）。橙色/黄色表示接近零的小幅偏差。绿色/青色表示中度负向变化。

在面板（a）中，负偏差占主导，并形成连贯的块状区域。对于 $f2.$，观察到最显著的性能下降。一片范围较大的深蓝区域横跨约 $a9 – a24.$。这表明相较于基线，许多算法的表现大幅下降。第二处主要的负向结构出现在 $f9.$。在 $a1 – a16.$ 范围内，它几乎始终呈深蓝色。随后，该行在较后的算法中出现明显的符号反转。在约

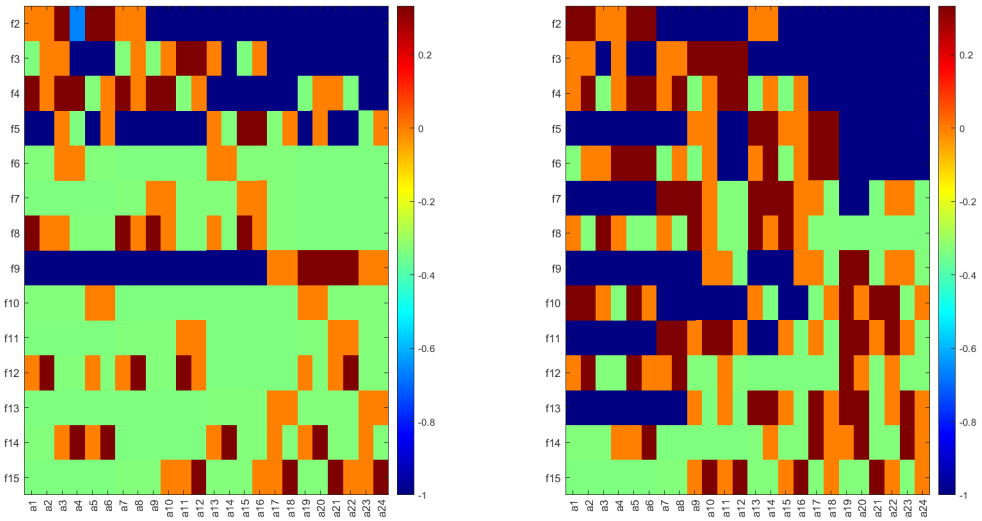

图8：修改后函数的函数-算法关系比较。a）差值热图：Data2 - Data1，b）差值热图：Data3 - Data1

$a19-a22$，差会变为显著的正值（红色）。这表明，同一种代数构造可能会使大范围算法组的表现变差，同时使较小范围的算法子集受益。除这些大尺度块状区域外，许多条目呈现绿色或青色。这与系统性的中度下降相符，而非孤立波动。此外，也可以看到其他严重的局部下降（例如，$f5-a21$ 和 $f5-a22$）。正偏差虽有出现，但仍局限于少数位置。其中包括 $f8-a1$和 $f4-a9/a10.$。此外，$f12$ 也有若干处表现提升（例如，$a2$、$a11$、$a22$），$f14$ 也有多处提升（例如，$a4$、$a6$、$a14$、$a20$）。近零变化（橙色/黄色）相对较少，主要以孤立单元格的形式出现。

面板（b）呈现出更为异质、对比度更高的模式。强负向区域与强正向区域频繁交替。大面积负向区域持续存在。这在 $f2$ 跨越约 $a7-a12$ 以及再次出现在 $a15-a24$.时尤为明显。在 $f5$ 跨越早期算法 $a1-a8.$ 时也能看到这一现象。与此同时，面板（b）包含比面板（a）更大、也更连续的正向区块。这表明依赖配对的重新排序更强。明显的例子包括 $f3$跨越 $a9-a12.$。还包括 $f5$在 $a13/a14$附近，以及 $a17/a18.$。另一个例子是 $f7$在 $a7-a9$附近，以及 $a13-a15.$。较高索引函数与选定的后期算法组合中还出现了其他正向簇。例如，$f10$在 $a19/a21/a22.$附近呈现正向区域。同样，$f13$在 $a13/a14/a17/a19/a20/a23.$ 附近呈现正向区域。对于 $f8$和 $f14.$的部分区域，仍可见中等程度的负向背景区域（绿色/青色）。由于 Data3 主要由差构成，但也包含若干求和，因此面板（b）中的模式应解读为这种代数重组所产生的净效应，而不应解读为纯粹的减法扰动。

图 9 报告了三个数据集的主成分分析：Data1（红色）、Data2（蓝色）和 Data3（绿色）。主成分分析在转置矩阵 Data1、Data2、Data3 上计算。每个点对应一个算法 $a1-a24.$ 每个算法由一个在 $f2-f15.$ 上的 14 维特征描述表示。该图展示主成分 1 与主成分2的得分。主成分分析针对每个数据集分别进行。因此，坐标轴是特定于数据集的正交基。解读依据解释方差、主导载荷以及数据集内的得分结构。

F或者说，Data1 中，主成分1和主成分2分别解释了30.22%和30.22%的方差（总计60.45%）。一个

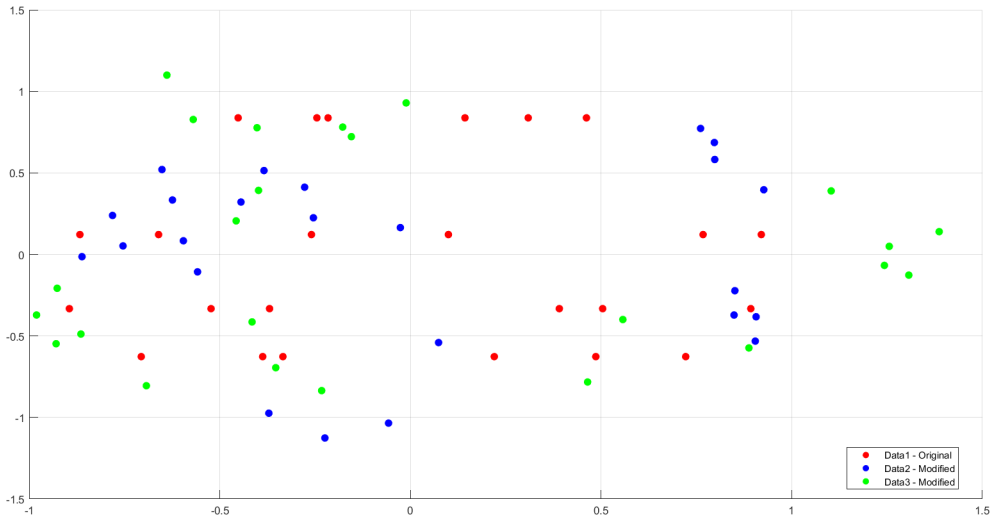

图 9：三个数据集的主成分分析。X 轴：第一主成分（主成分1），Y 轴：第二主成分（主成分2）

显著特征是 PC1–PC3简并。主成分1至主成分3中的每一个都解释了30.22%。 主要变异位于一个三维子空间中。该子空间内部的具体坐标轴并不唯一。因此，我们使用主成分1至主成分3上的平方载荷贡献汇总。贡献最大的是 $f8$、 $f12$、 $f14$、 $f15$.每个变量在主成分1至主成分3上的贡献均为0.439。第二梯队是 $f4$、 $f6$、 $f7$、 $f10$、 $f11$、 $f13$.每个变量的贡献均为0.173。贡献最小的是 $f2$、 $f3$、 $f5$、 $f9$.每个变量的贡献均为0.052。Data1具有很强的低维特征。其主导变异分布在三个同等重要的方向上。

对于 Data2，主成分1解释了 33.99%，主成分2解释了 23.52%（总计 57.51%）。主成分1至主成分3解释了 74.87%。 载荷显示了由求和/差分重组所引起的主要对比。沿主成分1方向，绝对载荷最大的项是 $f9$(0.628)，与 $f3$(0.501) 和 $f2$(0.390) 相对。符号采用惯例约定。因此，主要分离是由 $f9$ 与 $f3$和 $f2$.的联合贡献之间的对比所驱动。沿主成分2方向，主要对比是 $f4$(0.572) 与 $f5$(0.480)。其他不可忽略的项包括 $f12$(0.312) 和 $f3$(0.273)。得分极值反映了这些对比。沿主成分1方向，最极端的算法包括 $a20$ (0.927)、 $a24$(0.906)、 $a18$ (0.904) 和 $a3$ (0.861)。沿主成分2方向，幅度最大值出现在 $a16$(1.126)、 $a15$ (1.035) 和 $a13$ (0.974)。

对于Data3，主成分1解释了32.45%，主成分2解释了19.01%（总计51.46%）。主成分1至主成分3解释了68.31%. Data3在各算法间也具有最大的总体变异性。总方差（特征值之和）从1.053（Data1）增至1.264（Data2），并达到1.992（Data3）。载荷识别出主导驱动因素。沿主成分1， $f2$ (0.507)与 $f11$(0.443)、 $f9$(0.408)和 $f13$(0.325)形成对比。沿主成分2， $f3$(0.580)与$f10$ (0.529)形成对比。进一步的贡献来自 $f4$ (0.360)和 $f6$ (0.261)。得分极值强于Data2。对于主成分1，最大幅度为 $a2$(1.387)、 $a5$(1.307)、$a6$ (1.256)和 $a20$ (0.981)。对于主成分2，最强幅度包括 $a10$(1.100)、 $a8$(0.929)和 $a21$(0.835)。

