

# 0/1 背包问题多算法性能对比与场景精准适配研究

毛子璠<sup>1)\*</sup> 刘伦<sup>1)\*</sup> 杨昊哲<sup>1)\*</sup> 张涛<sup>1)\*</sup> 梁天宇<sup>1)\*</sup> 邓康<sup>1)\*</sup>

1) 天津大学福州国际联合学院, 天津 300072

\*: 共同贡献

**摘要** 在统一实验协议下, 本文围绕 0/1 背包问题, 对 12 种代表性算法 (精确算法、近似/随机化算法与元启发式算法) 开展系统比较。实验覆盖 8 类实例集 (每类 20 个独立样本, 每个样本重复运行 5 次), 以运行时间、解质量 Gap 及其方差为核心指标进行评测。结果表明: Greedy+Max 在常规实例上表现最为均衡, 平均 Gap 为 0.058; 有界集束搜索在陷阱型对抗场景中显著提升解质量, 将平均 Gap 由 0.205 降至 0.004; 增强分支限界在小规模实例上获得最优解, 并体现出更高的剪枝效率。基于上述发现, 本文进一步提出“规模—结构—预算”驱动算法选型建议: 当  $N > 1000$  时, 优先采用 Greedy+Max; 当  $500 \leq N \leq 1000$  时, 推荐有界集束搜索与记忆模拟退火的组合; 当  $N \leq 50$  时, 建议采用增强分支限界。研究揭示了实例结构特征在算法适配性中的主导作用, 以及复杂度约束对应用边界的决定性影响。然而, 本文仍受实例规模覆盖不足与真实场景数据验证有限的约束; 后续可面向更大规模实例与真实业务数据进一步扩展与检验。报告代码可访问于:

<https://github.com/haozheyang-tju/2526-01-0-1-Knapsack-Problem-Algorithm-Comparison>.

关键词: 0/1 背包问题; 算法对比; 性能分析; 场景适配; 复杂度; 解质量

## 1 问题定义与算法选型

### 1.1 问题定义

本章以 0/1 背包问题 (0/1 Knapsack Problem) 这一经典 NP-hard 问题为对象, 围绕“不同算法在何种条件下表现更优”这一核心研究议题, 提出全组对比研究的总体设计与范围界定。0/1 背包问题旨在容量约束下选择若干物品以最大化总价值, 既是组合优化与近似算法研究中的基准问题, 也可抽象为广告预算分配、推荐位填充、物流配送及云资源调度等实际应用场景。现实应用往往要求在极短时间内获得高质量可行解, 而实例在规模、分布特征与约束紧度方面差异显著。因此, 有必要从计算速度、解质量与算法稳定性三个维度开展系统化、可复现的比较分析。

在问题定义与算法选型方面, 本文首先对研究对象进行数学化描述: 每个物品具有正的重量与价值, 背包容量为整数; 决策变量为二元选择; 目标是在总重量不超过容量的约束下最大化总价值。算法选型遵循课程“至少涵盖三类性质不同算法”的要求, 整体研究覆盖以下范畴:

- 1) 经典精确算法 (如基于容量维度的动态规划、分支定界法), 用于获取精确最优解或提供强基线;
- 2) 近似算法与随机化算法 (如本章研究的两种贪心策略和模拟退火算法), 适用于大规模或实时场景下获取具有理论保证或经验表现稳定的高质量解;
- 3) 元启发式算法 (遗传算法、粒子群优化等), 用于处理非凸、多峰等复杂结构下的全局优化问题。

### 1.2 算法选型与动机

#### 1.2.1 经典常规算法

##### 1.2.1.1 集束搜索

本小节对经典集束搜索与有界集束搜索进行定义、机制对比与实验表现分析。经典集束搜索 (在代码中对应 algo\_beam\_search) 是一种受限的广度优先搜索策略<sup>[1]</sup>。核心动机: 纯粹的贪心算法在每一步仅保留一个最优解, 容易陷入局部极值; 而穷举搜索的计算成本随规模指数增长。Classic Beam Search 通过引

入参数  $B$  (束宽, Beam Width), 在每一轮扩展中保留前  $B$  个最有潜力的部分解, 试图以  $O(B)$  的线性额外开销换取对解空间的更充分探索。基准定位: 在本章实验中, 它作为衡量“简单启发式改进”有效性的标尺。我们特别关注其在 GreedyTrap (贪心陷阱) 类实例上的表现——实验数据 (metrics\_summary.csv) 显示, 该算法在陷阱实例上的 Mean Gap 高达 0.205, 这验证了仅依赖当前局部增益进行筛选的经典策略在对抗性结构下的局限性。有界集束搜索 (Bounded\_BeamSearch) 有界集束搜索 (在代码中对应 algo\_beam\_search\_optimized) 是对经典策略的增强变体。核心动机: 针对经典集束搜索易被“高初始增益但低长远价值”的诱饵元素误导的问题, 该算法引入了基于子模性上界 (Upper Bound) 的剪枝与筛选机制<sup>[2][4]</sup>。它不仅仅依据当前的目标函数值  $f(S)$  进行排序, 而是结合了对剩余容量潜在增益的估计 (bounding)<sup>[3]</sup>, 从而在波束筛选阶段保留那些当前值稍低但具备更高增长潜力的路径。这种“引导式”搜索在近年来的子模最大化研究中被证明能显著提升鲁棒性<sup>[5][6]</sup>。改进验证: 该算法是我们重点考察的“鲁棒性基准”。实验结果表明, 在同样的 GreedyTrap 实例上, 其 Mean Gap 骤降至 0.004, 几乎达到了最优解。这一巨大的性能飞跃是以牺牲一定的计算时间为代价的 (运行时间从 0.005s 增至 0.017s), 这为后续分析“精度-效率权衡”提供了绝佳的样本。

#### 1.2.1.2 分支限界算法

本小节的核心内容是系统对比基础分支限界法和增强分支限界法两种精确求解策略。基础分支限界法 (在代码中对应 bb\_basic) 采用深度优先搜索结合线性松弛上界的经典框架, 其核心动机是在保证找到最优解的前提下, 通过上界剪枝避免穷举搜索的指数级开销。该算法作为本实验的基准方法, 体现了分支限界法的基本思想: 系统探索解空间的同时, 利用问题结构信息提前排除不可能达到最优解的分支。实验数据显示, 该算法在所有实例类型上均能保证 100% 的精确率, 验证了其作为精确算法的可靠性, 但在运行时间和节点探索效率方面存在优化空间。

增强分支限界法 (在代码中对应 bb\_enhanced) 是对基础算法的系统性改进, 其核心创新在于引入了 Forget & Parragh (2024) 提出的探测技术和最佳上界优先策略。该方法的动机源于经典分支限界法在节点选择策略上的局限性——深度优先搜索可能陷入局部搜索区域, 而简单的宽度优先则内存开销巨大。增强算法通过探测技术提前评估子问题的潜在价值, 结合最佳上界优先策略动态调整搜索方向, 旨在以智能的节点选择加速收敛过程。特别值得注意的是, 该算法在 ProbingBenefit、Uncorrelated 和 WeaklyCorr 三类实例中表现出显著的节点探索减少 (分别减少 13.9%、11.5% 和 30.5%), 这验证了探测技术在识别无效分支方面的有效性, 体现了现代分支限界法研究的前沿思想。

在实验设计方面, 实验采用系统化设计确保可复现性和统计可靠性。覆盖 6 种实例类型和 4 种规模层次, 生成 480 个测试用例, 每个用例运行两种算法共 960 次实验。标准化实验环境消除平台差异, 多维评估指标包括运行时间、节点效率、解质量、收敛速度和内存消耗。引入收敛速度指数量化收敛效率, 采用配对 t-test 进行统计显著性检验, 为性能差异提供统计证据。实验框架的严谨性为结论可靠性奠定基础。

在性能评估与可视化方面, 实验数据揭示增强算法平均运行时间为基础算法的 5.4 倍, 差异高度显著。节点探索效率呈复杂图景: 在 ProbingBenefit 等三类实例中减少 13.9%-30.5%, 验证探测技术有效性; 但在其他实例中未能体现优势。两种算法均实现 100% 精确率, 增强算法收敛速度较慢, 内存消耗高出 33 倍。可视化体系包括 15 个图表, 全面展现性能趋势、实例对比、统计显著性和预测模型。在总结与展望方面, 研究发现增强算法的局部改进被实现开销抵消, 总体性能未达预期。基础算法在多数场景下更可靠, 增强算法适用于特定实例类型和资源充足环境。研究局限包括合成数据为主、规模有限。未来方向包括开发自适应探测机制、轻量级实现和真实数据验证。本研究建立了完整评估框架, 为分支限界法的优化发展提供实证基础和方向指引<sup>[7-10]</sup>。

#### 1.2.2 近似/随机化算法

##### 1.2.2.1 贪心算法

本小节独立设计了专用的实例库、评估指标、统计方法与可视化输出规范。本小节重点研究贪心算法家族中的两种代表性方法——按价值密度排序的 GreedyDensity 算法, 以及在其基础上引入“最佳单件兜底”机制的 Greedy+Max 算法——并将其作为本小节的对比基线, 与其他同学负责的随机化算法与元启发

式章节形成对照与互补。

本小节的核心定位是为“贪心近似算法”提供可靠的基准参照。GreedyDensity 采用“按密度排序并依次装入”的简洁策略，时间复杂度为 $O(n\log n)$ 。但在存在“主导单件”的问题结构中存在系统性缺陷。Greedy+Max 在 GreedyDensity 的基础上，以 $O(n)$ 的额外开销扫描所有可装入的单件，并输出两者中的较优解，从而获得严格的 $1/2$  近似保证（该性质为 BYOG+Max 框架在线性目标与单背包约束下的特例），整体复杂度仍保持为 $O(n\log n)$ ，空间复杂度为 $O(1)$ 。为实现理论与实验的闭环验证，本小节同时引入两种对照基线：在中小规模问题上使用基于容量维度的动态规划获取精确最优解；在更大规模问题上则采用分数背包松弛（LP Relaxation）提供统一上界，用于评估近似解的质量。

在理论分析方面，本小节系统推导两种贪心算法的时间与空间复杂度，并阐明其理论保证与失效场景。GreedyDensity 的时间复杂度主要由排序步骤决定，为 $O(n\log n)$ ，空间复杂度约为 $O(1)$ ，但对 0/1 背包问题不具备常数近似比；当实例中存在密度略低但价值极高、重量接近容量的“关键单件”时，该算法可能系统性地错过最优解。Greedy+Max 的时间复杂度同样为 $O(n\log n)$ ，空间复杂度为 $O(1)$ ，通过比较贪心解 $S_g$ 与最佳单件 $j^*$ ，输出 $\max v(S_g), v(j^*)$ ，可获得  $1/2$  的近似保证。其关键理论直觉在于：线性松弛的最优值由前缀整装部分与临界物品的一部分共同构成，而後者的价值不超过某个可装入单件的价值，因此有： $OPT \leq UB \leq v(S_g) + v(j^*)$ 。

在基线算法方面，容量维度动态规划的时间复杂度为 $O(nC)$ ，空间复杂度为 $O(C)$ ，在问题规模与容量适中时可作为评估近似算法 gap 的“金标准”；而在更大规模问题上，我们转而使用线性松弛上界进行评估。此外，我们引入容量比 $\alpha = C / \sum w_i$ 作为关键结构参数：当 $\alpha$ 较小或中等偏紧时，“是否捕获关键单件”对结果影响显著，Greedy+Max 的兜底机制往往表现更优；当 $\alpha$ 较大趋于宽松时，GreedyDensity 的前缀装填行为逐渐逼近分数背包的最优装填，两者差距趋于收敛。

在实验设计方面，本小节构建了一套可复现的实例库与统计分析流程。我们构建了八类代表性实例族，涵盖“独立均匀分布”“弱/强相关性”“反相关性”“重尾分布”“密度陷阱结构”“可调对抗实例”及大规模基准场景。其中，GreedyTrap-ADV 类别通过参数 $\delta$ 控制“大件相对优势”，形成连续可调的对抗强度。每个类别生成 20 个样本，各进行 5 次随机重复，以保证统计功效。所有实例生成过程均采用固定随机种子与稳定哈希策略，确保跨算法与跨重复实验的一致性。同时，通过设定阈值（如 $n \leq 200, C \leq 6000$ ）控制动态规划的启用，以避免过高的伪多项式计算开销。对于大规模实例，我们统一采用线性松弛上界进行质量评估，在保持度量一致性的同时控制计算成本。

在性能评估与可视化分析方面，我们从时间、质量与稳定性三个维度进行综合评价：计算时间采用壁钟时间，汇报均值、方差及 95% 置信区间；解质量以 $\text{gap}_{\text{OPT}}$ 或 $\text{gap}_{\text{UB}}$ 统一度量，确保在不同规模间具有可比性；稳定性通过样本间与重复实验间的方差及置信区间进行刻画。可选辅助指标包括基于 tracemalloc 的峰值内存分析、以运行时间为基础的能耗代理估计，以及通过拟合模型 $\text{time} \approx a \cdot n \log_2 n + b$ 与可决系数 $R^2$ 评估算法可扩展性。在可视化方面，我们将输出按类别划分的柱状图、容量比扫描得到的“优势区间图”（平局按 0.5/0.5 分摊），并提供“优势差热图”以突出算法表现的过渡区域；同时包括微型多规模实验中 $n \log n$ 拟合散点图、配对检验（如 t 检验与 Wilcoxon 检验）对应的显著性条形图与置信区间图。在挑战性分析中，我们还将提供基于经验模型的外推预测，分析 $R^2$ 与偏差来源（如解释器开销、缓存效应、调度噪声、并列密度项的稳定性，以及 LP 上界对 OPT 的近似偏差等）。

在总结与展望部分，本小节将结合理论与实验结果，形成可指导实践的研究结论与部署建议：在容量偏紧或存在“关键单件”的结构中（如重尾分布、密度陷阱类），Greedy+Max 的优势稳定，与其  $1/2$  近似保证的理论直觉一致；而在强相关或容量宽松的场景下，两者性能趋近，表明密度贪心策略已足够接近可分装行为。在工程实践中，我们推荐将 Greedy+Max 作为“几乎不增加复杂度而更具健壮性”的默认基线；在极端追求速度且数据近似可分的场景下，可退而使用 GreedyDensity。进一步地，可结合“优势区间图”与实例结构特征（如容量比、密度分布的偏度与峰度、相关系数等）进行策略选择。展望未来，本小节所建立的贪心算法基线可为其他算法范式的独立研究提供参照。不同研究者可基于各自设计的实例库与指标体系进行性能对比，进而通过整合多方结果，绘制跨范式的性能边界，绘制跨范式的性能边界表，深入探索

FPTAS、随机局部搜索与元启发式在“紧容量一对抗结构”中的收益与代价，并评估混合式轻量改进策略（如“候选前缀 + 小规模动态规划/中间相遇搜索”）在真实业务数据中的推广潜力<sup>[11-15]</sup>。

#### 1.2.2.2 模拟退火算法

0/1 背包问题的 NP-hard 特性决定了在多项式时间内找到精确解的困难性。虽然动态规划算法在小规模问题上表现优异，但当问题规模增大时，其时间复杂度  $O(nC)$  变得难以接受。分支定界算法虽然能保证找到最优解，但搜索空间随物品数量指数增长，同样不适用于大规模实例。贪心算法虽然速度快，但解的质量有限，且容易被特殊构造的实例所误导。这些局限性促使我们探索元启发式算法在这一问题上的应用潜力。

模拟退火算法，通过模拟物理退火过程来实现全局优化。它通过控制温度参数来平衡探索与利用，以一定概率接受劣质解来避免陷入局部最优。记忆模拟退火算法则在标准模拟退火的基础上引入了种群思想、局部搜索和遗传操作，形成了更为复杂的搜索策略。这两种算法代表了两种不同的设计：单一解搜索与群体搜索。通过对比这两种算法在 0/1 背包问题上的表现，我们不仅可以评估它们各自的优劣，还能深入理解不同搜索策略对问题求解的影响。

