**大语言模型驱动的个性化骑行路线规划：一种反思型进化算法框架**[[1]](#footnote-0)

**摘要：**针对个性化骑行路线规划中现有方法难以权衡平行路径且低估节点价值的问题，本文提出带节点收益的弧定向模型（AONP），通过显式建模平行路径与同步优化节点收益，更真实刻画骑行体验。为求解AONP，设计LLM强化反思型进化算法（LrEA），将大语言模型嵌入进化框架，通过经验树反思与元进化算子实现启发式策略的自动化迭代。实验表明，AONP在路径吸引力与时间效率权衡上优于传统模型，LrEA生成的求解器在基准测试中性能超越传统元启发式算法及先进LLM增强方法，验证了模型与算法的有效性。

**关键词：** 骑行线路；弧定向；大语言模型；进化算法

1 引言

骑行作为融合绿色出行、体育健身与休闲体验的运动方式，深度契合“健康中国”与“碳达峰碳中和”等国家战略。2024年行业数据显示，骑行类App累计用户已突破3亿，带动关联产业产值超3000亿元[1]。然而，作为核心体验环节的路径规划却明显滞后：主流工具（如高德地图、行者App）仍聚焦于最短路径或固定路线推荐，难以满足骑行场景对景观、安全性及体能分配的多维个性化需求，成为制约用户体验升级与产业价值释放的关键瓶颈。

针对这一挑战，学界近年来日益关注骑行路径的智能优化问题。Souffriau等[2]率先将骑行路径规划建模为弧定向问题（Arc Orienteering Problem, AOP），通过在路网弧段赋值收益，在资源约束下最大化总收益。这一建模方式契合骑行体验源于路段本身的特性，为风景、坡度、安全等路段属性的优化提供了理论基础。后续研究沿两个维度深化：一是模型结构优化，如Verbeeck等[3]引入起点灵活选择与顶点重复访问机制以提升解的实用性，Kaur等[4]则构建了融合通行成本与可导航性评分的路径评价体系；二是目标体系拓展，多目标协同优化成为主流趋势，如Tricoire等[5]权衡景观吸引力与碳排放，Dhein等[6]兼顾总收益与服务覆盖范围，Chen等[7]则在距离与景观质量间实现动态权衡。此外，环境动态性也得到重视：Chen等[8]将弧段属性建模为时变函数，Piedra-de-la-Cuadra与Ortega[9]则引入阶跃函数刻画吸引力变化并允许节点等待策略。

尽管上述研究取得显著进展，但两个核心局限仍制约其在真实场景中的适用性：（1）平行路径权衡缺失。现实路网中，同一节点对间常存在多条属性迥异的平行路径（如高效但单调的城市干道与耗时但风景优美的滨水绿道），为骑行者提供时间－体验权衡空间。然而，现有模型通常将节点间连接简化为单一弧段，难以支持个性化决策；（2）节点价值低估。骑行体验具有“行中有停、停中有赏”特性，观景平台、文化地标、补给站等节点本身具有重要吸引力，但当前研究普遍“重弧轻点”，将节点视为路径附属结构，忽略了其对骑行满意度的独立贡献，导致难以推荐包含高价值节点停靠的实用路径。

针对上述挑战，本文提出带节点收益的弧定向模型（Arc Orienteering Problem with Node Profits, AONP），通过两项核心创新构建更真实的骑行规划模型：其一，显式建模平行路径，保留同一节点对间多条弧段在时间、景观等维度的属性差异，支持基于偏好的精细化权衡；其二，同步优化弧段收益与节点收益，将高价值节点纳入目标函数，准确刻画“行停结合”的骑行行为特征。

在求解方法上，传统AOP研究依赖迭代局部搜索[3]、遗传算法[6]、模因算法[7,8]等元启发式方法，或分支定价[5]、分支定界[10]等精确算法。然而，这些方法高度依赖专家经验进行算子设计与参数调优，自动化程度低，面对AONP这类结构复杂的新问题时迁移性受限。近年兴起的大语言模型（LLM）为组合优化提供了新范式，其强大的语义理解与代码生成能力可自动设计优化算子[11]，但面临幻觉输出、提示敏感及复杂约束推理脆弱等挑战[12,13]。