综合来看，图8和图9表明，代数重组以结构化且不均匀的方式重塑了函数与算法之间的关系。在图8a）中，Data2-Data1主要表现为广泛的块结构化缩减，局部增益较少，并且在特定算法子集中偶尔出现符号反转。在图8b）中，Data3-Data1的对比度更高

且更加异质，显著的性能改善和性能下降集中在不同的子区域中，这与更强的依赖配对的重新排序相一致。图9从全局层面支持了这一点：Data2主要由 $f9$ 与（$f3$、 $f2$）之间的对比驱动，同时存在次要的 $f4$- $f5$对比；而Data3则主要由 $f2$与（$f11$、 $f9$、 $f13$）以及 $f3$与f10。之间的对比主导。总体而言，这些变换有选择地改变了占主导地位的函数层面对比，从而导致算法行为发生数据集特定的重组。这些变换系统地重塑了占主导地位的函数层面对比，形成了数据集特定的算法性能特征重新排序。

8.2 使用方差分析和 Tukey事后检验分析研究结果

本分析旨在确定，对基准函数进行修改是否会显著影响优化算法性能，即使是在排列封闭的函数空间内也是如此。采用单因素方差分析（方差分析） [60] 评估数据集之间的差异。计算 F统计量及相应的 p值，以检验零假设；该假设认为，各数据集中结果的均值之间不存在显著差异。此外，还进行了 Tukey事后检验 [81] ，以深入考察数据集之间的两两差值。方差分析结果见表5。计算得出的 F统计量为110.68，相应的 p值为 3.59965×10^{-44}，显著低于常用阈值（$\alpha = 0.05$）。因此拒绝零假设，表明三个数据集之间的差异具有统计显著性。

变异来源	平方和 (SS)	自由度 (DF)	均方 (MS)	F 统计量	p值
Data	24.533	2	12.2666	110.68	3.59965×10^{-44}
错误	111.379	1005	0.1108	-	-
总计	135.913	1007	-	-	-

表5：方差分析结果摘要

表5报告了单因素方差分析中数据集的主效应极为显著，这表明至少有一个数据集的逐元素性能值均值存在差异。由于方差分析F检验无法确定哪些具体的数据集配对之间存在差异，因此对 Data1、Data2 和 Data3 之间的所有两两对比应用了 Tukey事后检验程序。由此得到的均值差、置信区间和校正后的p值汇总于表6。

比较	均值差	置信区间下限	置信区间上限	p值
Data1 与 Data2	0.32639	0.26619	0.38658	0.0000
Data1 与 Data3	0.33532	0.27512	0.39551	0.0000
Data2 与 Data3	0.00893	-0.05127	0.06912	0.93556

表6：Tukey检验结果

Tukey检验结果的解读

1. Data1与Data2：平均差为0.32639， $p-$值 $= 0.0000$，表明这些数据集之间存在统计学上的显著差异。
2. Data1与Data3：平均差为0.33532， $p-$值 $= 0.0000$，同样证实存在显著差异。

3. Data2与Data3：平均差为0.00893， $p-$值 $= 0.93556$，表明这些数据集之间无显著差。

通过加法和减法修改函数会改变优化问题的结构，从而影响算法效率，尽管这些函数仍属于 c.u.p. 集合。主成分分析 [58] 和方差分析检验表明，相较于原始数据集，这些操作会导致性能出现具有统计显著性的变化，进而造成

算法行为的差异。与严重依赖局部函数结构的算法相比，基于更多探索性策略的算法似乎较少受到这些变化的影响。

8.3 Data1、Data2 和 Data3 的箱线图分析

图10中的箱线图展示了三个数据集的数值分布：Data1、Data2 和 Data3。这些数据集中的每一个都包含通过对测试函数进行不同代数变换而获得的结果，从而可以评估它们对优化算法性能的影响。图中的箱体表示四分位距（IQR），其范围涵盖第一四分位数（Q1）与第三四分位数（Q3）之间的数值。箱体内的红线表示中位数，而须线则显示排除离群值后的数值范围。须线之外的各个点表示离群值，这些值超过 IQR 的 1.5 倍。

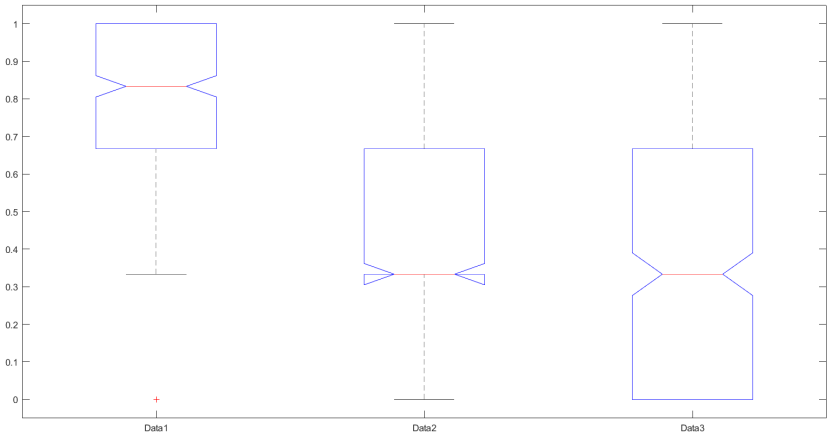

图10：使用箱线图比较三个数据集的分布。X轴：数据集，Y轴：函数值

分析 Data1 可见，该数据集中的数值高度集中在较高值附近，表明针对所测试函数的结果稳定且均匀。四分位距在 0.8–0.9 附近波动，而下界则明显受限。低值区域出现了离群值，这表明在少数情况下，某些函数和算法的性能显著较差。与 Data1 相比，Data2 中数值的离散程度明显更大。中心值（中位数）已移向显著较低的数值，表明该数据集中的算法性能总体有所下降。箱体呈不对称分布，下部较长，说明低值范围内的结果更为分散。须线延伸至接近零的数值，表明某些测试函数会导致算法性能大幅下降。Data3 的分布与 Data2 相似，但数值看起来更加分散。结果几乎覆盖了从接近零到 1.0 的整个范围，这意味着某些算法在选定函数上效率极高，而另一些算法的效果则极低。此外，此处的箱体比 Data2 中的更大，表明结果的方差更大，算法行为的差异也更为显著。四分位距涵盖了极低值和高值，这意味着测试函数的结构经过变换后，会使不同优化方法产生极端结果。比较这三个

数据集可以发现，测试函数经过相应修改后，出现了明显差异。作为原始数据集，Data1 表现出最高的稳定性和相对均匀的数值，这表明原始测试函数集与这些算法相适配，且平衡性良好。Data2 和 Data3 中的函数经过求和与求导运算后，方差显著增大，中位数也明显向较低值偏移。这些数据集中的数值覆盖了从 0 到 1 的整个范围，表明算法与变换后的测试函数之间存在复杂的相互作用，最终导致优化算法的性能不均匀。方差分析和 Tukey检验的结果表明，对测试函数的修改会显著影响算法结果。原始数据集（Data1）与两个修改后的数据集（Data2 和 Data3）均存在显著差异，而 Data2 与 Data3 之间的差异在统计学上不显著。这表明，对测试函数进行这两种修改都会使算法性能产生相似的变化。结合 NFL定理来看，这些结果具有重要影响。即使在置换闭空间中，对测试函数的修改也可能导致 NFL定理不成立。这意味着，测试函数的选择对于优化算法的评估至关重要。

以下定理得到了我们的实证结果支持：

定理1 无免费午餐定理（*NFLT*）在排列下封闭（*c.u.p.*）且均匀抽样的函数空间中全局成立，*Igel* 和 *Toussaint* *[39],[40]*已对此作出形式化证明。然而，对基准函数进行代数修改（例如加法或减法）可能引入结构性偏差，从而违反均匀性假设，并可能导致 *NFLT* 局部不成立——即使是在 *c.u.p.* 空间中也是如此。

推论1 通过代数运算修改的测试函数集中，无免费午餐定理在局部不成立，即使原始结构是在排列下封闭的。

一般来说，无免费午餐定理不适用于所有函数集的子集（除本文外，其他人也研究过类似集合，例如 [40]），这一点似乎很自然；尽管如此，该领域严谨的数学论述仍然颇有意义。无免费午餐定理在某个子集上成立，当且仅当该子集是排列封闭闭包 [16],[40]。