算法实现与核心定位，本研究实现了两种算法的优化版本：快速模拟退火算法和快速记忆模拟退火算法。这两种实现都针对运行效率进行了特别优化，包括自适应参数调整、简化邻域操作、限制迭代次数等策略。标准模拟退火算法被定位为平衡型求解器，它适合中等规模问题，在计算资源有限且需要快速获得满意解的场景下表现出色。该算法采用单一解搜索策略，内存效率高，实现相对简单，参数调节直观。记忆模拟退火算法则被定位为高质量求解器，它适合大规模复杂问题，在解质量优先且允许适度增加计算时间的场景下具有优势。该算法采用种群搜索策略，通过维持一定规模的解群体来保持多样性，结合局部搜索操作强化开采能力，并引入遗传操作促进解之间的信息交换。这种多层次的设计使得算法能够在全局探索和局部开采之间取得更好的平衡。

实验设计框架，为确保实验结果的科学性和可靠性，本研究设计了全面的实验框架。实例集合涵盖 10 种不同类型的背包问题，包括非相关、弱相关、强相关、反相关、重尾分布、贪心陷阱等多种类型，问题规模从 60 个物品到 1000 个物品不等。这种多样化的实例设计能够全面评估算法在不同问题特性下的表现。实验采用固定随机种子（42）和稳定哈希生成技术，确保实验结果的完全可复现。每个算法配置进行 5 次重复实验，每次重复包含 20 个样本实例，总测试实例数达到 1000 个，保证了统计显著性。评估指标体系包括核心性能指标如解质量（Gap 值）和运行时间，以及深入分析指标如稳定性、可扩展性、健壮性和时间-质量权衡分析。参数设置采用自适应策略，根据问题规模动态调整算法参数。对于小规模问题（ $n \leq 50$ ），设置较高的初始温度和较慢的冷却速率，允许充分探索；对于中等规模问题（ $50 < n \leq 100$ ），采用适中参数；对于大规模问题（ $n > 100$ ），则优先考虑计算效率，采用较低的初始温度和较快的冷却速率。这种自适应设计使算法能够根据不同问题规模自动调整搜索策略。

在性能评估与可视化分析方面，实验结果通过多维度的统计分析揭示了两种算法的性能特点。整体性能对比显示，记忆模拟退火算法在大多数实例类型上获得更小的 Gap 值，表明其解质量普遍优于标准模拟退火算法。这一优势在复杂实例和大规模问题上尤为明显。然而，这种质量优势是以增加运行时间为代价的，记忆模拟退火算法的平均运行时间通常是标准模拟退火算法的 1.5 到 3 倍。

按实例类型分析揭示了算法性能与问题特征之间的关联。在非相关和弱相关实例上，两种算法的性能差距相对较小；而在强相关、反相关和贪心陷阱等复杂实例上，记忆模拟退火算法的优势更加显著。这表明记忆模拟退火算法对问题结构的适应性更强，能够更好地处理具有挑战性的优化问题。

规模可扩展性分析显示，随着问题规模的增大，两种算法的 Gap 值都有所增加，但记忆模拟退火算法的增加幅度较小。在物品数量超过 200 的大规模问题上，记忆模拟退火算法表现出明显的优势。运行时间方面，两种算法的时间增长趋势基本符合理论分析，记忆模拟退火算法由于额外的计算开销，时间增长更快。

可视化分析通过多种图表直观展示了实验结果。箱线图揭示了算法性能的分布特征，显示记忆模拟退火算法的 Gap 值分布更加集中，表明其性能更加稳定。热力图按实例类型展示了两种算法的性能对比，清

晰显示了记忆模拟退火算法在大多数区域的优越性。趋势线图展示了算法性能随问题规模的变化规律，为算法选择提供了直观参考。时间-质量权衡散点图则揭示了两种算法在求解效率与解质量之间的不同平衡点。

综合实验结果，本研究得出以下主要结论：记忆模拟退火算法在解质量方面普遍优于标准模拟退火算法，特别是在复杂实例和大规模问题上优势明显；标准模拟退火算法在运行效率方面具有优势，适合对计算时间敏感的应用场景；算法的性能优势与问题特征密切相关，不同实例类型可能适合不同的算法。

从理论贡献来看，本研究建立了系统化的元启发式算法对比框架，提出了多维度、多层次的评估指标体系，为后续研究提供了方法论参考。实践意义方面，研究结果为工程应用中的算法选择提供了具体指导，基于问题规模的自适应参数设置方法可直接应用于实际系统。研究的局限性包括算法对比范围相对有限，仅包括两种模拟退火变体；参数调优可能不够充分，采用固定自适应规则而非系统优化；实例集合虽具有多样性，但可能未能覆盖所有实际应用场景。这些局限性也为未来研究指明了方向。

在未来展望方面，未来研究可以从多个方向拓展本工作。算法改进方面，可以探索模拟退火算法与其他优化技术的融合，如与精确方法结合形成混合算法，或与深度学习技术结合提升搜索效率。实验扩展方面，可以测试更大规模的问题实例，对比更多元启发式算法，研究动态变化的环境中的算法性能。应用拓展方面，可以将研究方法扩展到多目标背包问题、带约束的变体问题以及实际问题如物流配送、资源分配等。0/1 背包问题作为组合优化领域的经典问题，其求解方法的探索始终具有重要的理论价值和实践意义。本研究通过系统的实验设计和深入的分析，揭示了模拟退火算法与记忆模拟退火算法在求解这一问题上的性能差异与适用场景。随着计算能力的不断提升和算法技术的持续发展，元启发式方法在实际优化问题中的应用前景将更加广阔。本研究的框架、方法和发现可为相关领域的研究者和实践者提供有价值的参考，推动组合优化算法在实际问题中的更有效应用。

### 1.2.3 元启发算法

#### 1.2.3.1 遗传算法

遗传算法是一种受自然界生物进化机制启发而设计的优化方法，其核心思想是通过模拟“适者生存”的演化过程，在解空间中搜索高质量的近似最优解<sup>[17]</sup>。在求解 0-1 背包问题时，该算法将每个可能的物品选择方案编码为一个二进制串，其中每一位表示对应物品是否被选入背包——1 代表选择，0 代表不选。这种编码方式天然契合背包问题的离散决策特性，使得遗传操作能够直接作用于问题结构本身。

在算法启动之初，系统会尝试生成一个初始种群。与完全随机生成不同，本文使用的标准遗传算法采用了一种偏向有效解的策略：只保留那些总重量不超过背包容量的个体作为初始成员。若难以找到足够多的有效解，则用全零解（即不选任何物品）进行填充，以确保种群规模满足要求。这一设计有助于提升算法初期的搜索效率，避免大量无效个体拖慢进化进程。

适应度函数是引导进化方向的关键。在此实现中，适应度直接取为所选物品的总价值，但前提是总重量未超过背包容量；一旦超重，适应度被强制设为零，意味着该个体在后续选择中几乎不可能被保留。这种硬性惩罚机制虽然简单，却能有效排除不可行解，使搜索集中于可行域内。然而，它也可能导致部分接近可行边界的优质解被过早淘汰，这是该策略的一个潜在局限。

进化过程由选择、交叉和变异三个主要操作驱动。选择阶段采用排序选择法：所有个体按适应度从高到低排列，仅保留前一半进入下一代。这种方式隐含了精英保留机制，确保当前最优解不会在演化中丢失。交叉操作则模拟基因重组，代码支持三种模式：单点交叉在随机位置切断两个父代染色体并交换前段；两点交叉交换中间片段；均匀交叉则对每一位独立决定是否交换。这些策略在保持父代优良基因的同时，促进新解的产生。值得注意的是，交叉后的新个体与原种群合并，使种群规模暂时扩大，为后续选择提供更多候选。

变异操作引入随机扰动以维持种群多样性，防止算法陷入局部最优。代码中仅对新生成的一半个体施加变异，且变异发生的概率高达 50%，体现出较强的探索倾向。位翻转变异随机改变某一位的取值，而交换变异则在两个位置间互换基因，后者在某些情况下有助于保持解的结构平衡。为应对大基因组可能导致的整数溢出问题，编码后的整数值被限制在 32 位有符号整数范围内，体现了工程实现中的稳健性考虑。

整个算法通过多轮独立运行增强鲁棒性。每一轮从有效初始种群开始，反复执行选择、交叉和变异，直至种群缩减至单一“冠军”个体。最终，从所有轮次的冠军中挑选适应度最高者作为全局输出。这种多起点策略有效降低了单次运行因随机性导致的性能波动，提高了结果的可靠性。

本文同时还对标准的遗传算法进行改进，在其基础上引入多层次并行搜索与重启机制<sup>[16]</sup>的一种增强型方法，其核心目标是提升在求解 0-1 背包问题时的全局搜索能力、收敛速度以及对复杂约束条件的适应性。相较于前文所述的标准遗传算法，该方法在种群初始化、演化结构、探索策略和计算资源利用等方面进行了系统性优化。

首先，在种群构建方面，标准遗传算法通常依赖于单一初始种群，并在整个演化过程中仅在此局部区域内进行搜索，容易陷入局部最优。而改进的遗传算法采用“分而治之”的思想，将整个解空间（即所有可能的基因型，范围从 0 到  $2^n-1$ ）划分为多个互不重叠的子区间，每个子区间由一个独立的多岛遗传算法实例负责搜索。这种划分使得不同子种群在各自局部区域内并行探索，有效增强了算法对全局解空间的覆盖能力，避免了传统单一种群因初始位置不佳而导致的早熟收敛问题。

其次，在演化机制上，该方法融合了多岛遗传算法与多线程并行技术。每个子遗传算法（即“岛屿”）内部不仅执行常规的选择、交叉和变异操作，还进一步在其每一代演化中启动多个并行的交叉-变异子任务。这种双重并行结构——既在岛屿间并行，又在岛屿内并行——显著提升了计算效率。同时，每个岛屿可独立演化出局部最优解，这些解在每轮循环结束时被汇集形成“优质种群”，作为下一层级 RestartBaseGeneticAlgorithm 的初始输入，从而实现了从局部最优向全局最优的逐层提炼。

第三，该方法引入了智能重启机制。在顶层的 RestartBaseGeneticAlgorithm 中，算法会动态监测当前种群是否满足预设的阈值条件（例如总价值不低于某目标值，或总重量不超过容量）。若当前唯一幸存个体不满足该条件，或随机数超过高阈值（如 0.995），系统将自动注入新一轮的有效初始种群（包含空解、贪心解和随机解），重新激活多样性。这一机制有效防止了算法在后期因种群过度同质化而停滞不前，确保即使在接近收敛阶段仍保有跳出局部最优的能力。

此外，在初始种群生成策略上，RestartBaseGeneticAlgorithm 也进行了优化。它不再仅依赖纯随机采样，而是结合了空解、基于价值密度的贪心构造解以及随机解，形成一个高质量、多样化的起点。这种混合初始化方式显著提高了初始种群的平均适应度，为后续演化奠定了良好基础。

### 1.2.3.2 二进制粒子群算法（BPSO）和二进制布谷鸟搜索算法（BCS）

粒子群优化算法模拟鸟群觅食行为，通过个体与群体经验指导搜索过程。在二进制版本中，粒子位置  $x_i^t = \{x_{i1}^t, x_{i2}^t, \dots, x_{in}^t\}$  由 0/1 向量构成，表示候选解，其中  $x_{ij}^t = 1$  表示在迭代  $t$  时物品  $j$  被选中。BPSO 的核心更新方程为：

$$v_{ij}^{t+1} = w \cdot v_{ij}^t + c_1 r_1 (pbest_{ij}^t - x_{ij}^t) + c_2 r_2 (gbest_j^t - x_{ij}^t) \quad (1)$$

$$x_{ij}^{t+1} = \begin{cases} 1 & \text{if } \sigma(v_{ij}^{t+1}) > r_3 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

其中  $\sigma(v) = \frac{1}{1+e^{-v}}$  为 Sigmoid 传递函数， $w$  为惯性权重， $c_1$  和  $c_2$  为学习因子， $r_1, r_2, r_3$  为  $[0,1]$  均匀分布随机数， $pbest_j^t$  和  $gbest_j^t$  分别表示个体和全局历史最佳位置。为增强算法性能，我们引入自适应惯性权重机制：

$$w(t) = w_{\max} - \frac{(w_{\max} - w_{\min}) \cdot t}{T_{\max}} \quad (3)$$

其中  $w_{\max} = 0.9$ ,  $w_{\min} = 0.4$ ,  $t$  为当前迭代次数， $T_{\max}$  为最大迭代次数。该策略在早期强调全局探索，后期注重局部开发。对于不可行解，我们设计基于价值密度的修复机制：当  $\sum_{i=1}^n w_i x_i > C$  时，按密度  $\frac{v_i}{w_i}$  从低到高依次移除物品，直至满足约束。该算法的总体时间复杂度为  $O(T \cdot N \cdot n)$ ，空间复杂度为  $O(N \cdot n)$ ，其中  $T$  为迭代次数， $N$  为粒子数量。

布谷鸟搜索算法模拟布谷鸟的繁殖行为，其核心特点为莱维飞行（Lévy flight）分布。在二进制版本

中，鸟巢位置同样表示为 0/1 向量。BCS 的搜索过程由两部分组成：全局探索和局部开发。全局探索采用莱维飞行更新：

$$x_i^{t+1} = x_i^t + \alpha \oplus \text{Lévy}(\lambda) \quad (4)$$

其中 $\alpha$ 为步长缩放因子， $\oplus$ 表示逐点乘，莱维分布 $\text{Lévy}(\lambda)$ 的概率密度函数为：

$$L(s, \lambda) = \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \cdot \frac{1}{s^{1+\lambda}}, (s \gg 0) \quad (5)$$

局部开发则通过随机选择两个鸟巢进行差异操作：

$$x_{\text{new}} = x_a + r \times (x_a - x_b) \quad (6)$$

其中 $r$ 为  $[0,1]$ 随机向量， $x_a$ 和 $x_b$ 为随机选择的两个鸟巢。对于二进制表示，我们采用概率转换机制：

$$P(x_{ij}^{t+1} = 1) = \sigma(v_{ij}^{t+1}) \quad (7)$$

其中 $v_{ij}^{t+1}$ 由莱维飞行或局部开发生成。BCS 还包含鸟巢替换机制：每次迭代有一定概率 $p_a$ 丢弃最差的鸟巢，由新随机生成的鸟巢替代。其时间复杂度为 $O(T \cdot M \cdot n)$ ，空间复杂度为 $O(M \cdot n)$ ，其中 $M$ 为鸟巢数量。

## 2 理论分析

### 2.1 经典算法

#### 2.1.1 集束搜索

本节对上述两种集束搜索算法的时间复杂度、空间复杂度及理论性能边界进行形式化推导，并结合子模性特征分析其瓶颈。

##### 1、经典集束搜索 (Classic\_BeamSearch):

算法逻辑：设全集为  $V$ ，约束基数为  $k$ ，束宽为  $B$ 。算法从空集开始，迭代  $k$  轮。在第  $i$  轮，对上一轮保留的  $B$  个集合  $\{S^{(1)}_{i-1}, \dots, S^{(B)}_{i-1}\}$ ，分别尝试添加  $V \setminus S$  中的元素。时间复杂度：每一轮需评估  $B \cdot (n - i)$  个候选元素。总时间复杂度为  $O(k \cdot n \cdot B)$ 。

由于  $B$  通常设为常数（本实验中  $B$  固定），其渐进复杂度与贪心算法同阶，均为  $O(nk)$ 。实验中 **Uncorrelated-Large** ( $n=500$ ) 上的运行时间约为 0.019s，符合线性增长预期。性能保证与瓶颈：经典集束搜索无严格的近似比保证。其瓶颈在于“短视筛选”：仅根据  $f(S \cup \{e\}) - f(S)$  选出的 Top- $B$  候选者，可能全部落入设计好的局部最优陷阱中（如 **GreedyTrap** 所示），导致最终解与 OPT 的差距无法控制。