为此，本文设计LLM强化反思型进化算法（**L**LM-Enhanced **R**eflective **E**volutionary **A**lgorithm, LrEA），创新性地将大语言模型嵌入进化框架。与传统方法不同，LrEA通过LLM的模式识别与语境推理能力动态生成优化算子，并引入经验树反思机制实现策略的自我迭代与优化。该人机协同范式不仅摆脱对专家知识的过度依赖，更有可能发现超越人类直觉的新型启发式策略，为复杂组合优化问题的自动化求解提供新思路。

2 问题描述及数学模型

**2.1 问题描述**

本问题定义在一个有向多重图上。节点集合包括唯一的起点（节点 0）、唯一的终点（节点 ），以及中间兴趣点集合。每个兴趣点关联两个非负属性：停留时间和吸引力分数，分别表示在该点停留所需的时间消耗和所获得的体验价值。

弧集合允许任意有序节点对之间存在多条平行弧，以刻画现实路网中同一OD对间多路径并存的情形。记到的平行弧集合为，其中表示从节点到的平行弧数量。每条关联两个非负的属性：旅行时间表示通过该弧段所需的时间成本，吸引力分数则综合反映该弧段的体验收益。

给定一个总时间预算，问题的目标是寻找一条从起点 0 出发、抵达终点的路径，使得在总耗时（包括所有经过弧段的旅行时间与访问节点的停留时间之和）不超过 *T* 的前提下，最大化路径总吸引力。

**2.2 数学模型**

为形式化上述问题，引入以下符号：

**集合**

·，

·，其中为从到的平行弧集合

**参数**

·：兴趣点的吸引力分数

·：兴趣点的停留时间

·：弧的吸引力分数

·：弧的旅行时间

·：总预算时间

·：一个足够大的正数

**决策变量**

·：若路径包括，则为1，否则为0

·：若路径访问兴趣点，则为1，否则为0

·：到达节点的时间

**目标函数**

|  |  |
| --- | --- |
|  | (1) |

**约束条件**

·起点出度为1（必须从起点出发）：

|  |  |
| --- | --- |
|  | (2) |

·终点入度为1（必须到达终点）：

|  |  |
| --- | --- |
|  | (3) |

·兴趣点最多访问一次（入度）：

|  |  |
| --- | --- |
| *j* | (4) |

·流守恒（对于每个兴趣点，入度=出度）：

|  |  |
| --- | --- |
| *j* | (5) |

·时间预算约束：

|  |  |
| --- | --- |
|  | (6) |

·时间顺序与访问一致性：

|  |  |
| --- | --- |
| *, ,* | (7) |

·节点访问与弧选择的逻辑关系：

|  |  |
| --- | --- |
| *j* | (8) |

·变量类型：

|  |  |
| --- | --- |
|  | (9) |
|  | (10) |
|  | (11) |

3 求解算法

LrEA的核心设计理念是突破传统进化算法“直接优化问题候选解”的固有范式，转向“优化生成解的启发式算子”，通过深度融合LLM的语义理解与代码生成能力，实现算子的自动化设计、迭代与优化。

基于这一理念，LrEA构建了元进化架构，将可执行的启发式算子作为进化个体；同时引入了进化知识反思框架，通过经验树系统存储跨迭代的成败样本，借助LLM的语义分析能力提炼跨代成败经验，为算子生成提供指导；并设计了集成四类语义引导元算子的两阶段元进化算子框架，将算子生成拆解为规划制定与代码实现，结合反思经验明确优化方向。