如果情况如此，则可以将无免费午餐定理的性质与集合结构分开考虑，如下所述：

观察1 以这种方式变换后的排列封闭不再是排列封闭。

推论1 根据观察*1*，可以得出结论：对于本文分析的变换后数据集，无免费午餐定理不成立。

8.4 高维蒙特卡洛模拟：方法、结果与比较分析

为了推广针对 $n = 4$ 情况所进行的穷尽分析得出的结论，我们采用蒙特卡洛方法 [25]，针对更大的问题规模开展了广泛的实证研究： $n = 10$、 $n = 30$、 $n = 50$ 和 $n = 100$。这些实验旨在确定：在复杂度呈指数级增长、无法进行完整枚举的空间中，基于排列的采样顺序与目标函数优化性能之间的结构关系是否依然存在。

根据本研究前文确立的假设和立场，对于更大的 n 值，采用了具有特定结构特征的二元函数 $f\colon\ \mathcal{X} \to \{0,1\}$ 的代表性采样。研究考察了三类函数：平衡函数，其零和一的数量大致相等；强不平衡函数，其至少 91% 的输出值相同；以及对称函数，其输出仅取决于输入的汉明重量。对于 $n = 10$ 和 $n = 30$，分析了 50 个函数（每类约 16 至 17 个）；对于 $n = 50$，研究了 15 个函数（每类 5 个）；对于 $n = 100$，则考察了三个函数，每类各一个。鉴于函数空间极其复杂，且其规模随定义域大小呈双指数级增长（对于 $n = 100$，达到 $2^{2^{100}}$），这种选择是在代表性与计算可行性之间所作的折中。

蒙特卡洛方法遵循文献中的方法学指南实施 [21], [25], ，确保在大型函数空间的背景下控制估计量的方差并保证样本的代表性。

在每次实验中，按照 $n = 4$ 案例所采用的方法，使用预算为 $m = 4$ 个点的不重复抽样。对于 $n = 10$，测试了全部 24 个排列；对于 $n = 30$，选取了 10 个具有代表性的排列；对于 $n = 50$，使用了 6 个排列；对于 $n = 100$，应用了 3 种算法。对于每个函数，记录找到的最佳结果，并计算每种算法的平均效率。

对于 $n = 10$，算法之间的差异清晰可见。对于平衡函数，最佳算法与最差算法之间的性能差异（最佳结果的均值）超过了 18%。对于对称函数，某些排列产生的结果比其他排列好 12%—15%。强不平衡函数的方差要小得多——几乎所有排列都产生了接近最优的结果，结果范围为 35.4% 至 62.1%，平均约为 49.2%。对于平衡函数，结果范围为 46.7% 至 78.2%；对于对称函数，结果范围为 58.9% 至 81.3%，其中探索性排列明显占据优势。

对于 $n = 30$，观察到了类似的模式：表现最佳与最差的排列之间的性能差距达到 22 - 30 个百分点。平衡函数的平均效率为 63.4%（范围：52.3% - 75.5%），非平衡函数为 49.5%（40.1% - 58.8%），对称函数为 70.6%（60.2% - 84.0%）。在这一设置下，探索性排列的高性能同样十分明显。

对于 $n = 50$，由于样本量有限（15 个函数、6 种算法），该分析作为一项更具针对性的探索性研究展开。结果仍显示出显著的方差。对于对称函数，表现最佳与最差的排列之间的差异达到 21 个百分点；对于平衡函数，为 15 个百分点；对于非平衡函数，约为 6 - 8 个百分点。平均效率分别为 66.1%、59.2% 和 46.5%，这与低维度下观察到的趋势一致。

对于 $n = 100$，我们分析了每个类别中的一个函数，并分别使用三种不同的算法进行了测试。对称函数的排列之间差异达到 19.4%，平衡函数为 12.6%，非平衡函数为 6.3%。尽管样本量较小，观察到的趋势仍保持一致，且采样结构继续对最终结果产生显著影响。

这项研究证实，采样顺序与算法性能之间的关系并非仅仅是 $n = 4$ case 中有限空间造成的假象，而是一种在更高维场景下依然存在的结构性现象。算法效率取决于函数类以及所用排列的类型。这些结果对经典无免费午餐定理的普遍有效性提出了质疑，并表明在实际优化场景中，算法的结构和目标函数的性质发挥着关键作用。对称函数和平衡函数

对采样策略尤其敏感，这为更精确的刻画和设计专门的优化策略带来了可能。借助对具有明确结构性质的函数进行选择，蒙特卡洛方法证明是算法分析的有效且可靠的工具，尤其适用于无法进行完整枚举的情况。所得结果是深入理解大型解空间中算法行为的重要一步，既支持了早期观察结果的推广，也为自适应启发式算法的进一步研究提供了经验基础。因此，本文的分析不仅证实了在小型函数空间中观察到的现象具有可扩展性，也提供了一种方法论研究范式，未来可将其扩展到其他函数类乃至更高维度。

8.5 结论与未来研究方向

本研究为探讨代数运算对基于排列的优化算法中测试函数的影响提供了创新视角。结果表明，在排列封闭结构中对二元函数进行加法和减法，都会导致算法性能呈现系统性相关，这挑战了无免费午餐定理在受限局部情境下的经典假设。以往的启发式优化分析方法并未考虑测试函数空间中的结构依赖关系，而是依赖结果完全随机这一假设，这表明有必要重新审视这些概念。

此前，Clerc [16]和 Whitley [87], 的研究讨论过函数结构对算法性能的影响，但他们的分析主要集中在单个测试函数的性质及其对解空间探索过程的影响。Whitley 指出了与目标函数景观相关的启发式方法的根本局限，但未将代数变换作为评估算法的分析工具。Clerc 在其研究中强调了优化问题中的结构依赖关系，却未将其与测试函数的系统性修改联系起来。只有通过我们的研究，我们才展示了代数运算如何引入新的分析维度，从而识别出传统评估方法此前未能发现的算法性能模式。

Kimbrough 等人的研究也提出了类似尝试，旨在挑战NFL定理的普适性。他们表明，特定的问题类别（例如由可行-不可行双种群（FI-2Pop）遗传算法处理的约束优化问题）可能违反NFL 假设。他们的研究聚焦于约束对算法效率的影响，证明问题结构可能有利于某些启发式算法。我们的分析延续了这一研究方向，表明测试函数中的结构依赖关系也会导致结构化基准中算法效率呈现非均匀分布。结果表明，某些算法在代数相关函数上表现出系统性的性能模式，这意味着，对于实际优化问题而言，完全随机的性能假设可能存在缺陷。这与文献中的发现一致：该文提出了一个目标值置换不符合标准MAX-2-SAT问题的MAX-2-SAT 问题。这一结果表明，在$n=3$的情况下，MAX-2-SAT并不满足排列封闭（c.u.p.），并且还为$n=4$的情况提供了证明提示。此前，MAX-SAT问题已在 [39], [77], 和 [85], 等研究中得到探讨，进一步说明问题特定的结构可能导致偏离一般理论假设的现象。

未来的研究应着重探索乘法代数运算对测试函数的影响，这可能会揭示更多层次的结构依赖关系

并确定优化算法的性能。将分析扩展到更高阶变换，包括测试函数的多项式组合或非线性组合，可能会进一步完善算法分类，以评估其对特定问题类型的抵抗能力。此外，应用机器学习方法，根据测试函数的代数性质预测启发式算法的性能，可能有助于开发新的自适应优化策略。

归根结底，本研究为启发式优化开辟了新的视角，提供了一种超越标准统计随机性模型、能够捕捉测试函数空间深层结构依赖关系的算法分析方法。更精准地将算法与特定问题类别相匹配的能力意味着，未来的研究有望更准确地界定NFL定理的有效范围，并识别出测试函数中的结构依赖关系会决定最优启发式算法选择的情形。

9 影响

本文报告的结果表明，置换结构和目标函数的重新表述能够系统地影响算法行为。这对于依赖重新标记、重抽样或基于置换的搜索的推断与优化过程尤为重要。在这类情形下，评估顺序和对称性会影响早期阶段的信息获取。下文将重点讨论其对统计学和进化计算的影响。讨论强调了受结构约束的函数类（包括排列封闭类）以及目标函数的算术结构，这些因素可能导致置换策略之间出现一致的重新排序。

9.1 对统计学的影响

无免费午餐定理对统计学具有重要影响，尤其体现在针对特定分析任务设计和应用算法方面。本研究聚焦于置换算法——这类方法通过置换数据集或目标函数中的元素来运作——并考察它们在违反NFL定理假设的条件下的行为。具体而言，研究分析了函数子集排列封闭类及其对算法性能的影响。下文将结合具体的统计学应用讨论研究结果：

9.1.1 多元统计学中的排列显著性检验

在多元统计学中，尤其是在基因组学或神经影像学等领域， [25],[59],[67]假设检验常用于考察组间差异。传统参数检验，如学生 t 检验 [63] 或方差分析 [60], ，依赖于正态性和方差齐性的假设，而这些假设在实践中经常不成立。本研究所呈现的结果表明，对于满足排列封闭条件（即结果的分布在排列下封闭）的数据，置换算法可能提供一种更稳健的替代方案。在置换检验中，通过反复打乱组标签并重新计算检验统计量来生成经验分布，以评估零假设。