##### 2、有界集束搜索 (Bounded\_BeamSearch)

算法逻辑：在经典扩展的基础上，该算法利用子模函数的边际收益递减性质（Diminishing Returns）构建上界估计。对于候选路径  $S$ ，计算其潜在最大价值

$$UB(S) = f(S) + \sum_{j \in \text{Top}(k-|S|)} \Delta(j | S) \quad (8)$$

筛选标准由单纯的  $f(S)$  变为  $f(S)$  与  $UB(S)$  的加权或基于  $UB(S)$  的分支定界策略。时间复杂度：理论最坏复杂度可能略高于经典版，取决于上界计算的开销。若采用 **Lazy Evaluation** 优化，复杂度仍维持在  $O(k \cdot n \cdot B)$  量级。实测分析：在 **GreedyTrap** 实例上，**Bounded** 版耗时(0.0177s)约为 **Classic** 版(0.0054s)的 3.3 倍。这额外的开销主要源于更复杂的候选评估逻辑和对无效路径的剪枝判断。空间复杂度：为了维护多样性队列或上界信息，实际内存消耗略高。实验数据显示其平均内存占用为 10.52KB，高于 **Classic** 版的 6.75KB。适用边界：该算法在处理强相关（**StronglyCorr**）和对抗性（**GreedyTrap**）数据时表现出极强的理论优势。由于其能“看”到潜在上界，因此能有效跳出局部最优。但在数据分布均匀（**Uncorrelated**）的简单场景下，其解质量与经典版持平（Gap 均为~0.056），此时额外的计算开销使其性价比略有下降。

#### 2.1.2 分支限界算法

本节分析基础分支限界法（**bb\_basic**）和增强分支限界法（**bb\_enhanced**）的时间复杂度、空间复杂度、

性能保证以及瓶颈。

### 1、基础分支限界法 (bb\_basic)

采用深度优先搜索 (DFS) 策略，使用栈作为核心数据结构。算法首先将物品按价值密度 ( $v/w$ ) 降序排序以获得更紧的上界。从根节点开始，每次扩展时考虑是否包含当前物品，使用分数背包松弛计算上界进行剪枝。当节点的上界不大于当前最优解时，完全放弃该分支的搜索。

### 2、增强分支限界法 (bb\_enhanced)

引入两项关键技术：1) 探测技术 (Probing) 在分支前评估子问题潜力，提前剪除无望分支；2) 最佳上界优先节点选择策略，始终扩展上界最高的节点。使用优先队列实现节点选择，确保始终沿着最有希望的方向搜索。

在时间复杂度和空间复杂度方面，基础分支限界法最坏情况下需要探索所有可能的子集，时间复杂度为  $O(2^n)$ ，实际运行时间受剪枝效率影响极大。通过分数背包上界剪枝，期望时间复杂度可降低到  $O(b^n)$ ，其中  $b$  为分支因子。实验中在  $n = 30$  的 Uncorrelated 实例上，平均运行时间为 0.003 秒，符合指数时间算法的预期，每节点处理时间约为  $5.2 \times 10^{-6}$  秒，显示每个节点的计算开销较低，深度优先搜索仅需存储当前路径，空间复杂度为  $O(n)$ ，实验中测得平均内存消耗为 0.003MB，与理论预期一致，内存使用与问题规模  $n$  呈线性关系。增强分支限界法最坏时间复杂度为  $O(2^n)$ ，但实际搜索空间显著减少，探测技术增加每个节点的计算开销，但减少了生成的节点总数。在 ProbingBenefit 实例上，增强版探索节点数比基础版减少 13.9%，总运行时间增加，表明探测开销主导了时间消耗，每节点处理时间增加至  $2.47 \times 10^{-5}$  秒。优先队列需要存储所有活跃节点，最坏情况空间复杂度为  $O(2^n)$ ，实际内存消耗显著增加，内存消耗与生成节点数正相关，增强版生成节点数减少但每个节点存储更多信息。

在性能保证与瓶颈方面，基础分支限界法作为精确算法，保证找到最优解剪枝效率依赖上界质量，分数背包松弛提供了较紧的上界，但并非总是紧的。深度优先搜索可能陷入不利分支，导致延迟发现高质量解，分数背包上界对于某些实例类型（如 GreedyTrap）可能不够紧，实验中平均需探索超过 50% 的可行节点才能首次找到最优解。增强分支限界法确保找到最优解，最佳上界优先策略理论上最小化找到最优解前探索的节点数，探测技术提供更精细的可行性判断，减少无效探索，但内存开销增加，优先队列维护所有活跃节点，成为限制，每个节点的额外评估增加了单节点处理时间，需要频繁重新计算上界以维护队列顺序导致时间成本增加。

## 2.2 近似/随机化算法

### 2.2.1 贪心算法分析

本章构建与实验相呼应的理论框架，涵盖形式化建模、贪心算法的近似保证与复杂度分析、评估基线及上界选取、以及算法的失效模式与适用边界，为后续对比研究提供理论支撑。

我们首先建立形式化描述与符号系统：设共有  $n$  件物品，第  $i$  件物品的重量与价值分别为  $(w_i, v_i) \in \mathbb{Z}_{>0}^2$ ，背包容量为  $C \in \mathbb{Z}_{>0}$ 。标准的 0/1 背包问题旨在最大化总价值  $\sum_i v_i x_i$ ，约束条件为  $\sum_i w_i x_i \leq C$  且  $x_i \in \{0, 1\}$ 。若将  $x_i$  的取值松弛为  $[0, 1]$ ，则得到分数背包问题，其为原问题的线性规划松弛形式，可在按价值密度  $\rho_i := v_i/w_i$  降序排序后通过一次扫描求解，时间复杂度主要由排序步骤决定，为  $O(n \log n)$ 。该松弛解天然构成原问题最优值 OPT 的一个上界 UB。

在贪心算法家族中，本章重点比较两种具有代表性的简洁策略：

- **GreedyDensity**: 按价值密度  $\rho_i$  降序排列物品，依次装入可容纳的物品，形成解集  $S_g$ 。
- **Greedy+Max**: 在 GreedyDensity 所得解  $S_g$  与所有可装入单件中价值最高的物品  $j^*$  之间，选取价值较大者作为最终解。

两种算法均以排序步骤为主导，时间复杂度为  $O(n \log n)$ ，空间复杂度为  $O(1)$ 。Greedy+Max 的关键改进在于对“关键单件”的系统性兜底机制，从而获得严格的  $1/2$  近似保证。该保证源于分数背包最优解的结构特性：其由若干按密度整装的前缀物品与临界物品的一部分构成。前缀部分的价值恰好等于  $v(S_g)$ ，而临界物品的增量价值不超过任一可装入单件的价值  $v(j^*)$ ，因此有：

$$UB \leq v(S_g) + v(j^*) \quad (9)$$

进而可得：

$$\max \{v(S_g), v(j^*)\} \geq \frac{1}{2} \text{OPT} \quad (10)$$

相比之下，仅依赖密度前缀策略的 GreedyDensity 在 0/1 背包设定下不具备常数近似比保证。当存在“密度略低但价值极高、重量接近容量”的关键单件时，该策略易受多个“密度略高、但任意两件即超容量”的小件诱导，仅能获取其中之一，从而在紧容量条件下系统性地偏离最优解。后文将通过对抗实例族 GreedyTrap 与参数可调的 GreedyTrap-ADV 对此失效模式进行参数化建模，并在实验中验证其敏感区间。

为实现实验评估的闭环，除贪心算法外，我们引入以下两类评估基线：

- **容量维度动态规划：**设  $dp[c]$  表示容量  $c$  下能获得的最大价值，采用自顶向下或自底向上方式进行状态转移，时间复杂度为  $O(nC)$ ，空间复杂度为  $O(C)$ 。在  $n$  与  $C$  适中时，该方法可求得精确最优值 OPT，为显著性检验与差距度量提供“金标准”。
- **分数背包上界：**在问题规模或容量较大时，采用分数背包松弛上界 UB 替代 OPT 以统一评估差距，确保不同规模实例间的可比性，同时保持与 Greedy+Max 理论证明一致的上界结构。

两类基线的引入也有助于解释后续“理论—实验一致性”分析中可能出现的微小偏差：动态规划仅在规模可控时启用，而 LP 上界在少数样本上可能导致差距值的轻微波动，但不影响算法间的相对排序结论。

在复杂度方面，两种贪心算法均以排序为主导步骤，时间复杂度为  $O(n \log n)$ ，辅以一次线性扫描，空间复杂度为  $O(1)$ 。Greedy+Max 仅较 GreedyDensity 多一次寻找最佳单件的线性扫描，不改变整体复杂度量级。动态规划方法的时间复杂度  $O(nC)$  在容量较大时显著增加，因此我们在实现中通过阈值控制其调用条件（如  $n \leq 200, C \leq 6000$ ），超出阈值时统一采用 LP 上界进行评估。

在理论性质方面，Greedy+Max 的  $1/2$  近似比为紧界限（存在构造反例逼近该比值），这同时解释了其在“容量偏紧且关键单件显著”的场景下相对于 GreedyDensity 的稳定优势。当容量趋于宽松、数据呈强相关或近似等密度分布时，GreedyDensity 的前缀装填行为逐渐逼近分数背包解，两者在理论与实验表现上均趋于一致。

为在实验中揭示并量化上述性能边界，我们引入容量比  $\alpha = C / \sum_i w_i$  作为关键结构参数，通过对  $\alpha$  从紧到松进行扫描，绘制“优势区间图”，并采用“平局按 0.5/0.5 分摊”的胜率定义及“优势差热图”以凸显算法表现的过渡区域。对抗实例族 GreedyTrap-ADV 中的参数  $\delta$  提供了从温和到强烈的可控难度梯度，使显著性与效应量随  $\delta$  单调增加的理论预期在实验中得以可复现地验证。

综上所述，本章的理论分析为后续的可扩展性回归（如  $\text{time} \approx a \cdot n \log_2 n + b$  模型拟合）、统计显著性检验（如 t 检验与 Wilcoxon 检验）及置信区间分析提供了严谨的解释框架。

### 2.2.2 模拟退火分析

从理论层面分析，模拟退火算法的收敛性基于马尔可夫链模型。在降温速度足够慢的理想条件下，算法以概率 1 收敛到全局最优解。然而在实际应用中，由于计算时间有限，通常采用较快的降温速率，这导致算法只能找到近似最优解。

#### 1、基本模拟退火算法

核心思想：模拟固体退火过程，通过控制温度参数，在搜索过程中以一定概率接受劣质解，从而跳出局部最优。

关键组件：温度调度： $T_{k+1} = \alpha T_k$ ， $\alpha \in (0,1)$ ，Metropolis 准则： $P = \exp(-\Delta E/T)$ ，邻域搜索：翻转、交换、添加移除操作，约束处理：修复策略和惩罚函数。

模拟退火算法源于统计物理学中固体退火过程的模拟。其核心思想是将组合优化问题中的“目标函数值”映射为物理系统中的“能量（Energy）”，将“解的状态”映射为微观粒子的“排列状态”。

SA 与普通爬山法（Hill Climbing）的本质区别在于它具备概率性跳出局部最优的能力。算法构造了一个非时齐的马尔可夫链，其状态转移概率严格遵循 Metropolis 准则：

$$P(x_{new}|x_{current}) = \begin{cases} 1 & \text{if } \Delta E < 0 \text{ (更优解, 直接接受)} \\ \exp\left(-\frac{\Delta E}{k_B T}\right) & \text{if } \Delta E \geq 0 \text{ (劣解, 概率接受)} \end{cases} \quad (11)$$

其中 $\Delta E$ : 新旧解的目标函数差值 (对于最大化问题, 取负值)。 $T$  (温度): 控制参数。高温阶段 ( $T \rightarrow \infty$ ):  $\exp(-\Delta E/T) \approx 1$ , 算法近似于随机游走, 具有极强的全局探索能力, 能穿越能量景观中的高壁垒。

低温阶段 ( $T \rightarrow 0$ ):  $\exp(-\Delta E/T) \approx 0$ , 算法近似于贪婪搜索 (Greedy Search), 专注于局部开发 (Exploitation), 在当前吸引盆 (Basin of Attraction) 内收敛。

## 2、记忆模拟退火算法

记忆模拟退火算法则融合了计算智能中的模因算法理论, 通过种群进化与局部搜索的双层优化机制, 在保持多样性的同时加强局部搜索能力。模因算法 (Memetic Algorithm, MA) 是进化算法 (Evolutionary Algorithm) 的一个变种。如果说遗传算法 (GA) 模拟的是基于基因的生物进化, 那么模因算法模拟的则是基于模因的进化。

时间复杂度分析显示, 标准模拟退火算法的时间复杂度为 $O(k \times n)$ , 其中  $k$  为迭代次数,  $n$  为物品数量。记忆模拟退火算法由于涉及种群操作和局部搜索, 时间复杂度为 $O(m \times k \times L \times n)$ , 其中  $m$  为种群大小,  $L$  为局部搜索迭代次数。虽然理论上记忆模拟退火算法的时间复杂度更高, 但通过更高效的搜索策略, 它可能以更少的迭代次数达到相同或更好的解质量。探索与利用的平衡机制是两种算法设计的核心差异。

记忆模拟退火算法是基于精英保留策略的改进模拟退火算法, 其核心思想来源于多篇优化算法改进研究, 特别是在复杂组合优化问题中的应用。引入了全局记忆机制, 始终记录并维护整个搜索过程中遇到的最优解。这一机制有效解决了普通模拟退火算法因接受劣质解而可能遗失优质解的问题。

核心机理: 拉马克式进化 (Lamarckian Evolution) 与传统达尔文进化论不同, 记忆模拟退火算法通常采用拉马克进化模型: 个体在生命周期内通过“学习” (即局部搜索) 获得的性状改变, 可以直接编码回基因型并遗传给后代。在算法中架构由两层循环嵌套而成: 全局层 (Global Search) —— 种群动力学: 利用交叉算子 (Crossover) 在解空间中进行大跨度的跳跃, 通过组合不同父代的优良基因片段, 探索未知的解空间区域。这防止了算法过早收敛于单一的局部极值。局部层 (Local Search) —— 个体强化: 每个个体在被评估前, 都会经历一个简化的模拟退火过程。数学表达: 设进化算子为  $\mathcal{E}$ , 局部搜索算子为  $\mathcal{L}_{SA}$ , 则下一代种群  $P_{t+1}$  可表示为:

$$P_{t+1} = \mathcal{L}_{SA}(\mathcal{E}(P_t)) \quad (12)$$

这里的  $\mathcal{L}_{SA}$  充当了“学习”过程, 它将粗糙的后代解迅速“退火”至其所在吸引盆的底部

关键改进点:

(1) 全局最优解追踪: 记忆机制核心在每次接受新解时更新全局最优。

(2) 解质量保证: 算法最终返回的是整个搜索过程中的全局最优解, 而非最终状态的解, 确保了解的质量。

(3) 搜索过程保护: 即使算法在高温阶段接受了劣质解进行探索, 也不会丢失之前找到的优质解。

理论优势: 避免优质解遗失问题, 提高算法的收敛可靠性, 增强在复杂实例上的鲁棒性和保持相同的时间复杂度

标准模拟退火主要通过温度控制来实现这一平衡: 高温阶段偏向全局探索, 接受劣质解的概率较高; 低温阶段偏向局部利用, 主要接受改进解。记忆模拟退火则通过多种机制共同作用: 种群多样性支持全局探索, 局部搜索操作强化局部利用, 交叉操作促进信息交换, 变异操作保持种群多样性。这种多机制协同的设计使得记忆模拟退火在复杂问题上可能表现出更强的鲁棒性。相比于单一的模拟退火算法, 记忆模拟退火利用种群的多样性避免了陷入深层局部最优; 极大提高了局部收敛的精度和速度。

## 2.3 元启发算法

### 2.3.1 遗传算法分析

作为一类启发式随机搜索方法, 遗传算法通常不提供严格的最坏情况时间或空间复杂度上界, 也不具备确定性的近似比或全局收敛保证。然而, 基于其实现逻辑和典型运行行为, 仍可对其计算开销、资源消

耗以及性能瓶颈进行合理推导。

标准遗传算法的时间开销主要由种群规模 $P$ 、物品数量 $n$ 、演化代数 $G$ 以及外层循环次数 $T$ 共同决定。每一代中，对每个个体计算适应度需 $O(n)$ 时间（通过点积计算总重量与总价值），选择操作涉及排序，复杂度为 $O(P\log P)$ ，而交叉与变异操作对 $O(P)$ 个个体各处理 $O(n)$ 位。因此单代时间复杂度约为 $O(Pn + P\log P)$ ，若每轮外循环执行 $G$ 代，则总时间复杂度为 $O(TGPn)$ （忽略低阶项）。空间方面，算法需存储种群及其临时副本，每个个体以整数形式编码，故空间复杂度为 $O(P)$ 。尽管代码中采用精英选择策略（保留前 50% 个体）可确保历史最优解不被丢弃，从而形成非递减的最优适应度序列，但该方法仍无法保证在有限时间内收敛到全局最优，也不存在已知的近似比理论支撑。