在初始化阶段，LrEA构建简单进化框架作为种子源与评估平台，生成基于专家知识的初始种群。随后进入迭代进化阶段，每轮迭代中首先激活知识反思机制：从经验树中检索对应算子的历史数据，通过LLM分析成功与失败案例，生成策略修正指令。接着，算法结合该指令运用四类元进化算子（每类聚焦不同优化维度）生成子代个体。新个体随后被模块化嵌入评估框架，在标准测试集上量化性能指标完成评估。完成评估后，子代种群的个体关键信息（可执行代码、评估结果、错误诊断信息等）将分类存入经验树，随后子代种群直接替代原当代种群进入下一轮迭代，直至达到预设最大个体生成数。

在应用阶段，LrEA作为离线算子生成器，通过独立运行分别输出初始解构造、交叉、变异、局部搜索与弧选择五类优质算子，这些算子被整合至统一进化框架，最终构建出针对 AONP问题的高效求解器。

**3.1 简单进化框架**

LrEA的核心设计理念是将进化算法的优化对象从问题解本身转向生成这些解的“算法组件”——即可执行的启发式算子。为支撑这一元进化过程，LrEA构建了一个简单而完整的进化框架作为底层求解环境。该框架并非LrEA的元进化架构本身，而是作为“种子源”与“评估平台”发挥基础性作用：一方面，它为初始种群提供包含人类专家知识的种子算子；另一方面，它为新算子的模块化集成与性能测试提供标准化接口。

在结构组成上，该框架由五个功能互补的组件构成。其中，种群初始化通过随机抽取20%的节点生成满足起止点约束的路径；交叉算子实施单点交叉策略以组合父代片段；变异算子随机交换路径内部两个节点的访问顺序；局部搜索算子集成了节点插入、删除、交换和替换等邻域操作，并采用首次改进策略对当前解进行精细优化。此外，针对AONP模型中平行路径的特性，框架特别设计了弧选择算子。该算子依据单位时间收益比（即高低吸引力弧段间的收益增量与时间增量之比）在每对相邻节点间决策具体的连接弧：若路径超时，优先将比值低的高吸引力弧段降级以节省时间；若时间充裕，则尝试将低吸引力路段升级以增加得分。

上述组件不仅构建了一个基础求解器，在LrEA的元进化流程中更承担着关键的双重功能。首先，作为进化的起点，框架中包含的基础算子构成了高质量的种子算子集合。这些经过验证的启发式策略为大语言模型提供了可靠的参考范本，使其能够基于人类专家知识生成逻辑相似但效能更优的改进版本。其次，该框架充当了标准化的评估平台，支持新生成算子的无缝集成与性能验证。任意由语义进化算子生成的新个体均可替换框架中的原始对应组件，形成完整的算法实例。系统随后在预定义的基准测试集上执行这些实例，并量化平均目标函数值等关键性能指标。这种机制不仅确保了算子改进的有效性验证，更为进化知识反思框架提供了丰富的性能数据，支撑算法在迭代中持续学习与优化。

**3.2 个体表示**

LrEA将每个个体定义为一段可执行的启发式算子代码，而非传统意义上的问题解。在具体表示上，每个个体由严格规范的接口层与灵活多变的实现层构成。接口层定义函数签名，包括名称、输入参数与返回类型，确保组件兼容性；实现层包含具体操作逻辑，完全由LLM自主生成。例如，针对变异操作，个体可具象化为一个名为mutation\_v1的函数。该函数严格遵循“输入路径列表、输出新路径”的接口规范，而在函数体内部，则可自由实现“随机交换路径中两个非首尾节点”等具体逻辑。这种设计在不预设编码模板的前提下，既保证了组件的标准化集成，又赋予了LLM充分的语义生成自由度与探索空间。

基于这种代码级的个体表示，LrEA建立了一种即插即用的评估机制。在3.1节构建的简单进化框架基础上，任何新生成的代码个体均可作为模块直接替换框架中的原始对应算子（如用新生成的变体代码替换原有的交叉操作）。这种替换过程无需修改框架的其他部分，从而快速形成一个新的算法实例。随后，系统将该实例在预定义的基准测试集上运行，并记录其在多个算例上的平均目标函数值作为个体的适应度评分。