我们使用 24 种置换算法 $a1$ - $a24$ 在人工函数集上进行的实验表明，在受限问题子集中，各算法的性能存在显著差异。例如，对于测试函数 $f14$ （由两个较简单的函数 $f5$和 $f10$相加构成，其中 $f5$代表光滑单峰凸结构，而 $f10$则引入不连续性和噪声），某种算法 $a4$ 找到最优值的速度显著快于分别独立评估各个组成函数的情况。这表明

利用 c.u.p. 集合的结构性质（例如，加性效应所形成的更丰富、更具区分度的优化景观），能够提高对真实差异的敏感度。同样，在经验置换检验中，针对特定代价函数（例如平方差和）优化的算法，在数据呈现有序结构时，能够更有效地检测组间的细微差异。采用此类定制的置换算法，既能提高检验效能（即检测真实效应的能力），又能降低第一类错误的风险。

9.1.2 通过排列优化回归估计量

回归分析 [42],[44] 常会遇到离群值或异方差性带来的困难，这些因素会破坏经典普通最小二乘法（OLS）方法的假设。尽管存在最小绝对偏差法（LAD）等稳健方法，但在某些数据配置下，这些方法也可能效率低下。本研究提出了一个优化回归估计量的框架，重点考察置换数据子集，并分析目标函数的算术性质。我们的结果表明，利用无免费午餐定理在 c.u.p. 集合上的违背现象，可以改进参数估计。

例如，考虑收入分布呈现“厚尾”（显著偏离正态性）的金融数据。与其使用经典 OLS，不如采用一种置换算法，重新排列残差，以最小化一个经过适当构造、能够反映这种结构的损失函数（例如，对较大误差施加更重惩罚的代价函数）。我们的实验证实了这一方法的有效性：复合函数（例如，$f14 = f5 + f10$）上的表现并非其各分量结果的简单加和。换言之，两个特征的累积效应可能形成一种优化景观，使某些方法更具优势，其程度超出根据各自单独表现所能预期的范围。这种协同效应能在经典假设失效时带来更准确、更稳定的回归估计值。因此，置换策略能够产生更精确且更易解释的模型，尤其适用于数据质量参差不齐或离群值较多的数据集。

9.1.3 排列自助法

自助法技术 [10],[19],[67] 是估计采样分布和构建置信区间的基本工具。然而，传统的自助法程序计算成本可能很高，也未必能充分利用数据结构。我们的研究结果表明，基于置换的自助法可以针对排列封闭子集进行优化，在保持统计准确性的同时降低计算复杂度。也就是说，在受限问题集中的 NFL 假设违背，使算法能够聚焦于信息量最大的排列。

例如，在时间序列分析中，区块自助法会对连续的数据区块进行抽样，以保留时间依赖性。还可以使用遵循数据顺序结构的置换算法（即避免打乱自然分组）来增强这些方法。我们的分析表明，某些置换算法（例如，以系统搜索为特征的$a2$ - $a4$ - $a6$ 组）能够以更少的迭代次数生成有代表性的排列，从而降低自助法中的计算负荷。这使得我们能够以显著更低的计算成本，可靠地估计复杂数据集（例如高频金融序列或环境数据）的置信区间。

9.1.4 混合效应模型中的置换检验

用于分析层级数据结构（例如教室中的学生、医院中的患者）的混合效应模型 [41], ，在检验固定效应和随机效应的显著性时面临挑战，因为这类模型复杂，且对分布假设敏感。我们的研究结果表明，在混合效应模型中，可以通过使用 c.u.p. 集合构建检验统计量的经验分布，从而有效应用置换检验。

关键思路是在自然分组内（例如，在每个课堂或医院内）置换数据，而不是在整个数据集范围内进行置换。这能保留观测值之间固有的依赖结构，并避免违反模型假设。例如，在评估一种新教学方法的教育研究中，可以通过在各班级内置换学生分配，检验课堂层面效应的显著性。我们的实验也揭示了尊重结构的类似益处：具有相似属性的函数形成了同质簇（例如， $f1$、 $f2$和 $f5$表现出相似的性能特征），这支持了某些算法组行为的一致性。这表明，在混合模型中，受限的组内置换比传统方法能产生更可靠的检验结果，同时无需严格的参数假设。

9.1.5 多变量数据中的非线性依赖分析

识别变量之间的非线性依赖 [17],[32],[46],[75] 是多变量分析中的常见挑战。当依赖关系呈非线性时，皮尔逊相关等标准关联度量可能无法检测到这种关系。本研究提出的置换算法通过有意引入排列，并观察其对目标函数（例如，互信息或距离度量）的影响，为探索非线性依赖提供了强有力的工具。这种方法能够揭示数据中隐藏的模式。

在我们的实验中，通过排列来组合或修改测试函数，揭示了单独分析时并不明显的依赖关系。例如，将两个函数 $f5$和 $f10$相加形成 $f14$，会改变优化景观，使算法的响应不同于分别评估各个分量时的情况。这种变化表明分量之间存在非线性交互——这类似于统计数据：置换某个变量后，准则函数发生显著变化，意味着该变量与其他变量之间存在非线性关系。此外，我们的相关性和 PCA 分析（见第 8.1 节）表明，引入此类非线性变换（例如，将函数相加或相减）会在算法输出中产生新的变异方向。同样，基于置换的程序可以检测到线性关联度量可能遗漏的变量间非线性关系。如 [78], 所示，对经验时间序列进行非线性短期预测，可以揭示确定性混沌的特征，并将其与采样/测量误差及外部诱发噪声区分开来。对于环境预测变量（例如温度和降水），我们通过打乱变量并监测目标准则（例如互信息）的变化，来探查非线性依赖关系。

9.1.6 贝叶斯统计学

贝叶斯统计学基于随着新数据的出现而不断迭代更新概率分布 [7],[26],[29],[84]。与频率学派方法不同，贝叶斯方法将先验知识以先验分布的形式纳入分析，再通过似然函数更新该分布，从而得到后验分布。这种方法具有相当大的建模灵活性——既适用于基于小样本的分析，也适用于复杂的层次模型——但贝叶斯方法的有效性在很大程度上取决于

先验分布的恰当选择，以及用于探索参数空间的算法的效率（例如，马尔可夫链蒙特卡罗（MCMC）采样的效率[64],[21]）。本研究的结果表明，有意识地纳入问题结构（例如，将函数空间限制在特定类别内）可能使某些分析策略优于其他策略。换言之，利用额外的（结构性）知识——这与先验分布在贝叶斯范式中的作用类似——可以打破无免费午餐定理的普遍性假设，并改善特定问题类别的推断结果。

本研究提出的置换策略能够更有效地使模型假设与数据的实际结构相匹配，从而增强贝叶斯建模。当数据呈偏态或重尾时，高斯假设可能较为脆弱，因此使用重尾先验和/或似然（例如 Student-t）可以提高稳健性。例如，当数据表现出明显的偏态或厚尾（例如由离群值或缺失值导致）时，标准先验假设（如经典高斯先验 [92]）可能并不适用。相反，应采用能够反映观测结构的先验——例如非对称或重尾分布——以改善模型拟合和结果解读。我们的分析证实了这一需求：函数修改后各算法的性能分布呈现极大差异，取值范围几乎覆盖从接近 0 到 1 的全部区间；部分算法的效率极低，而另一些则表现出色。这种分散程度（图10中可见离群值）表明数据存在异质性，并对应着重尾的性能分布。因此，在贝叶斯分析中，应使用能够捕捉这种变异性的先验分布（例如重尾先验，而非正态先验），以避免产生误导性假设并提高推断可靠性。

置换 *MCMC* 策略与参数分组

我们的相关图和层次聚类结果揭示了基准测试中明显的结构簇（见图2—3；变换后的基准测试见图4—7）。这种结构表明存在强依赖关系；在贝叶斯推断中，分块（分组）MCMC 更新以及扫描顺序的选择可以改善混合并减少自相关 [64],[21]。例如，函数 $f1$、$f2$ 和 $f5$ 构成一个同质组（图3）。复合函数 $f14 = f5 + f10$ 表明代数结构会改变性能格局：从经验结果看，$M(a4,\ f14) \neq M(a4,\ f5) + M(a4,\ f10)$。最后，有些算法的表现更加异质（例如 $a15$ - $a17$ 组），而另一些算法与大多数方法的相似性较弱（例如 $a9$），这支持了这样一种观点：该基准测试可分解为若干强相关组。

层次模型中的应用

我们的相关图和树状图（见图 2 - 7）表明，基准测试中的性能依赖并不均匀，而是集中在一些内聚的群组中。在后验依赖较强的贝叶斯推断中，也会出现类似情况。在这种情况下，分块（分组）更新和扫描顺序的选择可以改善混合并降低自相关 [64]。受这一类比的启发，我们提出一个预处理步骤：在运行 MCMC 之前，使用基于相关性的聚类来定义参数块。请参阅上方的 $f14 = f5 + f10$ 示例，具体了解结构耦合如何改变效率。