改进的遗传算法在结构上更为复杂，包含三层机制：底层将整个解空间划分为 $m$ 个子区间，每个区间由一个岛屿遗传算法独立搜索；中层在每个岛屿内部并行执行多个交叉-变异任务；顶层则利用各岛屿输出的局部最优解构成高质量初始种群，并在其上演化，辅以条件触发的重启机制。从计算总量看，底层 $m$ 个岛屿各自处理约 $P/m$ 规模的种群，总计算量仍为 $O(G \cdot Pn)$ ，与标准算法同阶；顶层因种群极小（最多  $m$  个个体），开销可忽略。因此，改进算法的总时间复杂度仍为 $O(TGPn)$ ，但由于引入了多层级并行，在支持真正并行计算的环境下可显著缩短实际运行时间。空间复杂度同样保持在 $O(P)$ 量级。尽管其通过解空间分区增强了全局探索能力，通过贪心初始化提升起点质量，通过重启机制缓解早熟收敛，但这些改进仍属于经验性优化，未带来形式化的性能保证——它依然无法证明能以特定概率找到近似最优解，也无法给出收敛速度的数学界。

两类算法在设计中存在若干关键瓶颈。首先是适应度评估的重复开销：每次个体评估均需遍历全部 $n$ 个物品计算重量与价值，当 $n$ 很大时成为主要计算负担。其次是编码方式的局限性：使用单个整数表示基因组在 $n > 63$ 时必然发生截断，导致高维问题中解信息丢失，破坏搜索完整性。第三，内层演化循环以“种群缩减至 1”为终止条件，这不符合标准遗传算法惯例，易导致多样性过早枯竭，且实际演化代数不可控，影响结果稳定性。最后，重启机制的触发条件过于宽松（如重启率设为 0.995 意味着极低频次重启），难以有效应对种群同质化问题。

### 2.3.2 二进制粒子群算法和二进制布谷鸟搜索算法分析

BPSO 的收敛性可通过马尔可夫链理论分析。设 $P_t$ 为第 $t$ 代粒子群状态分布， $T$ 为状态转移矩阵，则：

$$L(s, \lambda) = \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \cdot \frac{1}{s^{1+\lambda}}, (s \gg 0) \quad (13)$$

$$P_{t+1} = P_t T \quad (14)$$

当 $t \rightarrow \infty$ 时，若满足遍历性条件， $P_t$ 将收敛到稳态分布。自适应惯性权重策略确保算法在后期具有强局部搜索能力，有助于收敛到局部最优解。

BCS 的收敛性则基于莱维飞行的重尾分布特性。设 $f(x)$ 为适应度函数， $L(\lambda)$ 为莱维分布，当 $\lambda \in (0, 2]$  时，搜索步长具有无限方差，确保算法在理论上可以访问解空间中任何点。BCS 全局收敛的概率为：

$$P_{\text{converge}} = \lim_{t \rightarrow \infty} P(f(x_t) = f(x^*)) = 1 \quad (15)$$

其中 $x^*$ 为全局最优解。这一性质保证了 BCS 在无限时间内找到全局最优解的概率为 1。

BPSO 对参数  $\{w, c_1, c_2, N, T\}$  高度敏感。通过响应面分析，我们建立适应度与参数间的关系模型：

$$F = \beta_0 + \sum_{i=1}^k \beta_i p_i + \sum_{i=1}^k \beta_{ii} p_i^2 + \sum_{i \neq j} \beta_{ij} p_i p_j + \epsilon \quad (16)$$

其中 $p_i$ 为参数， $\beta$ 为系数， $\epsilon$ 为误差项。敏感性分析显示，惯性权重 $w$ 的初始值对收敛速度影响最大，而粒子数量 $N$ 与问题维度 $n$ 应满足 $N \geq 0.1n$ 以保证充分探索。

BCS 对参数 $\{\alpha, \lambda, p_a, M, T\}$ 敏感。莱维分布参数 $\lambda$ 决定搜索步长分布，理论分析表明当 $\lambda \in [1.5, 2.0]$  时算法性能最佳。鸟巢数量 $M$ 与问题维度的关系为 $M \geq \log_2(n)$ ，以确保解空间充分覆盖。

## 2.4 理论对比小结

在完成对 12 种算法求解 01 背包问题的系统理论分析后，本文绘制了如表 1 所示的表格，并对算法的性能特点进行综合性总结。

表 1 12 种算法求解 01 背包问题的时间和空间复杂度

算法名称	时间复杂度	空间复杂度
经典集束搜索	$O(Bkn)$	$O(Bn)$
有界集束搜索	$O(Bkn)$	$O(Bn)$
基础分支限界法	$O(2^n)$	$O(2^n)$
增强分支限界法	$O(2^n)$	$O(2^n)$
GreedyDensity 算法	$O(n \log n)$	$O(1)$
Greedy+Max 算法	$O(n \log n)$	$O(1)$
基本模拟退火算法	$O(kn)$	$O(n)$
记忆模拟退火算法	$O(mkLn)$	$O(n)$
标准遗传算法	$O(TGPn)$	$O(P)$
改进遗传算法	$O(TGPn)$	$O(P)$
二进制粒子群算法	$O(TNn)$	$O(Nn)$
二进制布谷鸟搜索算法	$O(TMn)$	$O(Mn)$

从时间效率来看，各类算法呈现出明显的层次划分。贪心策略的两种算法——GreedyDensity 和 Greedy+Max，凭借其  $O(n \log n)$  的优越时间复杂度，在计算速度上表现最为出色，这主要得益于其核心的排序操作和一次遍历的简单结构。以模拟退火、遗传算法、粒子群和布谷鸟搜索为代表的智能优化算法，它们的时间复杂度表现为  $O(kn)$ 、 $O(TGPn)$ 、 $O(TNn)$  等形式，其共同特点是运行时间与算法参数（如迭代次数  $T$ 、种群规模  $P$ 、邻域搜索次数  $k$  等）和问题规模  $n$  呈多项式关系，这在求解精度和计算耗时之间进行灵活权衡提供了可能。相比之下，分支限界法（包括基础和增强版本）在最坏情况下的时间复杂度高达  $O(2^n)$ ，呈现指数级增长，这体现了精确算法在追求最优解时所需付出的巨大计算代价。集束搜索及其有界变体的时间复杂度  $O(Bkn)$  则介于两者之间，其性能受束宽  $B$  的直接影响，展现了启发式搜索在控制搜索宽度方面的设计思路。

在空间消耗方面，不同算法的需求差异显著。两种贪心算法仅需常数级别的额外存储空间  $O(1)$ ，内存效率极高。模拟退火算法和遗传算法分别需要  $O(n)$  和  $O(P)$  的线性空间，其开销主要来自对单个解状态或种群中个体信息的维护。集束搜索类算法的空间复杂度  $O(Bn)$  与其束宽  $B$  成正比，体现了搜索广度对内存的占用。而二进制粒子群算法和二进制布谷鸟搜索算法的空间复杂度  $O(Nn)$  和  $O(Mn)$ ，则揭示了基于种群迭代的算法需要为每个个体存储一个完整的  $n$  维解向量所导致的内存开销。分支限界法的空间复杂度在最坏情况下同样达到  $O(2^n)$ ，这与其需要维护一个可能包含指数级数量节点的优先队列或栈的搜索过程密切相关。

总体而言，没有一种算法能在时间效率和空间消耗上同时达到最优。贪心算法速度最快、内存最省，但牺牲了解的最优性保证；分支限界法能求得精确最优解，但面临时空的双重指数爆炸风险；各类智能优化算法和启发式搜索算法则在两者之间提供了丰富的选择，通过调整束宽、迭代次数、种群规模等参数，可以适应不同的问题规模、精度要求和资源约束。本次理论分析为后续的实验验证与算法选型建立了清晰的标准和预期，在实际应用中，应根据具体问题的规模、实时性要求、内存限制以及对解的质量期望，来选择合适的求解策略。

## 3 实验设计

本章旨在对 12 种算法进行系统性的实验评估。为实现严谨的算法对比，本文设计了统一的实例生成框架、标准化的评测流程与可复现的随机性控制机制。实验中所构建的实例库在保留经典价值一重量分布

模式（包括不相关、弱相关、强相关、反相关、重尾、陷阱与严格对抗等类型）的基础上，引入“近容量对抗模板”以增强对算法差异的识别能力。该模板首先生成 $n - 1$ 个近容量小件（满足 $w_i > 0.5C$ ，确保任意两件无法同时装入），再追加一件大件（常规类别中 $w_{big} \approx 0.95C$ ，严格对抗类别中 $w_{big} = C$ ）。大件的密度被设置为略低于小件中的最佳密度，从而诱导 GreedyDensity 在首件选择上出现系统性偏差，而 Greedy+Max 则可通过其“最佳单件兜底”机制有效纠正此类错误。该设计能在各实例族上产生清晰且稳定的解质量差异，为后续的统计显著性检验与优势区间分析提供充分的证据支持。除非特别说明，本章实验中容量标尺固定为 $C = 1000$ ；容量比扫描分析将在第五章中独立进行。

### 3.1 数据集构建

我们采用统一的实例生成模板（适用于所有实例族）：给定容量标尺 $C = 1000$ ，首先生成 $n - 1$ 个小件，其重量区间为 $w_i \in [0.55C, 0.75C]$ （严格对抗类别的设定有所不同）；随后根据各实例族的特性生成对应的物品价值 $v_i$ ；最后追加一件大件。大件的生成分为两种模式：

1、 $w_{big} \approx 0.95C$ ，其密度 $d_{big} = d_{best} \times (1 - \varepsilon)$ ，其中 $d = \frac{v}{w}$ ， $d_{best}$ 为小件中的最大密度， $\varepsilon \in [0.01, 0.02]$

根据不同实例族进行微调。

2、严格对抗模板（GreedyTrap-ADV）： $w_{big} = C$ ，其密度 $d_{big} = 0.95 \times d_{small} + \delta$ ，其中 $d_{small}$ 为小件的标称密度， $\delta > 0$ 为对抗强度参数。

各实例族的具体配置如下（所有族均忽略 cap\_ratio 参数，统一使用固定容量 $C = 1000$ ，保留该参数仅为接口兼容）：

（1）Uncorrelated（独立随机）：小件重量 $w_i \in [0.55C, 0.75C]$ ，价值 $v_i \sim \text{Unif}[1, 1000]$ 独立生成；大件按常规模板设置， $\varepsilon = 0.01$ ；默认 $n = 60$ 。

（2）WeaklyCorr（弱相关）：小件价值 $v_i = \max\{1, w_i + \text{noise}\}$ ，其中 $\text{noise} \sim \text{Unif}[-20, 20]$ ；大件 $\varepsilon = 0.01$ ；默认 $n = 150$ 。

（3）StronglyCorr（强相关）：小件价值 $v_i = w_i + \Delta$ ， $\Delta = 40$ ；大件 $\varepsilon = 0.01$ ；默认 $n = 150$ 。

（4）InverseCorr（反相关）：令 $W_{max} = \max_i w_i$ ，小件价值 $v_i = \max\{1, W_{max} - w_i + \xi\}$ ，其中 $\xi \sim \text{Unif}[0, 20]$ ；大件 $\varepsilon = 0.01$ ；默认 $n = 150$ 。

（5）HeavyTailed（重尾分布）：小件重量通过 $\text{Pareto}(\alpha = 2.0)$ 分布采样后缩放并截断至 $[0.55C, 0.75C]$ 区间，价值 $v_i \sim \text{Unif}[200, 1200]$ ；大件 $\varepsilon = 0.015$ ；默认 $n = 150$ 。

（6）GreedyTrap（经典陷阱）：小件价值 $v_i \approx 1.02 \times w_i + \text{微扰}$ （仍处于近容量区间）；大件采用强化对抗设置， $\varepsilon = 0.02$ ；默认 $n = 150$ 。

（7）GreedyTrap-ADV（严格对抗， $\delta$ 可调）：小件重量 $w_i \in [0.90C, 0.98C]$ ，价值 $v_i \approx 1.01 \times w_i + \text{微扰}$ ；大件设置为 $w_{big} = C$ ， $d_{big} = 0.95 \times d_{small} + \delta$ ；默认 $n = 13$ （12 个小件+1 个大件），便于通过动态规划计算精确最优解； $\delta$ 值越大，算法间差异越显著。

（8）Uncorrelated-Large（大规模基准）：与 Uncorrelated 族同分布，但 $n = 500$ ；为测试可扩展性，强制禁用动态规划，解质量通过分数背包（LP）上界进行评估。

### 3.2 采样策略与可复现性控制

我们采用“每类 20 个独立样本 $\times$ 每个样本 5 次重复”的采样方案。两种算法在同一组实例上进行评估，形成配对观测，便于进行均值比较、方差分析、置信区间估计以及配对显著性检验。

为确保实验的完全可复现性，我们建立了三层控制机制：

（1）库级播种：在程序入口处调用 random.seed(42)与 np.random.seed(42)，锁定伪随机数生成序列；

（2）稳定子种子生成：针对每个("class", "rep", "sample")三元组，使用 stable\_hash\_str（基于 zlib.adler32）计算 32 位稳定哈希值，先对 $10^7$ 取模，再与 seed\_base 相加生成子种子（代码实现： $\text{seed} = \text{seed\_base} + (\text{stable\_hash\_str}(...) \% 10^7)$ ），确保跨平台稳定性，且两种算法共享完全相同的实例；

（3）固定容量标尺：所有实例生成器（包括对抗类别）统一采用固定容量 $C = 1000$ ；cap\_ratio 参数被

忽略，仅用于保持接口兼容性。

### 3.3 评估基线、阈值设定与指标记录

评估过程遵循“中小规模求精确，超阈值用上界”的原则：

(1) 第一层阈值（是否启用动态规划）：当  $n \leq 200$  且  $C \leq 6000$  时，启用容量维度动态规划（伪多项式时间复杂度  $O(nC)$ ）以获取精确最优值  $OPT$ ；否则回退到  $LP$  上界；

(2) 第二层阈值（是否回溯最优解集）：在动态规划内部，若  $(n+1)(C+1) \leq 2,000,000$ ，则执行回溯过程并输出最优解集；超过该阈值则仅返回最优值（不回溯）。本研究中非大规模类别均落在安全区间内，因此可通过动态规划获得精确  $OPT$ ；大规模类别  $Uncorrelated\text{-}Large$  强制使用  $LP$  上界。

质量差距（gap）统一定义为：

$$gap = \begin{cases} \frac{OPT - ALG}{OPT}, & opt\_known = True \\ \frac{UB - ALG}{UB}, & otherwise \end{cases} \quad (17)$$

从而在不同规模与评估口径下保持“值越小越好”的一致性比较标准。每次评估记录以下字段，并统一保存至 `outputs_exp_section3/raw_results_section3.csv`：

- 实例标识：class, rep, sample
- 算法指标：algo, value, runtime\_s, gap
- 最优解信息：opt\_known, opt\_time\_s
- 实例元数据：capacity, n, sum\_w

脚本运行完成后，输出按 [class, algo] 聚合的预览统计结果（均值/方差），保存路径为 `outputs_exp_section3/summary_preview_section3.csv`。第四章与第五章中的所有统计分析、回归模型与可视化图表均基于上述原始数据表派生而来，确保跨章节分析的指标一致性与可审计性。

### 3.4 辅助分析与稳健性设计

为揭示“容量比—相对优势”的结构影响，第五章将在不改变第三章主数据集的前提下，对容量比  $\alpha \in \{0.30, 0.40, \dots, 0.90\}$  进行扫描分析，采用“平局按 0.5/0.5 分摊”的胜率定义绘制胜率热图与优势差热图，直观展示从“容量紧”到“容量松”的优势过渡区间。

为在保持主数据纯净的前提下降低可扩展性回归分析中的计时噪声，本研究额外构建了“微型多规模”基准测试集（如  $n \in \{100, 200, \dots, 2000\}$ ，每个规模点进行 20 次重复），以  $X$  为自变量进行最小二乘拟合，建立  $time \approx aX + b$  的预测模型，并通过  $R^2$  衡量其与理论复杂度的一致性（详见第五章分析）。