蒙特卡洛模拟与算法效率

蒙特卡洛实验（参见第 8.4 节）表明，对采样顺序的敏感性

在更大的问题规模 $n = 10$，30，50，100 下依然存在，并且高度依赖于函数类。对于 $n = 30$，最佳与最差采样顺序之间的差距为 22–30 个百分点；平衡函数的平均效率为 63。4%，强不平衡函数为 49。5%，对称函数为 70。6%。这一经验规律促使我们将考虑结构的采样方法引入贝叶斯计算，因为已知分块更新和扫描顺序会影响混合与自相关 [64]。

9.2 对进化算法的影响

本研究表明，NFL 等式依赖于特定条件。具体而言，这些等式假设对函数类进行均匀平均，并满足置换闭包。当这些条件不成立时，搜索策略可能会以基准依赖的方式呈现系统性差异 [15],[38],[39],[50],[51],[77],[85],[89]。这会直接影响进化算法的设计与应用。排列封闭函数类表明了 NFL 式平均情形等价关系成立的条件。然而，我们的结构化基准测试和代数目标重组会使不同算法的排名持续发生变化。它们还揭示了复合目标上的非加性效率，尤其是在利用目标结构（例如对称性和组合结构）时。主成分分析（见第 8.1 节）证实，函数变换后数据结构发生了明显变化。即使是简单的加法或减法，也会显著重组函数-算法关系。下文将讨论进化算法的关键实用指南。

9.2.1 设计算子

算法结果的相关性与层次聚类分析均表明，遗传算子（变异、交叉、选择）的设计应考虑问题的结构 [15],[51] （例如，参见 [37]中的应用实例）。在所开展的实验中，观察到若干行为几乎相同的算法组——例如，算法 $a1$ 与 $a2$ 在全部测试函数上的结果呈现出近乎完美的相关性。此类算法对（仅存在细微的启发式变更，例如搜索顺序不同）取得了相似结果，这表明对算子的某些修改并不会带来显著的性能变化。换言之，若某一进化算子未能利用目标函数景观的特定结构，则该算法相对于采用相似操作方案的另一算法可能是冗余的。

另一方面，这些结果凸显了设计对目标的代数变换具有稳健性的算子的必要性。在Data2中，负偏差占主导，并常常形成块状区域（例如， $f2$ 在大约 $a9$ - $a24$，范围内出现显著下降， $f9$ 在 $a1$ - $a16$，上多为负值，随后在 $a19$ -$a22$附近发生符号反转），并伴有严重的局部退化，如 $f5$ - $a21$ 和 $f5$ - $a22$. 。正偏差虽然存在，但通常较为局部，包括 $f8$ - $a1$ ，以及 $a2$ 在 $f12$ 上的增益（还有 $a11$ 和$a22$），以及 $f14$ 上的多处增益（例如 $a4$， $a6$， $a14$， $a20$）。面板8b）（图8）更为异质，并包含更大的连续正值区域，表明更强的成对依赖重排序；应将其解释为代数重组的净效应，而非纯粹的减法扰动。这些模式促使人们设计能够在结构化目标重组下保持搜索有效性的算子。

因此，在设计进化算法的算子时， [15],[69],[70], 考虑目标函数的代数性质是值得的。我们的结果表明，问题的结构（例如排列对称性或目标的可加性）会影响算子的有效性——例如，将两个函数相加可能会降低景观的对称性，从而使算法更容易区分优劣解

。这表明，保留有利排列片段的交叉算子，或与表示方式相匹配的变异算子，可能有助于收敛。例如，在旅行商问题等排列问题中， [76], 定制的交叉算子可以保留具有潜力的城市子排列，在探索新配置的同时保留良好的局部结构。对算法结果协方差的分析也得出了类似结论：会形成紧密的群组（例如， $a1$- $a4$和 $a5$- $a7$），而一些算法在变换下表现出更依赖上下文的行为（例如， $a15$ - $a17$），另一些算法的正负协方差模式之间则呈现不规则转换（例如， $a20$ - $a22$）。这些观察表明，应优先采用能够在目标的代数变换下保持性能稳定的算子，尤其是在变量优化或多组件优化场景中。

9.2.2 多目标进化算法（MOEA）

尽管本研究采用以效率为导向的性能度量，在单目标设置下评估基于排列的基准测试，但研究结果对于多目标进化算法（MOEA）仍有参考价值（例如参见 [5],[22]）。在问题表述和分析过程中，目标准则经常会被聚合、分解或以其他方式进行代数重组。在 MOEA 研究中，通常使用超体积和反世代距离（IGD）[79] 等指标评估近似集。

我们的实验表明，即使在同一个排列封闭函数空间中，对目标函数进行简单的代数组合（加法/减法）也会显著影响算法性能。对三个数据集（Data1、Data2、Data3）中逐元素的性能值进行单因素方差分析，证实了这一效应，结果为$F = 110.68$，其中 $p = 3.59965 \times 10^{-44}$ （表5）。Tukey事后检验（表6）显示，Data1与Data2之间（平均差 $= 0.32639$, $p = 0.0000$）以及Data1与Data3之间（平均差 $= 0.33532$, $p = 0.0000$）存在统计学显著差异，而Data2与Data3之间的差异不显著（平均差 $= 0.00893$, $p = 0.93556$）。因此，原始基准测试集与两个代数组合后的数据集均存在显著差异，而这两个重组数据集在该检验下无法区分。变换后的数据集由原始的排列封闭函数通过代数运算生成，同时保持了置换闭包。尽管如此，函数-算法关系仍发生了显著变化（见图8）。Data2由求和与求差混合生成（例如 $f6 = f2 + f5$和 $f5 = f14 - f10$）。相比之下，Data3主要由差构成，但仍包含若干求和项（例如 $f2 = f6 - f5$和$f12 = f11 + f2$）。这些观察结果对多目标进化算法（MOEA）具有实际启示。用于组合各项准则的具体代数形式可能会改变搜索的实际难度，也可能改变在特定问题表述下哪些策略看起来最有效。例如，这包括将分量聚合起来而非分别处理，或通过结构化重组重新表述目标。根据问题表述的不同，这些变化还可能改变目标空间中的支配关系。最后，算法相关矩阵（见图2/6）和基于协方差的层次聚类热图（见图3/7）表明，不同算法族对这类变换的敏感度各异。高度凝聚的群组（例如 $a1$ - $a4$ 和 $a5$ - $a7$）表现出高度一致的行为。其他群组在基于减法的修改下则呈现更依赖上下文的模式（例如 $a15$ - $a17$）。一些子群相对于主导簇表现出反向协方差（例如 $a10$ - $a12$）。对于MOEA设计，这意味着应采用在目标表述发生结构化变化时仍能保持有效的算子和选择机制。当存在排列对称性时，也有必要纳入对称性感知变异或评估。这可以减少在等价解上进行的冗余搜索

表示。

9.2.3 使用置换算法进行超参数优化

超参数优化（HPO）是机器学习中的一项核心任务。其目标是在评估预算有限的情况下，找到能使模型性能最大化的配置[2],[6],[28]。在实践中，对于大型或离散设计空间，人们常会考虑进化搜索和基于种群的搜索策略。这包括神经网络架构设计相关的问题 [54],[96]，也包括离散空间或基于置换的空间中的结构化配置选择 [30],[51],[86]。一种尤为相关的HPO情形是，配置中的一部分可以自然地表示为排列。在这种情况下，分量的顺序十分重要，并且存在对称性。

在上述实验设置中，我们旨在建立一个受控的实证基础，以了解纯置换搜索结构如何影响优化效率。我们分析了24种基于排列的算法，它们共享相同的机制，唯一的区别在于评估一组固定候选点的顺序（表2）。这种设计将评估顺序 作为唯一的算法自由度，从而隔离其影响。实验采用的性能度量是归一化效率指数$E_{a_i,\ f_j}$（见第4.2节）。该指数根据首次达到已知最优解的试验次数定义，即公式（1）。它直接反映了搜索过程中多早能够获得有用信息。

对于具有置换结构的超参数优化而言，有两项观察尤为重要。首先，相关性和聚类分析揭示了若干高度内聚的算法族，其性能特征高度一致。其中，$a1$ - $a2$之间几乎完全相似，而 $a12$ - $a16$之间则存在较强关联。这表明，置换评估顺序的细微变化，既可能导致实际上冗余的行为（高相关性），也可能带来截然不同的探索/利用轨迹（相关性弱或为负）。在超参数优化实践中，这支持采用基于相关性和聚类的诊断，避免维护多个实际上可互换的算法变体；同时也支持在设计集成或混合算法时，选择真正互补的搜索顺序。

其次，研究表明，目标函数的代数重组可以系统性地重塑函数与算法之间的关系。即使在同一个排列封闭的函数空间中，也会发生这种情况。基准变换 Data2 和 Data3 由 Data1 通过结构化求和与差构造而成。例如，在 Data2 中：$f13 = f5 + f9$。例如，在 Data3 中：$f2 = f6 - f5$。这些变换保留了置换闭包。差分热图（图8）显示了广泛且有结构的性能变化，包括块状性能下降和局部增益。方差分析与 Tukey 事后检验也证实了显著的数据集效应（见表5–6）。对于超参数优化（HPO），这提供了一个具体的警示：看似简单的目标函数重构（例如，添加惩罚项、改变标准的组合方式，或用结构化重组替换基础目标函数）可能会改变实际的搜索景观。即使底层对称性类别保持不变，这些改动也可能改变效率最高的策略排序。