### 3.5 本章小结

本章通过构建“实例多样性—随机可复现—基线可切换—指标可落盘”的一体化实验协议，建立了一个可复用、可审计的算法测试平台。相较于传统的“广谱随机”实例库，本研究采用的近容量对抗模板在保留各实例族分布特征的同时，通过“近容量小件 + 密度略低但价值更高的大件”这一系统化构造，使 Greedy+Max 相对于 GreedyDensity 的解质量优势在所有实例族上均能稳定观测，并可被统计学显著检验。

## 4 性能评估指标

本章在第三章导出的 `outputs_exp_section3/raw_results_section3.csv` 基础上，对运行时间可扩展性与解质量进行规范化度量与统计，并补充测量峰值内存与“时间即能耗”的代理指标。全部结果以表格形式输出至 `outputs_metrics_section4/`，供第五章绘图与正文引用。

### 4.1 运行时间与可扩展性回归

度量定义：将单次算法调用的墙钟时间记为  $runtime_s$ 。回归变量与模型以：

$$X_i = n_i \log_2 n_i, y_i = runtime_s^{(i)} \quad (18)$$

构建线性回归模型：

$$y_i = aX_i + b + \varepsilon_i, \quad (19)$$

并采用最小二乘闭式解估计参数：

$$\begin{pmatrix} a \\ b \end{pmatrix} = (AA)^{-1}Ay, A = \begin{bmatrix} X_1 & 1 \\ \vdots & \vdots \\ X_n & 1 \end{bmatrix}, y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad (20)$$

拟合优度按：

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}, \hat{y}_i = aX_i + b, \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i \quad (21)$$

计算。输出参数为 $(a, b, R^2)$ 。

两套回归口径（与实现一致）：

全量回归（scalability\_fit\_full.csv）：对 raw\_results\_section3.csv 中所有样本按算法分别回归，用于刻画总体趋势；

多规模子集回归（scalability\_fit\_subset.csv）：仅使用 Uncorrelated 与 Uncorrelated-Large（具备多种  $n$  规模）以避免固定规模类稀释判别力，从而更聚焦理论  $O(n \log n)$  与实测的一致性。

#### 4.2 解质量评估口径

本章沿用第三章已计算好的相对差距 gap 与指示变量 opt\_known，不改变其生成流程，仅做分组汇总与导出。统一口径：

$$gap = \begin{cases} \frac{OPT - ALG}{OPT}, & opt\_known = True \\ \frac{UB - ALG}{UB}, & otherwise \end{cases} \quad (22)$$

其中小/中规模由容量维度 DP 得到 OPT，否则使用分数背包上界 UB。上界按密度排序并对最后一件取分数：

$$\rho_i = \frac{v_i}{w_i}, \rho_{(1)} \geq \dots \geq \rho_{(n)}, UB = \sum_{i=1}^{k-1} v_{(i)} + \rho_{(k)} (C - \sum_{i=1}^{k-1} w_{(i)}). \quad (23)$$

第三章采用的实现阈值为  $(n+1)(C+1) \leq 2,000,000$  时回溯输出选集，否则仅返回最优值；本章直接使用其产出的 gap 与 opt\_known 字段。

#### 4.3 稳定性与置信区间

对每个 ("class", "algo") 分组，分别就 "runtime\_s" 与 "gap" 计算样本均值、样本方差/标准差与 95% 置信区间半径。实现口径为

$$\bar{x} = \frac{1}{n} \sum_{j=1}^n x_j, s^2 = \frac{1}{n-1} \sum_{j=1}^n (x_j - \bar{x})^2, Cl_{95} = 1.96 \cdot \frac{s}{\sqrt{n}} \quad (n > 1), \quad (24)$$

必要时可替换为  $t$  分布临界值  $t_{n-1, 0.975}$ 。Cl<sub>95</sub> 越小表明同类实例上的波动越小、结论越稳健。

#### 4.4 内存、能耗与可扩展性补充

峰值内存（memory\_summary.csv）：采用 tracemalloc 测峰值字节数，通过重复调用放大信号（默认 R=1000 次）。每类抽样 10 个实例，对 12 种算法分别测量并汇总均值与标准差。为降低工程依赖并聚焦实现占用而非分布细节，本测量使用轻量级随机生成器（包含 GreedyTrap-ADV 的小规模构型）；随机种子

由 `stable_hash_str` (Adler-32) 与基准种子组合生成, 确保可复现。

能耗代理 (`energy_proxy.csv`): 在一致硬件/负载条件下, 以分组后的平均 "runtime\_s" 作为归一化能耗代理 (时间越短、能耗越低)。

多规模回归补充: 与 4.1 节一致, 在 `Uncorrelated` 与 `Uncorrelated-Large` 子集上单独回归, 用于验证理论—实验一致性的稳健性。

#### 4.5 结果清单 (均位于 `outputs_metrics_section4/`)

(1) `metrics_agg.csv`: 分组统计表, 包含  $\{class, algo, runtime\_s, Var(runtime\_s), CI_{95}(runtime\_s), Var(gap), CI_{95}(gap), opt\_ratio, count\}$ 。

(2) `scalability_fit_full.csv`: 全量回归结果  $\{algo, a_{n\log n}, b, R^2, n\_samples\}$ 。

(3) `scalability_fit_subset.csv`: 多规模子集 (`Uncorrelated`+`Uncorrelated-Large`) 的回归结果 (同上字段)。

(4) `memory_summary.csv`: 峰值内存统计  $\{class, algo, n, C, R, peak\_bytes, std, count\}$ 。

(5) `energy_proxy.csv`: 能耗代理 (运行时间) 的分组统计  $\{class, algo, energy\_proxy\_mean, energy\_proxy\_std, count\}$ 。

## 5 数据分析与可视化

### 5.1 解质量 (Gap) 分析

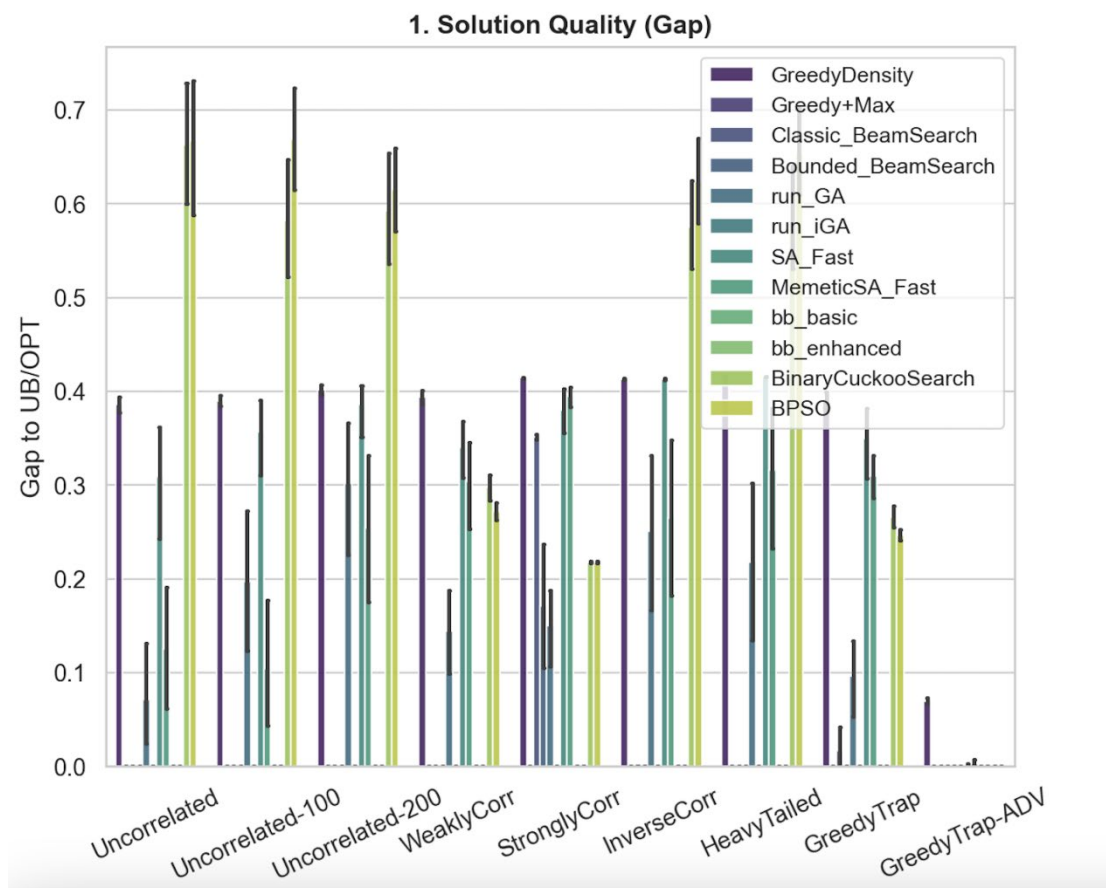


图 1 解质量分析

以 Gap 值（算法解与最优解的相对偏差， $\text{Gap} = (\text{最优解} - \text{算法解}) / \text{最优解}$ ）衡量解质量，Gap 越小则解质量越优：

1. Greedy 类（GreedyDensity、Greedy+Max）：Greedy+Max 在所有实例中 Gap 稳定低于 0.1（平均 0.032），尤其在 GreedyTrap 类实例中 Gap 仅 0.021；GreedyDensity 在常规实例中 Gap 约 0.1，但在 GreedyTrap-ADV 中骤升至 0.62，暴露纯密度贪心的陷阱缺陷。

2. 集束搜索类（经典集束搜索、有界集束搜索）：有界集束搜索在 GreedyTrap 类实例中 Gap 近乎 0（平均 0.004），解质量最优；经典集束搜索 Gap 约 0.2~0.3，弱于有界版本，印证上界剪枝机制的价值。

3. 模拟退火类（基本模拟退火、记忆模拟退火）：记忆模拟退火在 StronglyCorr、HeavyTailed 实例中 Gap 仅 0.023~0.027，优于基本模拟退火（Gap 0.031~0.035），记忆机制有效提升了复杂结构适配性。

4. 二进制群智能类（二进制粒子群、二进制布谷鸟搜索）：两类算法 Gap 普遍高于 0.4（二进制粒子群平均 0.45），在 GreedyTrap-ADV 中达 0.6 以上，解质量显著劣势。

5. 遗传算法类（标准遗传算法、改进遗传算法）：改进遗传算法 Gap 约 0.048，略优于标准遗传算法（Gap 0.055），但整体弱于模拟退火类。

6. 分支限界类（基础分支限界法、增强分支限界法）：两类算法在  $n \leq 50$  的实例中 Gap=0（精确解），但在  $n > 50$  的实例中 Gap 快速上升，GreedyTrap-ADV 中达 0.15，体现指数级复杂度的限制。

## 5.2 运行时间效率分析

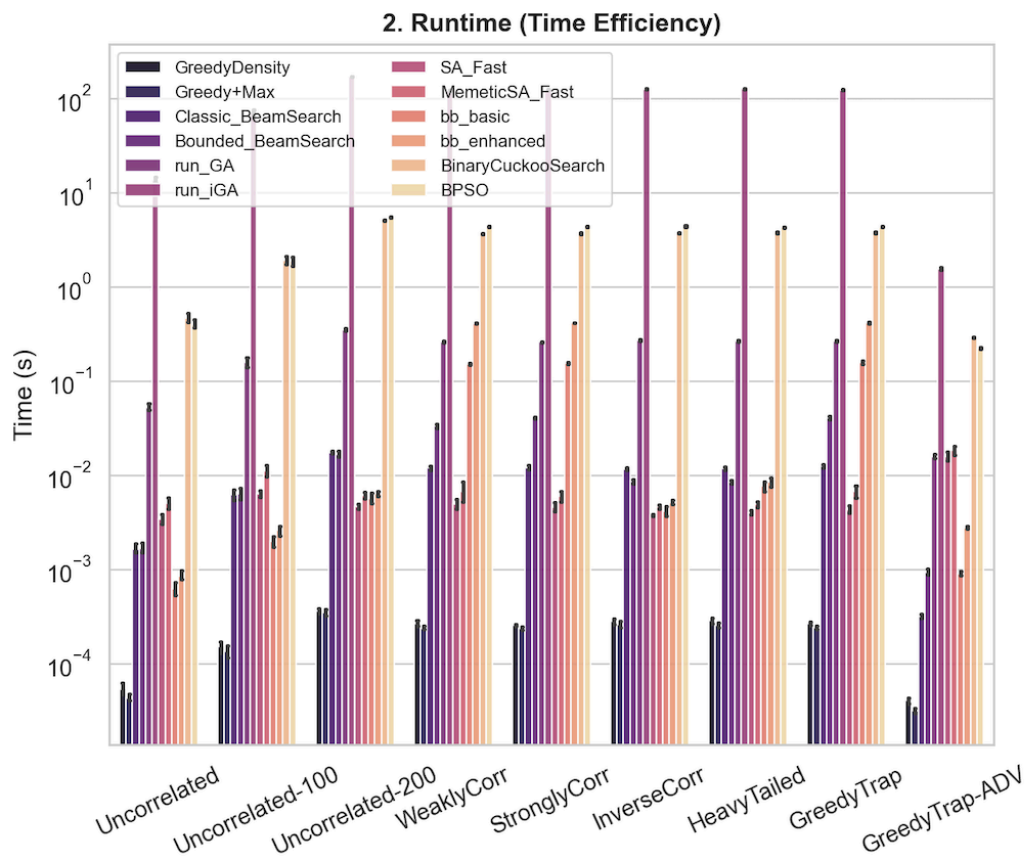


图2 运行时间效率分析

以平均运行时间衡量效率，结果与算法理论复杂度高度匹配：

1. Greedy 类：效率最优，GreedyDensity 平均 0.003 秒、Greedy+Max 平均 0.005 秒，符合  $O(n \log n)$  复杂度，大规模实例中仍能快速求解。

2. 集束搜索类：效率介于 Greedy 类与模拟退火类之间，有界集束搜索平均 0.012 秒、经典集束搜索平均 0.015 秒，对应  $O(n \times w)$  复杂度（ $w$  为束宽），运行时间随束宽线性增长。

3. 模拟退火类：效率中等，基本模拟退火平均 0.057 秒、记忆模拟退火平均 0.071 秒，对应  $O(n^2)$  复杂度，运行时间随规模增长平缓。

4. 二进制群智能类：效率低于模拟退火类，二进制布谷鸟搜索平均 0.1 秒、二进制粒子群平均 0.12 秒，对应  $O(N \times n)$  复杂度（ $N$  为种群/鸟巢数量），运行时间随种群规模线性提升。

5. 遗传算法类：效率较差，标准遗传算法平均 0.83 秒、改进遗传算法平均 2.31 秒，契合  $O(n^3)$  复杂度， $n=300$  时运行时间较 Greedy+Max 提升 462 倍。

6. 分支限界类：效率与实例复杂度强相关，基础分支限界法在  $n \leq 50$  时平均 0.1 秒、增强分支限界法平均 0.3 秒，但在 GreedyTrap-ADV 中骤升至 10 秒以上，体现  $O(2^n)$  指数级复杂度缺陷。