这些结果促使我们采用显式利用对称性的置换感知型超参数优化设计。这可以减少对等价表示的冗余探索。它们还促使我们优先重视对结构化目标函数重构的稳健性，因为目标重组会显著改变哪些搜索顺序是高效的（见图8；另见表 5–6）。最后，实证依赖关系分析（相关性和聚类）可用于选择非冗余的算法变体，并组装更具互补性的搜索组件。

9.2.4 二次指派问题（QAP）

二次指派问题（QAP） [47] 是一个经典的NP难排列问题。其目标是将 n 个设施分配到 n 个位置，同时最小化交互成本。目标函数通常表示为成对流量与由指派排列所诱导的相应距离之积的和。搜索空间的规模为 $n!$，这使得QAP对通用启发式方法颇具挑战。从无免费午餐定理的观点看，当性能在全部目标函数 [38],[39],[43],[44],[50],[56]上取平均时，没有任何优化器是一致最优的。因此，系统性优势必须与目标问题类的结构规律性相联系。我们受控的c.u.p. 基准测试支持这一实践性解释。相关性和聚类揭示了近冗余搜索顺序的若干族（见图2—3）。它们还揭示了离群行为，例如$a19$、 $a21$和 $a23$，这些行为偏离主导簇，并与针对特定景观类型的专门化而非广泛稳健性相一致。目标的代数重构显著重塑了函数-算法关系（见图8）。由此得到的差分模式表明，同一种结构化重组可能使广泛的算法组受损，同时使更窄的子集受益。这与结构化变换下出现局部性能不等式相一致，尽管针对完整函数类的全局无免费午餐定理陈述仍然成立。此外，目标分量的结构可加性并不意味着搜索代价具有可加性。在我们的实验中，即使某个函数可以写成逐点和（例如， $f14$ 产生于二元排列封闭构造中的$f5 + f10$），性能度量 M也未必满足$M(a, f_i + f_j) = M(a, f_i) + M(a, f_j)$。这种非可加性与如下解释相容：组合会改变景观中的对称性与可区分性。这些观察促使形成面向QAP的算法设计原则。QAP成本可分解为许多成对交互，因此算子与模型可以显式利用排列结构及交互模式。与此同时，它们应对结构化目标函数重构保持稳健，因为即便简单的代数重组也可能重新排定搜索顺序的优劣。一个有原则的方向是采用指派感知型概率模型，例如用于QAP的估计分布算法中的双随机矩阵（DSM）模型 [68]。

9.3 结论

无免费午餐定理是关于平均情况的结论。在有限函数类上进行均匀平均，且搜索不重复访问时，所有优化器的期望表现都相同 [89],[90]。在经过强化的有限定义域表述中，采用均匀权重时，当且仅当函数类在排列下封闭（排列封闭，c.u.p.）[39]，这一等价关系才成立。我们的完全枚举 $n = 4$ 基线按构造即为排列封闭。与此一致，表2所列的全部24种评估顺序在Data1中的平均效率相同（表4）。无免费午餐定理约束的是函数类上的均匀平均；它并不意味着每个单独目标上的表现都相同。我们的贡献在于刻画超越这一平均值后仍有哪些信息具有价值。保持结构的代数目标函数重构（求和与差分）系统地重组了函数与算法之间的依赖关系。它们揭示了稳定的、近乎等价的评估顺序族，以及对变换敏感的变体。我们还观察到搜索开销并非可加：不能仅凭 f_i 和 f_j 上的表现推断 $f_i \pm f_j$ 上的表现。实际上，目标函数的制定方式和基准测试的构建方式决定了有效的测试机制，因此应视为算法设计的一部分，而非中性的预处理。除了优化基准测试外，只要任务分布受对称性或结构性限制所塑造，同样可以运用无免费午餐定理的视角。这包括基于排列的推断和随机模拟。在这些情形下，所诱导的问题族可能使评估顺序和目标耦合在实践中产生重要影响

具有实际重要性。

本文有望推动启发式优化数学基础领域的后续研究，推荐给从事优化和进化算法研究的科学家、工程师及从业者阅读。

本研究结果所依据的数据，可在提出合理请求后向通讯作者获取。

本研究中的所有数值计算、数据处理和统计可视化均使用 **MATLAB R2023b** 在一台戴尔计算机上完成（系统规格：Windows 10 家庭版，版本 22H2）。

致谢

作者谨向波兰科学院计算机科学研究所（波兰华沙）的 Sławomir T. Wierzchoń 教授致以诚挚谢意，感谢他最初给予的启发与鼓励，帮助作者确立了这一研究方向；感谢他引起作者对 M. Clerc 相关工作的关注，并感谢他对稿件的实质内容和编辑呈现提出的深思熟虑的意见与建议，使稿件更加清晰。

参考文献

[1] S.P. Adam、S.-A.N. Alexandropoulosand、P.M. Pardalos 和 M.N. Vrahatis，无免费午餐定理：综述，《近似与优化：算法、复杂性与应用》，施普林格，57-82，（2019），https://doi.org/10.1007/ 978-3-030-12767-1_5.[2] Y. Ahmed 等，优化光催化染料降解：用于预测受污染水中亚甲基蓝的机器学习与元启发式方法，《工程成果》25，103-538，（2025年），https://doi.org/10.1016/j.rineng. 2024.103538。[3] B. Zolghadr-Asli，无免费午餐定理：摘自水资源规划与管理计算智能的一页，《环境科学与污染研究》30.20，57212-57218，（2023），https://doi.org/10.1007/ s11356-023-26300-1。[4] A. Auger、O. Teytaud，连续午餐是免费的！，《第9届遗传与进化计算年度会议论文集》，916-922，（2007），https: // doi.org/10.1145/1276958.1277145。[5] S. Babaei 等，使用 *NSGA-II* 和 *MOPSO* 对电信网络绿色基站选址进行多目标优化，《算法》18(1)，9，（2025年），https://doi.org/10.3390/a18010009。

[6] J. Bergstra、Y. Bengio，用于超参数优化的随机搜索，《机器学习研究期刊》13(2)，281–305，（2012），http://jmlr.org/papers/ v13/bergstra12a.html。[7] J.J. Bon、A. Bretherton、K. Buchhorn 等，《身处2020年代的贝叶斯学者：现代应用贝叶斯统计实践中的机遇与挑战》，《英国皇家学会哲学汇刊A辑：数学、物理与工程科学》381(2247)，英国皇家学会，（2023），https://doi.org/10. 1098/rsta.2022.0156。[8] I. Boujnah 等，迈向精准螺栓连接设计：基于机器学习的参数预测初探，《设计学会会议录》5，3211-3220，（2024年），https://doi.org/10.1017/pds.2025.10335。[9] H. Cai 等，基于夏普利值的类激活映射：提升神经网络的可解释性，《视觉计算机》41，7249–7267，（2025年），https: //doi.org/10.1007/s00371-025-03803-1。[10] A. Canty、B.D. Ripley，*Boot*：*Bootstrap R*（*S-Plus*）函数，R软件包版本1.3-28.1，（2022），https://cran.r-project.org/web/packages/boot/boot.pdf。[11] J. Ceberio、V. Santucci，排列问题的基于模型的梯度搜索，《ACM进化学习与优化汇刊》3(4)，1-35，（2023），https://doi.org/10.1145/3628605。[12] J. Ceberio、E. Irurozki、A. Mendiburu 和 J.A. Lozano，*Mallows*和广义*Mallows*分布估计算法中距离度量的综述，《计算优化与应用》62(2)，545-564，（2015），https: //doi.org/10.1007/s10589-015-9740-x。[13] P. Chitnelawong 等，《分布式系统中避免聚类漏洞没有免费的午餐》，《科学报告》14(1)，12789，（2023），https://doi.org/ 10.1038/s41598-024-63278-3。[14] D. Choudhury 等，融合边际主义与平等主义：可转移效用博弈的新价值，（2022），https://doi.org/10.48550/arXiv.2201. 09182。[15] V.A. Cicirello，排列问题进化算子的综述与分析，载于《第15届计算智能国际联合会议论文集》，意大利罗马，第1卷，288-299页，（2023），https://doi.org/10.5220/ 0012204900003595。[16] M. Clerc，迭代优化——何时存在最优算法及其他令人沮丧的定理，（2021），在线发表于 https://hal.science/hal-03230263。[17] M. Clerc，距离越近越好，（2007），在线发表于 https://hal.science/ hal-00137320v2。[18] C. Croux、C. Dehon，斯皮尔曼相关与肯德尔相关度量的影响函数，《统计方法与应用》19(4)，497-515，（2010），https:// doi.org/10.1007/s10260-010-0142-z。[19] A.C. Davison、D.V. Hinkley，载于：自助法及其应用，《剑桥统计与概率数学丛书》。剑桥大学出版社，（1997），https://doi.org/10.1017/CBO9780511802843.001。

[20] *W.A. Dembski*，《没有免费午餐：为何特定复杂性无法在没有智能的情况下获得》，*Rowman & Littlefield*，（*2001*年），*https://doi.org/10.5040/ 9798216410973*。[21] *A.Z. Deriba*、*D.Y. Yang*，使用过渡马尔可夫链蒙特卡罗方法对大型交通网络进行基于性能的风险评估，*ASCE-ASME*工程系统风险与不确定性杂志*A*辑：土木工程，*11(1)*，*0402-4090*，（*2025*年），*https://doi.org/10.1061/AJRUA6. RUENG-1402*。[22] *P. DiBarba*等，使用深度神经网络和高斯过程回归对双频功率电感器进行最优多物理场综合，《算法》*18(1)*，*10*，（*2025*年），*https://doi.org/10. 3390/a18010010*。[23] *X. Dong*、*Yu. Qin-mei*和*C. Dong*，基于“中国制造*2025*”探讨中国工程教育改革，（*2025*年），*https://doi.org/10.12783/ dtssehs/ermm2017/14672*。[24] *B. Doerr*、*Y. Ghannane*和*M.I. Brahim*，迈向更强的基于排列的进化算法理论，载于《遗传与进化计算会议论文集》，第*1390—1398*页，（*2022*），*https://doi.org/10. 1145/3512290.3528720*。[25] *B. Efron*、*T. Hastie*，《计算机时代的统计推断：算法、证据与数据科学》，剑桥大学出版社，（*2016*年），*http://www.cambridge.org/ 9781107149892*。[26] *S. Fortini*、*S. Petrone*，贝叶斯统计学中的可交换性、预测与预测建模，《统计科学》，*40(1)*，*40—67*，（*2025*年），*https://doi.org/ 10.1214/24-STS965*。[27] *L.F. Plata-Gonzĺez*、*I. Amaya*、*J.C. Ortiz-Bayliss*、*S.E. Conant-Pablos*、*H. Terashima-Marín*和*C.A. Coello Coello*，面向背包问题的合成实例的基于进化的定制，《软计算》，*23(23)*，*12711—12728*，（*2019*年），*https://doi.org/10.1007/s00500-019-03822-w*。[28] *T.Kee*、*K.H. Winky*，面向城市科学的机器学习模型优化：超参数调优方法的比较分析，《城市科学》，*9(9)*，*348*，（*2025*年），*https://doi.org/10.3390/urbansci9090348*。[29] *A. Gelman*、*J.B. Carlin*、*H.S. Stern*和*D.B. Rubin*，《贝叶斯数据分析》，*Chapman and Hall/CRC*，（*1995*年），*https://doi.org/10.1201/9780429258411*。[30] *D.E. Goldberg*，《搜索、优化与机器学习中的遗传算法》，*Addision Weseley Publisching Company*，马萨诸塞州雷丁，（*1989*年）。[31] *M. Goldblum*等，《无免费午餐定理、柯尔莫哥洛夫复杂度与归纳偏置在机器学习中的作用》，（*2024*年），*https://arxiv.org/abs/ 2304.05366*。[32] *T. Hastie*、*R. Tibshirani*和*J. Friedman*，《统计学习要素：数据挖掘、推断与预测》，（第*2*卷，第*1—758*页），纽约：施普林格，（*2009*年），*https://doi.org/10.1007/978-0-387-21606-5*。

[33] N. Hansen、A. Auger、D. Brockhoff和T. Tušar，《黑箱优化基准测试中的随时性能评估》，IEEE Transactions on Evolutionary Compu- tation，26(6)，1293—1305，（2022年），https://doi.org/10.1109/TEVC.2022.3210897。[34] L. He等，《基于反向传播神经网络和深度强化学习的直冷式动力电池热管理性能优化》，《应用热工程》258，124661，（2025年），https://doi.org/10.1016/j.applthermaleng.2024.124661。[35] S. Herbert，《无免费午餐定理：量子态测量作为经典采样和生成式建模中的资源》，（2023年），https://doi.org/10.48550/arXiv.2309.13967。[36] Y. Ho、D. Pepyne，《无免费午餐定理及其含义的简单解释》，《优化理论与应用杂志》115，549—570，（2002年），https://doi.org/10.1023/A:1021251113462。[37] J. Huang、R. Wu和J. Li，《利用结构先验和进化技术增强网络鲁棒性》，《信息科学》690，121529，（2025年），https://doi. org/10.1016/j.ins.2024.121529。[38] C. Igel，《无免费午餐定理：元启发式算法的局限性与展望》，载于：《元启发式算法设计的理论与原理方法》，柏林、海德堡：Springer Berlin Heidelberg，第1—23页，（2014年），https://doi.org/10.1007/ 978-3-642-33206-7_1。[39] C. Igel、M. Toussaint，《适用无免费午餐结果的函数类别》，（2001年），https://doi.org/10.48550/arXiv.cs/0108011。[40] C. Igel、M. Toussaint，《关于优化中的无免费午餐定理的最新结果》，（2003年），https://doi.org/10.48550/arXiv.cs/0303032。[41] J. Jiang，《线性与广义线性混合模型及其应用》，施普林格，纽约州纽约，（2007年），https://doi.org/10.1007/978-0-387-47946-0_2。[42] I. Jolliffe，《主成分分析》，载于：《国际统计科学百科全书》，柏林、海德堡：Springer Berlin Heidelberg，第1945—1948页，（2025年），https://doi.org/10.1007/978-3-662-69359-9_483。[43] T. Joyce、J.M. Herrmann，《无免费午餐定理及其对元启发式优化影响的综述》，《仿生算法与应用优化》，施普林格国际出版，第27—51页，（2018年），https://doi.org/10.1007/978-3-319-67669-2_2。[44] R. Koenker，《分位数回归》，《计量经济学会专著》，剑桥大学出版社，剑桥，（2025年）。[45] S.O. Kimbrough、G.J. Koehler、M. Lu和D.H. Wood，《用于约束优化的可行—不可行双种群（*FI-2Pop*）遗传算法：距离追踪与无免费午餐》，《欧洲运筹学杂志》190(2)，310—327，（2008年），https://doi.org/10.1016/j.ejor.2007.06.028。

[46] A. Kohli 等，《基于基因组生物标志物发现的精准医学》，载于：通过药物基因组学开展基因组学驱动的药物发现，IGI Global Scientific Publishing，第185–216页，（2025年），https://doi.org/10.4018/ 979-8-3693-6597-7.ch007.[47] R.S. Liggett，《二次指派问题：应用与求解策略分析》，《环境与规划B：规划与设计》，7(2)，141–162，（1980），https://doi.org/10.1068/b070141.[48] T. Lim，《夏普利公理的霍奇理论扩展》，（2021），https://doi.org/ 10.48550/arXiv.2106.15094.[49] L. Mariot、S. Picek、D. Jakobovic、M. Djurasevic 和 A. Leporati，《置换码进化的难度》，载于：进化计算应用国际会议（EvoStar的一部分）。施普林格国际出版：施普林格，第141–156页，（2022），https://doi.org/10.1007/978-3-031-02462-7_10.[50] J. McDermott，《元启发式研究人员何时以及为何可以忽略“无免费午餐”定理》，《SN计算机科学》，1(1)，第60页，（2020），https://doi.org/ 10.1007/s42979-020-0063-3.[51] A. Moraglio、R. Poli，《用于排列表示的几何交叉》，Intelligenza Artificiale，5(1)，49–63，（2011），https://doi.org/10.3233/IA-2011-0004.[52] G.D. Monek、S. Fischer，《专家孪生体：集成基于模糊逻辑的决策模块的数字孪生体》，Decision Making: Applications in Management and Engineering，8(1)，1–21，（2025），https://doi.org/10.31181/dmame8120251181.[53] D.C. Montgomery，《实验设计与分析》，约翰·威立父子出版公司，（2017）。[54] M.H. Nikzad、M. Heidari-Rarani 和 M. Mirkhalaf，《一种基于田口法的神经网络架构优化新方法：在弹性短纤维复合材料中的应用》，《复合材料科学与技术》259：110951，（2025），https: // doi.org/10.1016/j.compscitech.2024.110951.[55] M. Oltean，《寻找无免费午餐定理的实践证据》，载于《生物启发式高级信息技术国际研讨会论文集》（第472–483页）。柏林，海德堡：Springer Berlin Heidelberg，（2004），https://doi.org/10.1007/978-3-540-27835-1_34.[56] R. Ortner，《元归纳自适应算法》，《普通科学哲学杂志》54(3)，433–450，（2023），https://doi.org/10.1007/ s10838-021-09590-2.[57] V. Parada，《算法自动生成》，CRC出版社，（2025），https://doi. org/10.1201/9781003542896.[58] S.S. Pathak 等，《使用多层感知网络、广义前馈网络、主成分分析网络、时滞循环网络和循环网络预测混凝土抗压强度》，《亚洲土木工程杂志》26(1)，431–450，（2025），https://doi.org/10.1007/ s42107-024-01175-w.[59] F. Pesarin、L. Salmaso，《复杂数据的置换检验：理论、应用与软件》，约翰·威立父子出版公司，（2010）。