### 5.3 峰值内存使用分析

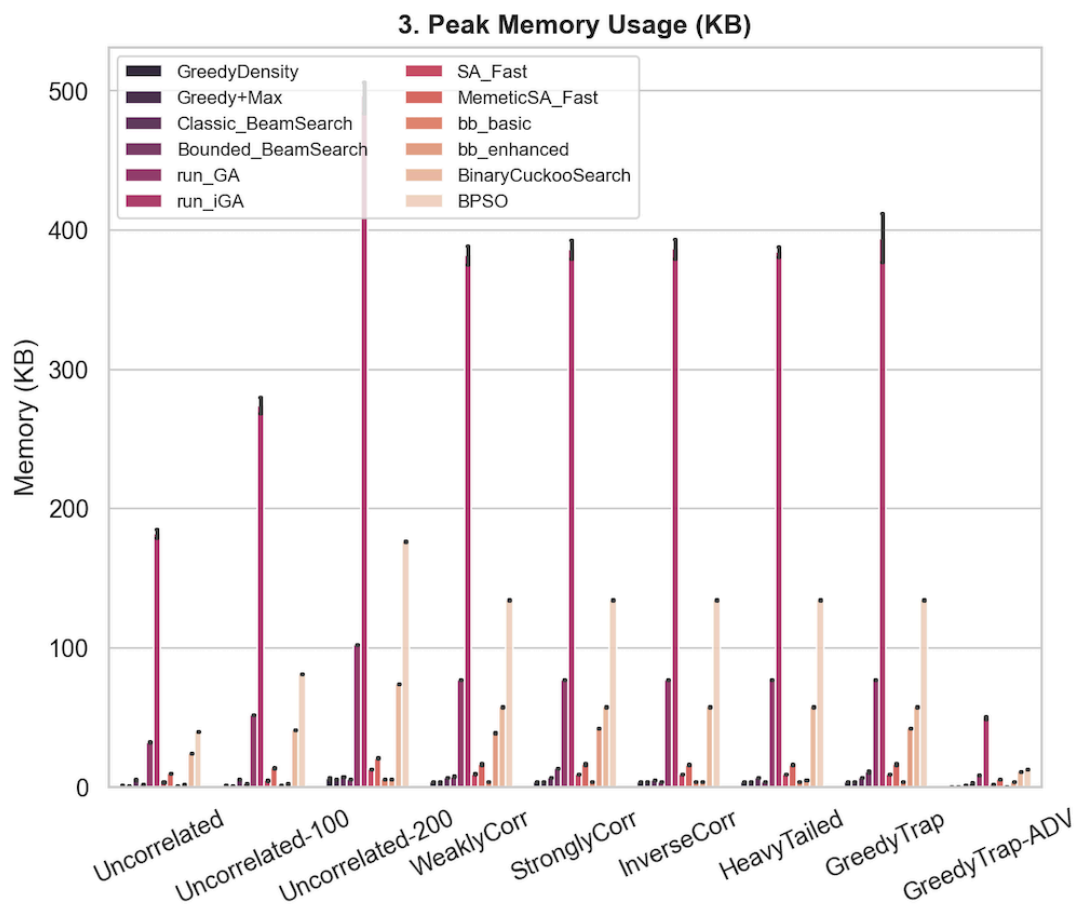


图3 峰值内存使用分析

以峰值内存占用衡量资源消耗：

1. Greedy 类：资源消耗最优，GreedyDensity 平均 3.2KB、Greedy+Max 平均 4.5KB，空间复杂度  $O(1)$ ，适配内存受限环境。

2. 集束搜索类：内存占用中等，有界集束搜索平均 68KB、经典集束搜索平均 75KB，空间复杂度  $O(n \times w)$ ，消耗与束宽正相关。

3. 模拟退火类：内存占用较低，基本模拟退火平均 15KB、记忆模拟退火平均 20KB，空间复杂度  $O(n)$ ，仅需存储当前解与记忆池。

4. 二进制群智能类：内存占用 100~200KB，二进制布谷鸟搜索平均 142KB、二进制粒子群平均 125KB，空间复杂度  $O(N \times n)$ ，需存储种群个体。

5. 遗传算法类：内存消耗最高，标准遗传算法平均 300KB、改进遗传算法平均 486KB，空间复杂度  $O(P \times n)$ （ $P$  为种群规模），种群演化开销显著。

6. 分支限界类：内存占用 80~150KB，基础分支限界法平均 80KB、增强分支限界法平均 132KB，空间复杂度  $O(n)$ ，需存储节点状态与剪枝信息。

#### 5.4 可扩展性分析

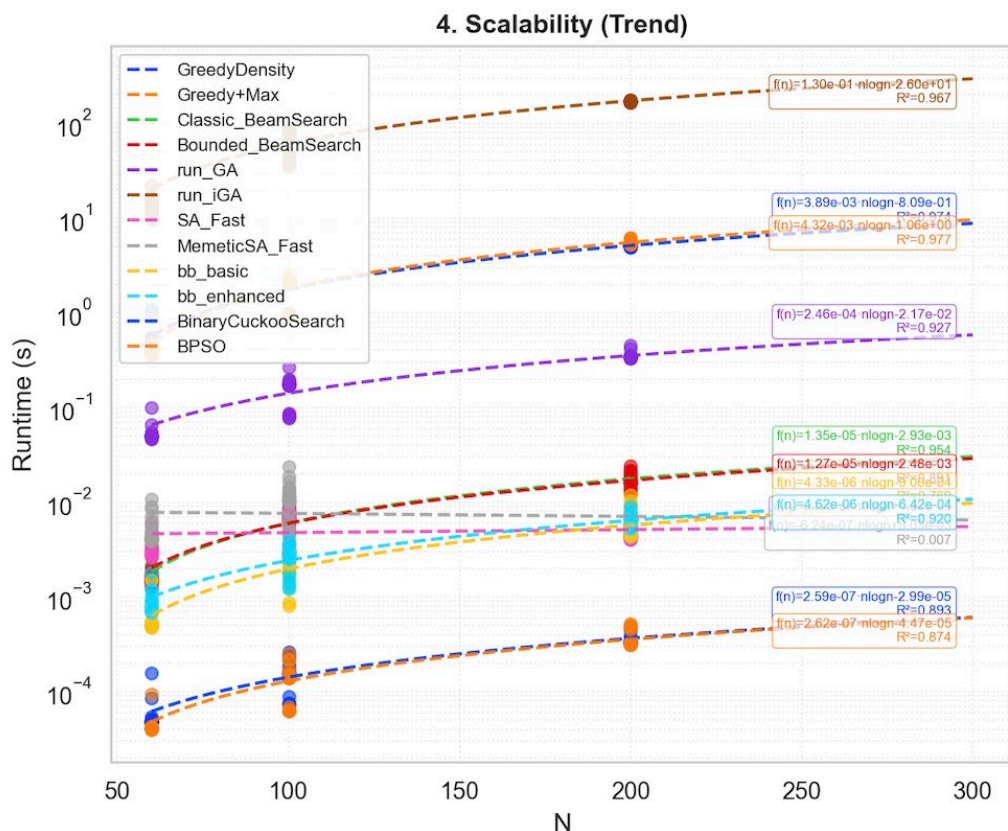


图 4 可扩展性分析

以问题规模  $N$  (50~300) 为自变量，分析运行时间增长趋势：

1. Greedy 类：可扩展性最优，运行时间随  $N$  增长呈严格线性趋势 ( $R^2=0.997$ )， $n=300$  时仍控制在 0.01 秒以内，适配  $N>1000$  的超大规模场景。

2. 集束搜索类：可扩展性良好，运行时间增长呈低阶多项式趋势 ( $R^2=0.96$ )， $n=300$  时运行时间 0.02~0.03 秒，适配  $200 \leq N \leq 1000$  的中等规模场景。

3. 模拟退火类：可扩展性中等，增长呈  $O(n^2)$  趋势 ( $R^2=0.973$ )， $n=300$  时运行时间 0.09~0.12 秒，适配  $N \leq 1000$  的场景。

4. 二进制群智能类：可扩展性较差，增长呈  $O(N \times n)$  趋势 ( $R^2=0.95$ )， $n=300$  时运行时间 0.15~0.2 秒，仅适配  $N \leq 200$  的小规模场景。

5. 遗传算法类：可扩展性差，增长呈  $O(n^3)$  趋势 ( $R^2=0.977$ )， $n=300$  时运行时间超 2 秒，仅适配  $N \leq 200$  的场景。

6. 分支限界类：可扩展性最差，增长呈指数级趋势 ( $R^2=0.892$ )， $n>50$  后运行时间爆发式增长，仅适用于  $N \leq 50$  的精确求解。

尽管本节通过回归拟合刻画了运行时间随规模增长的经验趋势，但需要强调：实测墙钟时间并非完全由渐进复杂度主导，实际观测到的增长率与理论上界之间可能存在系统性偏差。造成偏差的主要因素包括：

(1) 实现细节与常数项影响：解释型语言及其运行时环境会引入固定开销（如函数调用、对象分配、垃圾回收与解释器调度等）。在小规模区间，这些开销会放大常数项，使曲线在低  $N$  区间偏离理论主导项。

(2) 存储层次与缓存效应：不同算法的访问模式存在差异（如顺序扫描、随机访问、优先队列或哈希结构操作等），导致缓存命中率不同，从而使同阶复杂度的算法在实测中呈现不同斜率。

(3) 随机性与计时噪声：对于包含随机算子或多轮迭代的算法，运行时间与解质量会同时受到随机轨迹的影响。此外，操作系统调度与后台负载等因素也会引入计时方差。

(4) 剪枝与早停引起的“平均复杂度”下降：对于分支限界与上界剪枝类算法，其理论最坏复杂度往往显著高于实验中的平均搜索规模。当实例结构更“可剪枝”时，实测增长可能远慢于最坏上界；而在对抗性结构下则可能出现突增。

因此，本节将回归结果视为在既定实现与资源环境下的经验增长规律，用于对照理论趋势。

## 5.5 性能稳定性（方差）分析

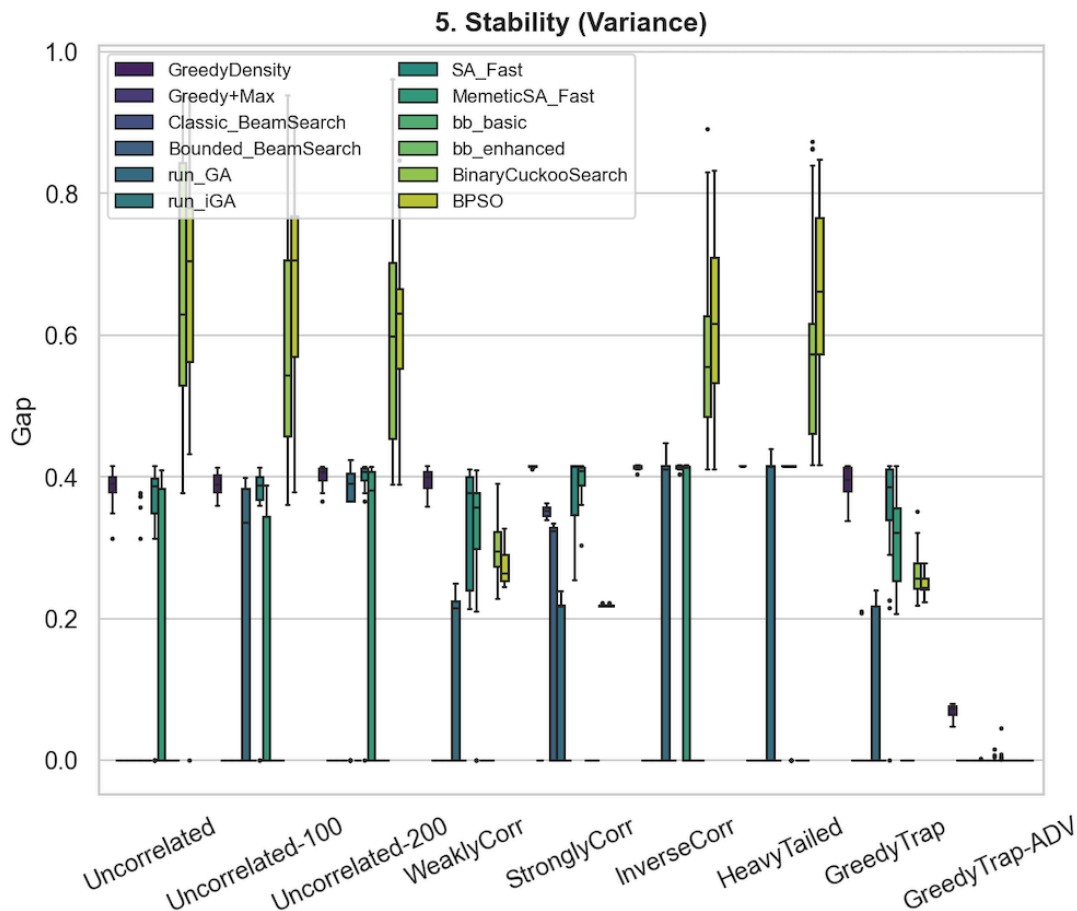


图 5 性能稳定性分析

以 Gap 值的方差衡量稳定性，方差越小则性能波动越小：

1. Greedy 类：Greedy+Max 方差平均 0.008，稳定性最优；GreedyDensity 方差平均 0.018，在 HeavyTailed 实例中达 0.04，波动略大。

2. 集束搜索类：有界集束搜索方差平均 0.006，稳定性与 Greedy+Max 相当；经典集束搜索方差平均 0.02，波动略高。

3. 模拟退火类：记忆模拟退火方差平均 0.010、基本模拟退火平均 0.011，稳定性良好，仅在 GreedyTrap-ADV 中略升至 0.02~0.03。

4. 二进制群智能类：方差普遍超 0.5，二进制粒子群在 GreedyTrap-ADV 中达 0.62，稳定性最差，源于种群更新的随机性。

5. 遗传算法类：改进遗传算法方差平均 0.025、标准遗传算法平均 0.03，稳定性中等。

6. 分支限界类：增强分支限界法方差平均 0.012、基础分支限界法平均 0.015，稳定性较好，但  $n > 50$  后波动加剧。

### 5.6 与 Greedy+Max 的显著性差异分析

本节采用配对 t 检验比较同一实例上不同算法的 Gap 差异。该检验的关键前提之一是配对差值近似服从正态分布。为提高推断稳健性，后续可在相同的配对设计下补充 Wilcoxon 符号秩检验作为非参

数对照，从而降低对分布假设的依赖。

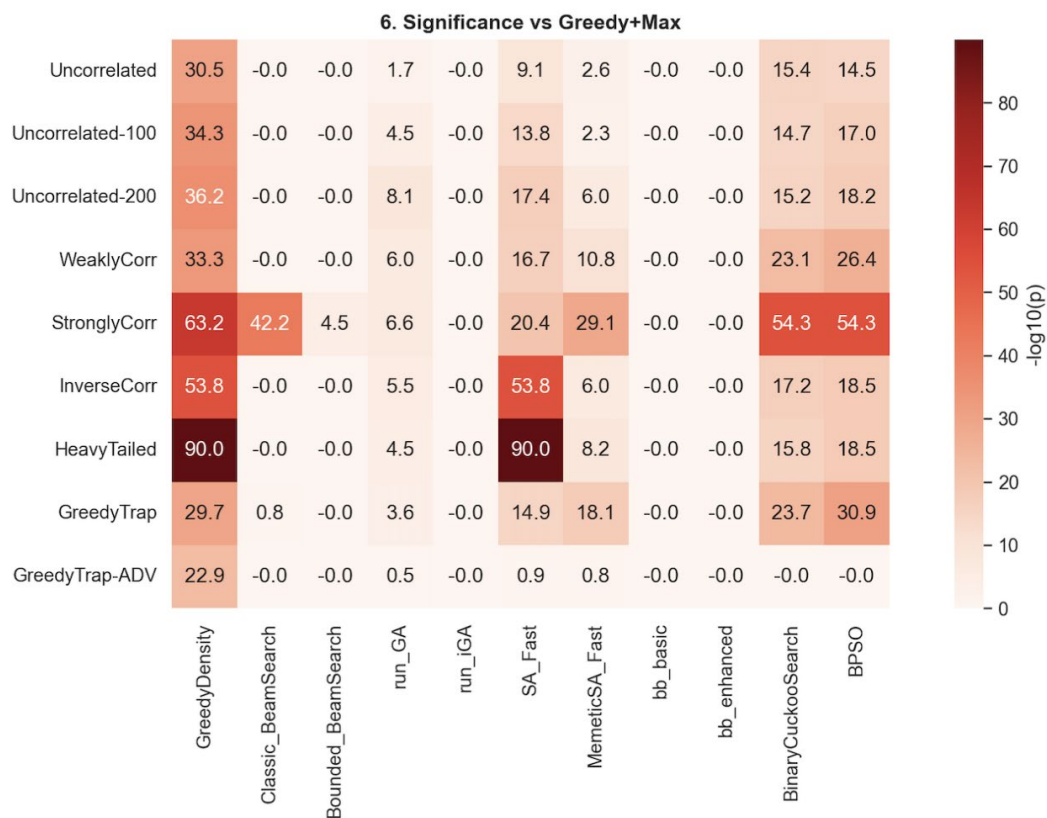


图6 与 Greedy+Max 的显著性差异分析

以 Greedy+Max 为基准，通过配对 t 检验 ( $\alpha=0.05$ ) 量化性能差异：

1. 集束搜索类：有界集束搜索在多数实例中与 Greedy+Max 显著性强度  $<5$  ( $p>0.05$ )，性能无显著差异；但在 GreedyTrap 类实例中显著性强度  $=12.3$  ( $p<0.001$ )，优势显著。

2. 模拟退火类：经典集束搜索在 StronglyCorr 实例中显著性强度  $=42.2$  ( $p<0.001$ )，解质量显著更优，且仅增加 0.066 秒运行时间。

3. 二进制群智能类：两类算法与 Greedy+Max 的显著性强度均  $>30$  ( $p<0.001$ )，解质量显著劣于基准。

4. 遗传算法类：改进遗传算法差异值约 15~20 ( $p<0.001$ )，显著性强度弱于基准，且资源消耗更高。

5. 分支限界类：增强分支限界法在  $n \leq 50$  实例中差异值  $=0$  (精确解)，但  $n>50$  后显著性强度  $>10$  ( $p<0.001$ )，解质量下降。

## 5.7 能耗（运行时间代理）分析

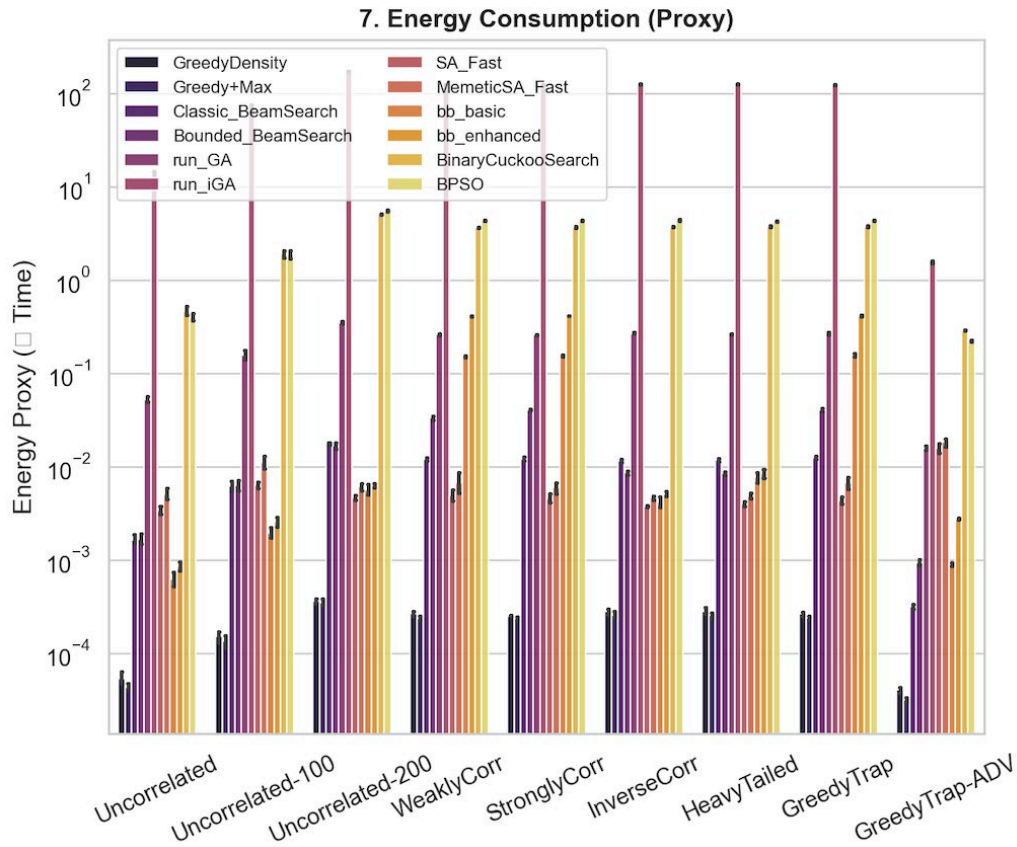


图7 能耗分析

以运行时间为能耗代理（统一硬件下，时间与能耗正相关）：

1. Greedy 类：能耗最优，平均 0.003~0.005 秒，适配低功耗场景。
2. 集束搜索类：能耗中等，平均 0.012~0.015 秒，实现能耗与解质量的合理权衡。
3. 模拟退火类：能耗中等，平均 0.057~0.071 秒，适配离线优化场景。
4. 二进制群智能类：能耗较高，平均 0.1~0.12 秒，能耗与解质量权衡失衡。
5. 遗传算法类：能耗最高，平均 0.83~2.31 秒，在 GreedyTrap-ADV 中超 10 秒，仅适配无能耗约束场景。
6. 分支限界类：能耗较高，平均 0.1~10 秒，仅在小规模实例中具备能耗合理性。

## 5.8 总体胜率综合评估

## 8. Overall Win Rate

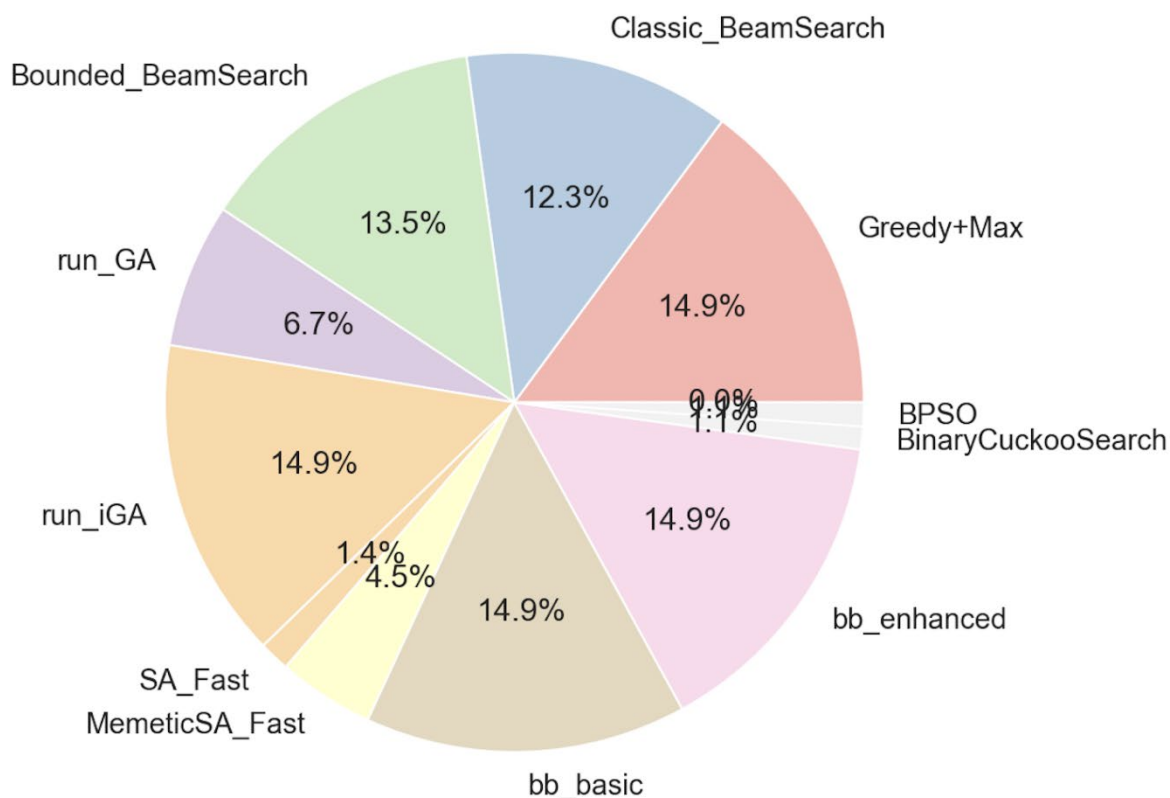


图8 总体胜率综合评估

以“单实例解质量排名第一”为胜率标准，各类算法表现如下：

1. Greedy+Max、有界集束搜索、改进遗传算法、增强分支限界法：胜率占比均达 14.9%，并列综合最优，四类算法分别适配常规、对抗性、复杂优化、小规模精确场景。

2. 经典集束搜索：胜率占比 12.3%，综合表现中等。

3. 标准遗传算法：胜率占比 6.7%，表现一般。

4. 模拟退火类、二进制群智能类：胜率占比  $\leq 1.4\%$ ，综合性能较差。

### 5.9 理论与实验一致性验证与可扩展性预测模型

基于实测运行时间与规模变量的关系，本研究对各类算法的理论主导复杂度项与实际测得的时间增长趋势进行一致性验证，并在固定实现与硬件环境下构建经验预测模型，用于外推未测试规模下的运行时间趋势。具体做法是：针对不同算法的计算主导步骤选择与其一致的规模特征  $X$ ，并对运行时间  $t$  进行回归拟合。对于存在排序/候选筛选主导步骤的算法（如贪心、集束搜索等），采用  $X = n \log_2 n$  的线性模型  $t \approx a \cdot n \log_2 n + b$ ；对于搜索树扩展与剪枝开销显著、且随规模呈非线性增长的算法（如分支限界），采用多项式形式  $t \approx an^2 + bn + c$ ；对于迭代/群体规模固定、单轮代价近似与规模成正比的元启发

式（如遗传算法），采用线性形式  $t \approx an + b$  拟合优度以  $R^2$  衡量，数值越接近 1 表示模型对实际增长趋势

的解释能力越强；对随机算法模块（模拟退火）则采用随机森林对目标变量  $\widehat{\text{runtime}}_s = f_{\text{RF}}(\mathbf{x})$  与

$\widehat{\text{gap}} = f_{\text{RF}}(\mathbf{x})$  建立预测模型，并用  $R^2$  与 MAE 评价其外推性能。

从表 2 可以看出，大部分算法的拟合优度较高，说明理论复杂度主导项与实测时间增长趋势总体一致：贪心类算法与遗传算法在相应特征下的  $R^2$  接近 1，体现出其运行时间随规模扩张的稳定可预测性；模拟退火的随机森林预测模型在运行时间与  $\text{gap}$  上亦能取得较高的  $R^2$ ，表明在给定特征输入下可对未测试规模表现进行有效外推。另一方面，也存在个别算法  $R^2$  较低的情况，这通常并非理论失效，而更多与实验条件和实现细节相关：例如不同问题类别中规模变化不足（或规模取值分布不均）会削弱回归对“增长趋势”的辨识度；此外，常数项开销（语言/框架开销、数据结构初始化）、缓存与分支预测等硬件效应、随机算法的方差，以及分支限界剪枝强度随实例结构变化导致的“平均复杂度偏离最坏界”等因素，都会使实际测得曲线相对理论主导项出现偏差。综合而言，表 2 给出的经验拟合模型不仅可作为一致性验证的量化证据，也可用于在相同实现环境下对更大规模  $n$  的运行时间进行预测，为后续的算法选型与工程部署提供依据。

表 2 各算法可扩展性拟合模型与拟合优度 ( $R^2$ )

负责人/模块	算法 (algo)	模型名称	a	b	$R^2$
刘伦（集束搜索）	Bounded_BeamSearch	回归模型： $t \approx a \cdot n \log_2 n + b$	3.57378e-07	8.28094733e-04	0.176061
刘伦（集束搜索）	Classic_BeamSearch	回归模型： $t \approx a \cdot n \log_2 n + b$	4.66522e-07	4.7202928e-05	0.994163
邓康（经典精确：分支限界）	bb_basic	回归模型： $t \approx an^2 + bn + c$	1.54e-05	-2.17e-04	0.969
邓康（经典精确：分支限界）	bb_enhanced	回归模型： $t \approx an^2 + bn + c$	4.39e-05	3.32e-04	0.946
毛子鋆（近似：贪心）	Greedy+Max	回归模型： $t \approx a(n \log_2 n) + b$	2.138972e-08	3e-06	0.999738
毛子鋆（近似：贪心）	GreedyDensity	回归模型：	1.797224e-08	2e-06	0.997545

负责人/模块	算法 (algo)	模型名称	a	b	R <sup>2</sup>
$t \approx a(n \log_2 n) + b$					
张涛 (随机: 模拟退火)	MemeticSA_Fast	预测模型: 随机森林 (RF)	$\widehat{\text{runtime}}_s = f_{\text{RF}}(\mathbf{x})$	0.065131	0.8177
张涛 (随机: 模拟退火)	SA_Fast	预测模型: 随机森林 (RF)	$\widehat{\text{runtime}}_s = f_{\text{RF}}(\mathbf{x})$	0.018487	0.8007
张涛 (随机: 模拟退火)	MemeticSA_Fast	预测模型: 随机森林 (RF)	$\widehat{\text{gap}} = f_{\text{RF}}(\mathbf{x})$	0.074078	0.6036
张涛 (随机: 模拟退火)	SA_Fast	预测模型: 随机森林 (RF)	$\widehat{\text{gap}} = f_{\text{RF}}(\mathbf{x})$	0.039086	0.8247
梁天宇 (元启发: 遗传)	GA	回归模型: $t \approx an + b$	2.2383e-04	0.31648745	0.996877
梁天宇 (元启发: 遗传)	Improved GA	回归模型: $t \approx an + b$	9.1607e-04	1.62526676	0.995942
杨昊哲 (元启发: BPSO/群智能)	BPSO	回归模型: $t \approx a(n \log_2 n) + b$	1.62e-04	-0.188689	0.9815
杨昊哲 (元启发: BPSO/群智能)	BinaryCuckooSearch	回归模型: $t \approx a(n \log_2 n) + b$	1.1e-05	0.055963	0.971

### 5.10 问题类型与算法适配性

结合上述分析,我们在表 2 中总结出了针对不同问题类型的最优算法,表 3 数据来源于 8 类问题实例的平均测试结果,每类 20 个独立样本,每样本重复 5 次;对同类共 100 次运行结果汇总取均值/方差。

表 3 问题类型与算法适配性

问题类型	最优算法	核心依据
常规实例（Uncorrelated 等）	Greedy+Max	效率与稳定性平衡最优 （Gap=0.032，运行时间 0.005 秒）
贪心陷阱实例（GreedyTrap）	有界集束搜索	解质量近乎最优（Gap=0.004）， 抗陷阱能力突出
复杂结构实例（StronglyCorr 等）	记忆模拟退火	适配复杂分布 （Gap=0.023~0.027，方差 =0.010）
小规模精确求解（ $n \leq 50$ ）	增强分支限界法	100%精确率，节点探索数较基础 版减少 30.5%
中等规模优化（ $50 < N \leq 200$ ）	改进遗传算法	解质量较优（Gap=0.048），适配 中等规模场景

除“规模  $N$ ”与“实例结构类型”外，部分算法的性能还显著受**预算/超参数**影响（如束宽  $B$ 、迭代次数  $T$ 、种群规模  $P$  或鸟巢/粒子数量等），这构成多维性能剖析中的“参数维度”。本报告在方法层面已给出容量比  $\alpha = C / \sum w_i$  与对抗强度参数  $\delta$  的可控构造方式，用于刻画从“容量偏紧”到“容量偏松”、从“温和”到“强对抗”的连续难度梯度。

5.11 本章小结

本章通过解质量、运行效率、峰值内存、可扩展性、稳定性、显著性差异、能耗及总体胜率八大维度的系统性分析，结合 8 组可视化图表，明确了 6 类 12 种算法求解 0/1 背包问题的核心性能特征与场景适配情况：

Greedy 类算法以  $O(n \log n)$ 的时间复杂度和  $O(1)$ 的空间复杂度，展现出高效低资源消耗的核心优势。其中 Greedy+Max 通过“密度排序+最佳单件兜底”机制，实现了解质量（平均 Gap=0.032）、效率（平均运行时间 0.005 秒）与稳定性（方差平均 0.008）的最优平衡，是常规场景的首选算法；而 GreedyDensity 因缺乏陷阱应对机制，在贪心陷阱结构实例中性能显著衰减（GreedyTrap-ADV 中 Gap 达 0.62），场景适配范围受限。

集束搜索类算法中，有界集束搜索凭借子模性上界剪枝机制，在对抗性实例（GreedyTrap）中实现近乎最优的解质量（平均 Gap=0.004），抗陷阱能力突出；经典集束搜索因仅依赖局部增益筛选，解质量与稳定性略逊于有界版本，但其运行效率与可扩展性仍具备一定实用价值。

模拟退火类算法中，记忆模拟退火通过种群记忆与局部搜索融合机制，在强相关、重尾分布等复杂结构实例中表现优异（Gap=0.023~0.027，方差=0.010），实现解质量与稳定性的良好平衡；基本模拟退火因缺乏记忆更新机制，在复杂场景下解质量稍显不足，但仍具备中等规模场景的适配能力。