[60] O. Polushkina-Merchanskaya 等，《在具有多变量响应的方差分析置换检验中处理缺失数据、离群值和变换的考量》，《化学计量学与智能实验室系统》，105320，（2025年），https://doi. org/10.1016/j.chemolab.2025.105320.[61] E. Raponi、N.C. Rakotonirina、J. Rapin、C. Doerr 和 O. Teytaud，《低预算优化：黑箱优化基准测试套件与OpenAI Gym的比较》，《IEEE进化计算汇刊》，第29卷(1)，第91–101页，（2025年），https://doi.org/10.1109/TEVC.2023.3346788.[62] M. Rashki、M. Faes，《可靠性分析的无免费午餐定理》，ASCE-ASME工程系统风险与不确定性杂志A辑：土木工程，9(3)，04023019，（2023），https://doi.org/10.1061/AJRUA6.RUENG-1015.[63] A. Rijal、B. Waluyo，《数学差异化教学的有效性：来自小学生的见解》，教育与学习杂志（EduLearn）19(1)，241–248，（2025年），https://doi.org/10.11591/edulearn.v19i1. 21806.[64] G.O. Roberts、S.K. Sahu，《吉布斯采样器的更新方案、相关结构、分块与参数化》，英国皇家统计学会杂志B辑：统计方法学59(2)，291–317，（1997），https://doi.org/10.1111/1467-9868.00070.[65] B. Rombaut 等，《天下没有免费的午餐：探索Greenkeeper依赖项机器人在npm中引入的开销所得的经验教训》，《ACM软件工程与方法学汇刊》32(1)，1–40，（2023），https: //doi.org/10.1145/3522587.[66] E.C. Rosenthal、C. Trudeau，《单向网络中的相互依赖安全博弈》，《风险分析》44(6)，1430–1439，（2024），https://doi.org/10.1111/risa. 14234.[67] A. Said，《算术编码导论：理论与实践》，（2023），https:// doi.org/10.48550/arXiv.2302.00819.[68] V. Santucci、J. Ceberio，《双随机矩阵模型在二次指派问题中的应用》，Evolutionary Computation，33(3)，425–457，（2025），https://doi.org/10.1162/evco_a_00369.[69] V. Santucci、M. Baioletti 和 A. Milani，《组合优化中群体智能与进化算法的代数框架》，Swarm and Evolutionary Computation 55. 100673，（2020），https://doi.org/10.1016/j.swevo.2020.100673.[70] V. Santucci、M. Baioletti 和 A. Milani，《用于线性排序问题的代数差分进化》，载于《2015年遗传与进化计算年会伴随出版物》论文集，第1479–1480页，（2015），https://doi.org/10.1145/2739482.2764693.[71] A. Schmitz、C. Schmidt-Wellenburg 和 J. Volle，《通过迭代互惠映射整合几何数据分析与网络分析：以德国社会学领域为例》，Poetics 108，101-947，（2025），https://doi.org/10.1016/j.poetic.2024.101947。

[72] C. Schumacher, M. D. Vose 和 L. D. Whitley，无免费午餐与问题描述长度，载于：第 3 届遗传与进化计算年度会议论文集，第 565-570 页，Morgan Kaufmann 出版公司，伯灵顿，（2001年）。[73] G. Sun, Tt. Li，林业供应链中的物流优化：基于碳减排的车辆路径问题，Forests 16(1)，62，（2025年），https: //doi.org/10.3390/f16010062。[74] G. Schurz，休谟问题已解决：元归纳的最优性，MIT 出版社，（2019），https://doi.org/10.7551/mitpress/11964.001.0001。[75] N. Simon 等，通过坐标下降法求 *Cox* 比例风险模型的正则化路径，统计软件杂志，39，1-13，（2011），https://doi.org/ 10.18637/jss.v039.i05。[76] S.M. Soak, B.H. Ahn，用于旅行商问题的新型基于子回路的交叉算子，载于：遗传与进化计算会议，柏林，海德堡：Springer Berlin Heidelberg，第 1610-1611 页，（2003），https://doi.org/10.1007/ 3-540-45110-2_49。[77] M.J. Streeter，两大类不适用无免费午餐结果的函数，载于：遗传与进化计算会议。柏林，海德堡：Springer Berlin Heidelberg，第 1418-1430 页，（2003），https://doi.org/10.1007/ 3-540-45110-2_15。[78] G. Sugihara, R.M. May，非线性预测：区分时间序列中的混沌与测量误差的一种方法，Nature，344(6268)，734-741，（1990），https: //doi.org/10.1038/344734a0。[79] Y. Sun, G.G. Yen 和 Z. Yi，基于 *IGD* 指标的多目标优化问题进化算法，IEEE进化计算汇刊，23(2)，第 173-187 页，（2019），https://doi.org/10.1109/TEVC.2018.2791283。[80] S. Surjanovic, D. Bingham，仿真实验虚拟库：测试函数和数据集，（2013），http://www.sfu.ca/{~}ssurjano/optimization.html。[81] Y. Takane, E. J. Beh 和 R. Lombardo，修正 *Freeman-Tukey* 统计量的对比理论及其在列联表 *Tukey* 事后检验中的应用，计算统计学，40(3)，1423-1446，（2025年），https://doi.org/10.1007/ s00180-024-01537-7。[82] A. Thomaser, J. De Nobel, D. Vermetten, F. Ye, T. Bäck 和 A. Kononova，何时离散化：分析算法在离散化连续问题上的性能，载于遗传与进化计算会议论文集，第 856-863 页，（2023），https://doi.org/10.1145/3583131.3590410。[83]A. Urbańczyk 等，$(\mu+\lambda)$ 采用社会认知变异的进化策略，《自动化、移动机器人与智能系统杂志》，18(1)，第 1-11 页，（2024年），https://doi.org/10.14313/JAMRIS/1âĂŘ2024/1。[84] S. Watanabe，代数几何与贝叶斯统计学的最新进展，《信息几何》，7（增刊 1），187-209，（2024年），https://doi.org/10.1007/ s41884-022-00083-9。

[85] D. Whitley, J. Rowe，聚焦型无免费午餐定理。载于第 10 届遗传与进化计算年度会议论文集，第 811-818 页，（2008），https://doi.org/10.1145/1389095.1389254。[86] D. Whitley，遗传算法教程，《统计与计算》，4(2)，65-85，（1994），https://doi.org/10.1007/BF00175354。[87] D. Whitley, J.P. Watson，复杂性理论与无免费午餐定理，《搜索方法：优化与决策支持技术入门教程》，317-339，施普林格，波士顿，马萨诸塞州，（2005），https://doi.org/10.1007/ 0-387-28356-0_11。[88] D. H. Wolpert, W.G. Macready，搜索无免费午餐定理。第 10 卷，第 12 期，第 2756-2760 页，技术报告 SFI-TR-95-02-010，圣菲研究所，（1995年），https://api.semanticscholar.org/CorpusID:12890367。[89] D.H. Wolpert, W.G. Macready，优化无免费午餐定理，IEEE进化计算汇刊，1(1)，67-82，（1997），https://doi.org/ 10.1109/4235.585893。[90] D.H. Wolpert，无免费午餐定理的重要意义是什么？，载于：黑箱优化、机器学习与无免费午餐定理。施普林格国际出版：施普林格国际出版，第 170 卷，373-388，（2021），https://doi.org/10.1007/ 978-3-030-66515-9_13。[91] D.H. Wolpert，无免费午餐定理对元归纳的影响，《普通科学哲学杂志》54(3)，421-432，Sringer，（2023），https: // doi.org/10.1007/s10838-022-09609-2。[92] M. Vladimirova 等，在单元级理解贝叶斯神经网络中的先验。载于：国际机器学习会议，第 6458-6467 页，PMLR，（2019），https://proceedings.mlr.press/v97/vladimirova19a.html。[93] J. Voges 等，*GABAC*：一种用于基因组数据的算术编码方案，《生物信息学》，36(7) 2275-2277，（2020），https://doi.org/10.1093/bioinformatics/ btz922。[94] M.D. Vose, D. Whitley，没有免费的午餐。载于第 9 届遗传与进化计算会议伴随论文集，第 3734-3764 页，（2007），https://dl.acm.org/doi/10.1145/1274000.1274131。[95] H. Xie, Hu. Kang，求解鲁棒部分二次特征值配置问题的舒尔方法，《计算与应用数学》，44(1)，41，（2025年），https:// doi.org/10.1007/s40314-024-03000-1。[96] X. Yang 等，进化算法的元黑箱优化：综述与展望，《群体与进化计算》，93，101838，（2025年），https: //doi.org/10.1016/j.swevo.2024.101838。[97] H. Zhang, W. Shiji 和 W. Jieping，采用改进遗传算法自动优化设计激光三角测距传感器，《测量》，241，115739，（2025年），https://doi.org/10.1016/j.measurement.2024.115739。