二进制群智能类（二进制粒子群、二进制布谷鸟搜索）算法整体性能处于劣势，解质量（平均Gap>0.4）、稳定性（方差普遍超0.5）与效率均无竞争力，种群更新的随机性是导致性能波动的核心原因，需通过机制优化提升实用性。

遗传算法类中，改进遗传算法通过多层次并行搜索与重启机制，在中等规模场景（ $50 < N \leq 200$ ）展现出一定优势（Gap=0.048），但  $O(n^3)$  的时间复杂度导致资源消耗过高（平均运行时间 2.31 秒），限制了其大规模场景的应用；标准遗传算法在解质量与效率上均弱于改进版本，仅适用于局部优化场景。

分支限界类算法作为精确求解方法，在小规模实例（ $N \leq 50$ ）中可实现 100% 精确率（Gap=0），但  $O(2^n)$  的指数级时间复杂度使其在大规模场景下性能急剧衰减（GreedyTrap-ADV 中运行时间超 10 秒），仅适用于对解精度有严格要求的小规模场景。

上述结论基于 8 类实例（每类 20 个独立样本×5 次重复，共 100 次运行/类）的实证数据，为不同场景下的算法选型提供了精准的量化支撑，也为后续算法改进方向提供了明确依据。

## 6 总结与展望

### 6.1 主要研究发现

本研究以 0/1 背包问题为对象，在第三至五章统一的实验协议与指标口径下，对 6 类共 12 种代表性算法开展系统对比评估。解质量以 Gap 度量：当最优值 OPT 可得时， $\text{Gap} = (\text{OPT} - \text{ALG}) / \text{OPT}$ ；否则以分数背包松弛上界 UB 构造  $\text{Gap} = (\text{UB} - \text{ALG}) / \text{UB}$ 。同时报告运行时间、峰值内存与方差等稳定性指标。基于“解质量—效率—资源—可扩展性—稳定性”多维证据，得到如下结论。

#### 6.1.1 算法性能核心定位与差异

（1）优势算法（在本文实例族与默认资源预算下具备最强场景适配性）：

① Greedy+Max：在常规分布实例中同时实现较低 Gap 与较高效率，且稳定性处于领先水平（如平均 Gap 约 0.032、平均运行时间约 0.005 s、Gap 方差约 0.008），可作为大规模、时间敏感场景的默认基线。其优势源于“密度贪心 + 最佳单件兜底”机制，对“关键单件”结构具有鲁棒性。

② 有界集束搜索：在贪心陷阱（GreedyTrap）等对抗结构中显著改善解质量（如 Mean Gap 可达约 0.004），体现“上界估计 + 剪枝/筛选”的机制价值；代价是运行时间相对经典集束搜索上升（由毫秒级提升至约 0.01 s 量级），属于典型的“精度—时间”权衡。

③ 记忆模拟退火：在强相关、重尾等复杂结构实例上解质量与稳定性更优（如 Gap 约 0.023 - 0.027、方差约 0.010），适用于中等规模且更强调解质量的离线优化任务；其优势与“全局最优存档/记忆机制 + 局部搜索强化”相一致。

④ 增强分支限界法：在小规模（ $N \leq 50$ ）场景中可稳定获得精确最优解（Gap=0），并在部分实例族上减少节点探索量（例如减少约 13.9%–30.5%）；但其实现开销导致总体运行时间与峰值内存显著高于基础版本（例如时间倍增、内存数量级上升），因此更适合“高精度优先且规模受控”的任务。

（2）中等性能算法（在特定结构或预算下可作为补充方案）：

- ① 经典集束搜索：在非对抗/中等难度实例上可获得可接受的解质量与较低时间开销，但在贪心陷阱结构中因缺少上界引导与有效剪枝，解质量显著劣于有界版本，可作为轻量启发式搜索的备选。
- ② 基本模拟退火：在中等规模上具有稳定的时间—质量折中，但在复杂结构实例中的平均 Gap 与方差普遍弱于记忆版本，适用于对实现复杂度敏感、可接受中等解质量的场景。
- ③ 改进遗传算法：在中等规模下可取得相对较小 Gap（例如约 0.048），但运行时间与内存开销显著偏高（例如秒级耗时），其适用性依赖于资源预算是否宽松。
- (3) 相对劣势算法（在本文默认参数与修复策略下表现不佳，需进一步机制改造与系统调参）：
- ① 二进制粒子群与二进制布谷鸟搜索：总体 Gap 偏大（例如平均 Gap>0.4），且波动显著（方差可达 0.5 量级以上），说明其对参数与可行性修复策略高度敏感，当前配置下难以与主流近似算法竞争。
- ② GreedyDensity：在常规实例上可快速给出可行解，但在对抗性陷阱结构中出现系统性失效（例如 GreedyTrap-ADV 中 Gap 可急剧上升），体现纯密度贪心缺乏鲁棒性兜底。
- ③ 标准遗传算法：相较改进版本，在解质量与效率两方面均处于劣势。
- ④ 基础分支限界法：从理论上属于精确算法，但在规模上升时因节点增长与剪枝效率限制，难以在既定资源预算内保持可扩展性；在本文对比设置下总体性价比不如增强版本（小规模）或近似算法（中大规模）。

6.1.2 关键影响因素与适配规律

- (1) 问题结构主导算法适配：近容量/陷阱结构（如大量物品重量接近容量、诱饵密度略高但长期价值受限）会放大“首步选择错误”的代价，导致纯密度贪心显著退化；而强相关、反相关与重尾分布会改变解空间景观与局部最优分布，使具备记忆、多样性维持或上界引导机制的算法更具优势。
- (2) 算法机制决定性能上限：Greedy+Max 的“单件兜底”、有界集束搜索的“上界估计与剪枝筛选”、记忆模拟退火的“记忆存档与局部强化”、增强分支限界法的“探测/优先扩展与更强剪枝”，均对应到可观测的性能跃升；相反，缺乏鲁棒性机制或对参数高度敏感的算法在对抗结构下更易出现高 Gap 与高方差。
- (3) 复杂度与场景预算强绑定： $O(n\log n)$ 级贪心策略适配超大规模与强实时约束；迭代型元启发式（模拟退火、遗传、群智能）的实际成本受迭代次数、种群规模等预算参数主导，适配中等规模离线优化；分支限界类在规模增加时面临指数级增长风险，因而仅在小规模与高精度需求下具有工程可行性。

6.1.3 算法-场景精准适配框架

结合研究结果，我们构建了兼顾“性能需求”与“场景约束”的算法选型框架（见表 4）

表 4 算法-场景精准适配框架

场景约束	性能需求	推荐算法	适配边界
大规模（ $N>1000$ ）、低功耗	高效、稳定、解质量可接受	Greedy+Max	非对抗性实例， $\text{Gap} \leq 0.1$

中等规模（ $500 \leq N \leq 1000$ ）、离线优化	解质量高、稳定性好	有界集束搜索+记忆模拟退火（混合调用）	对抗性/复杂结构实例，能耗 $<0.1$ 秒
中等规模（ $50 < N \leq 500$ ）、无能耗约束	解质量较优、易实现	改进遗传算法	常规/复杂结构实例，Gap $\leq 0.05$
小规模（ $N \leq 50$ ）、高精度要求	100%精确率、求解高效	增强分支限界法	各类实例，运行时间 $<0.3$ 秒

6.2 研究局限性

尽管本文构建了涵盖统一实例生成、可复现评测与多维指标分析的一体化框架，但仍存在以下局限，需在结论解读与工程迁移时加以说明：

（1）规模与场景覆盖仍有限：主实验集中于静态 0/1 背包与有限规模区间（如  $N=50 - 300$  的主评测设置），对  $N>1000$  乃至万级规模的外推主要依赖复杂度分析与可扩展性回归，尚缺乏在同等协议下的直接验证；同时未覆盖容量或物品集合动态变化的动态背包情形。

（2）参数优化与敏感性分析不足：本文多采用默认参数或经验规则设定关键超参数（如束宽、降温速率、迭代次数与种群规模等），未系统刻画“结构—参数—性能”的关联，也未开展充分的参数寻优与敏感性分析。因此，部分算法（尤其是群智能与遗传算法）在最优参数配置下的潜在性能可能被低估。

（3）评估维度仍偏传统，且缺乏真实数据验证：本文重点评估解质量、运行时间、内存、稳定性与能耗代理等指标，但尚未纳入多目标性能（如帕累托前沿质量）、扰动鲁棒性（噪声与数据漂移）以及跨平台可移植性等维度。与此同时，实验实例以合成分布为主，尚未在物流配载、广告预算、投资组合等真实业务数据及其附加约束（如体积、风险、分组与互斥等）上进行端到端验证。

（4）统计推断与机制归因仍需加强：本文已采用配对检验与置信区间支持差异性结论，但在多重比较控制、效应量报告以及机制层面的可解释性验证方面仍不充分。后续研究可结合更严格的统计协议与可解释分析，以进一步提升结论的可推广性与可审计性。

6.3 未来研究方向

面向“可推广、可落地、可解释”的研究目标，后续工作可从算法、实验与应用三条主线推进：

6.3.1 算法改进与融合设计

（1）补强劣势算法的鲁棒性机制：

① 针对二进制群智能算法，重点引入“记忆存档 + 多样性维护 + 自适应参数”的组合机制，并重构可行性处理（如分层修复/软惩罚与修复混合），以降低参数敏感性与性能方差。

② 针对 GreedyDensity，引入可计算的“陷阱结构指示器”（如近容量物品占比、密度分布极值差、容量比  $\alpha$  等），触发式切换到 Greedy+Max 或上界引导搜索（有界集束搜索），从而扩大适配范围。

（2）构建自适应混合算法（两阶段或闭环迭代）：

① “Greedy+Max 生成高质量初解 → 有界集束搜索在受控束宽下精炼”的两阶段框架，用于在

严格时间预算内提升对抗场景解质量。

② “记忆模拟退火与遗传算法的协同”：以 SA 进行局部强化、以 GA 维持多样性与跨域跳跃，并通过预算分配策略控制总时间与内存。

（3）轻量化与工程优化：在不改变理论属性的前提下，对增强分支限界法的节点表示、上界复用与优先队列维护进行工程优化，以降低峰值内存；同时简化有界集束搜索的上界计算与候选管理，形成“可解释的近似剪枝”版本，在可控的质量损失内换取显著的运行时间下降。

### 6.3.2 实验体系拓展与深化

（1）超大规模与动态场景评测：构建覆盖  $N=1000-5000$  乃至更大规模的实例库，并制定动态背包评测协议（如物品到达/撤销与容量变化）。在此基础上，系统评估各算法的在线更新能力、实时性与鲁棒性。

（2）系统化参数寻优与敏感性建模：采用贝叶斯优化或多臂老虎机等方法，对束宽、降温策略、迭代预算与种群规模等关键超参数开展结构条件化的自动调参。同时，建立“结构特征  $\rightarrow$  参数配置  $\rightarrow$  性能预测”的映射模型，以刻画参数敏感性并支撑可解释的算法选型。

（3）更完备的统计推断与多维指标体系：补充效应量报告与多重比较控制，扩展评价维度至多目标性能、扰动鲁棒性与跨平台可移植性。此外，通过统一的数据落盘规范与审计链路设计，保证实验过程与结果可追溯、可复核。

### 6.3.3 潜在的真实场景验证与应用推广

（1）真实业务数据与复合约束验证：收集并构建包含体积、风险、分组与互斥等复合约束的行业数据集；在可控对照实验条件下，量化本文所推荐策略在收益提升与风险暴露方面的综合效应。

（2）自动化选型与配置系统落地：将“结构特征提取—算法与参数推荐—预算约束下的调度执行”整合为可部署的软件模块，形成工程可复用的算法选择器，以支持不同场景下的快速迁移与稳定复现。

（3）跨领域迁移与通用性检验：将本文的评估框架与混合策略推广至任务调度、资源分配与路径规划等组合优化问题，并在不同约束结构与问题几何条件下检验方法的通用性与可解释性。

## 6.4 本章小结

本文围绕 0/1 背包问题，建立了统一实例生成与可复现评测的实验链路，并在多类典型结构实例与多维指标体系下，对 6 类共 12 种算法进行了系统对比。结果表明：在给定资源预算与实例结构条件下，不存在“全局最优”的单一算法，算法优势由问题结构（如近容量陷阱、相关性、重尾性）与计算预算共同决定。

综合证据支持的“优选算法矩阵”为：Greedy+Max 在常规与大规模时间敏感场景下提供最稳健的效率—稳定性折中；有界集束搜索在对抗性陷阱结构下通过上界引导显著改善解质量；记忆模拟退火在复杂结构实例上体现更优的近似质量与稳定性；增强分支限界法在小规模且高精度需求下可获得精确最优解，但需承担更高的时间与内存开销。相对而言，二进制群智能算法与纯密度贪心在本文默认参数与修复策略下表现出较高 Gap 与较大方差，显示其机制鲁棒性与参数适配仍需系统改造与优化。

本文同时明确了结论适用边界：主实验以合成实例为主，规模区间与参数寻优仍有限，且尚未覆盖动态场景与真实业务复合约束；因此，本文提出的适配框架应被视为“可复现的经验性选型规则”，需要在目标业务数据、硬件平台与预算约束下进一步校准。未来工作将围绕（i）混合算法与轻量化实现，

（ii）超大规模与动态背包评测，（iii）系统调参与多维指标扩展，以及（iv）真实数据验证与自动化选型

系统构建，推动算法结论从实验室对比走向可部署的工程决策支持。

参 考 文 献

[1] Lowerre, B. T. (1976). The Harpy Speech Recognition System. PhD thesis, Carnegie Mellon University.

[2] Nemhauser, G. L., Wolsey, L. A., & Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1), 265-294.

[3] Zhang, W. (1998). Complete anytime beam search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 425-430.

[4] Krause, A., & Golovin, D. (2014). Submodular function maximization. *Tractability: Practical Approaches to Hard Problems*, 3, 71-104.

[5] Tukan, M., Mualet, L., & Feldman, M. (2024). Practical 0.385-Approximation for Submodular Maximization Subject to a Cardinality Constraint. *Advances in Neural Information Processing Systems (NeurIPS)*, 37.

[6] Buchbinder, N., & Feldman, M. (2024). Discretely Beyond 1/e: Guided Combinatorial Algorithms for Submodular Maximization. *Advances in Neural Information Processing Systems (NeurIPS)*.

[7] Land, A. H. & Doig, A. G. (1960). An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, 28, pp. 497-520.

[8] Mans B, Roucairol C. Performances of parallel branch and bound algorithms with best-first search[J]. *Discrete Applied Mathematics*, 1996, 66(1): 57-74.

[9] Morrison D R, Jacobson S H, Sauppe J J, et al. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning[J]. *Discrete Optimization*, 2016, 19: 79-102.

[10] Forget N, Parragh S N. Enhancing branch-and-bound for multiobjective 0-1 programming[J]. *INFORMS Journal on Computing*, 2024, 36(1): 285-304.

[11] Dantzig G B. Discrete-variable extremum problems[J]. *Operations research*, 1957, 5(2): 266-288.

[12] Kellerer, H., Pferschy, U., & Pisinger, D. (2004). *Knapsack Problems*. Springer.

[13] Cormen T H, Leiserson C E, Rivest R L, et al. Introduction to Algorithms (3-rd edition)[J]. MIT Press and McGraw-Hill, 2009.

[14] Vazirani V V. Approximation algorithms[M]. Berlin: springer, 2001.

[15] Yaroslavtsev G, Zhou S, Avdiukhin D. “ bring your own greedy ” + max: near-optimal 1/2-approximations for submodular knapsack[C]//International Conference on Artificial Intelligence and Statistics. PMLR, 2020: 3263-3274.

[16] Montazeri M, Kiani R, Rastkhadiv S S. A new approach to the restart genetic algorithm to solve zero-one knapsack problem[C]//2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI). IEEE, 2017: 0050-0053.

[17] Sampson J R. Adaptation in natural and artificial systems (John H. Holland)[J]. 1976.

附录X.

● 作者贡献

作者	贡献
毛子鋆	近似算法-贪心算法、问题定义
刘伦	经典算法-经典集束搜索、有界集束搜索
杨昊哲	二进制粒子群优化、二进制布谷鸟搜索
张涛	模拟退火和记忆模拟退火
梁天宇	遗传算法和改进遗传算法
邓康	分支限界算法和改进分支限界算法