**3.3 初始种群**

在LrEA框架中，初始种群以第3.1节所述简单进化框架中的五个基础算子作为种子，通过结构化指令引导LLM对这些经过验证的启发式策略进行定向改进与创新，实现从结构化模板向多样化候选个体的演化。这种设计确保进化过程从具有实际求解能力的起点开始，而非在无效解空间中盲目探索。

初始种群的生成以预定义种子个体为基础，通过结构化指令（图 1）引导 LLM 对种子进行定向修改与创新迭代，该过程循环执行，直至种群规模达到预设阈值。每条指令由外层提示词和内层提示词两部分构成，形成完整的语义引导体系。外层提示词采用角色扮演机制，赋予大语言模型以知名算法设计专家的身份（如“你是启发式算法领域的先驱，Dr. Alexander Thompson”），以增强其专业性与创造性输出能力。

内层提示则包含四个关键元素：原始种子算子代码（{seed}）提供可执行基础；函数命名规范（{func\_name}）确保命名一致性；功能增强要求（{func\_desc}）指明改进方向；问题背景描述（{problem\_desc}）界定AONP特有的约束条件与优化目标。例如，针对变异算子的提示中，{seed}包含基础交换逻辑，{func\_desc}可能指定“增强路径多样性同时维持时间约束可行性”，而{problem\_desc}则强调“路径必须从起点到终点且总时间不超过预算”。这种结构化指令使LLM在保持接口兼容性的前提下，生成语义等效但性能更优的新版本。

|  |
| --- |
|  |
| **图1. 初始种群指令模板** |

**3. 4进化知识反思机制**

为突破传统进化算法的历史知识利用低效（如未进入下一代的个体中蕴含的潜在有效策略被直接丢弃）、搜索策略固化（如缺乏根据问题特性动态调整的自适应能力）及失败经验浪费（如未深入挖掘非可行解或低性能个体背后的算子设计缺陷）等瓶颈，LrEA设计进化知识反思机制，通过构建具备长期记忆能力的经验树，系统萃取跨代的成败经验，进而驱动语义进化算子的自我迭代与策略自适应。

进化知识反思机制的核心是结构化经验树，采用双分区存储设计以实现知识的有效组织与检索。经验树由“成功样本区”与“失败样本区”两个互补分支构成，每个分区均按生成算子类型（对应3.5节定义的四类元进化算子）建立多级索引。在成功样本区，系统记录每个成功个体的可执行代码及其目标函数评估值。在失败样本区，除记录可执行代码外，还捕获运行时错误信息，并依据错误类型进行分类存储。

进化知识反思机制的运作遵循样本提取、语义分析和指令生成的流程。具体而言，当执行特定类型语义进化算子时，系统首先从经验树中检索该算子对应的历史数据：性能最优的两个成功个体及最高频错误类别中的一个代表性失败案例。随后，这些样本信息被嵌入预设提示词模板（图2），输入LLM进行分析。LLM通过对比成功与失败案例，识别该算子生成高质量解的关键特征及常见缺陷根源，进而输出针对该算子的优化焦点与错误规避规则，形成结构化修正指令。最终，该指令被整合至语义进化算子生成流程中，确保新生算子在继承成功经验的同时有效规避高频错误。

|  |
| --- |
| /Users/keqi/Pictures/反思.svg反思 |
| **图2. 反思操作指令模板** |

**3.** **5元进化算子框架**

现有基于LLM的进化方法普遍采用一次性开放式提示，例如“生成与之前完全不同的个体”。这类提示语义宽泛、缺乏结构约束，导致 LLM 需在单次推理中同时完成“想策略”与“写代码”双重任务。LrEA 提出元进化算子框架，摒弃了一步生成的常用模式，将算子生成任务分解为“规划制定”与“代码实现”两个阶段。更重要的是，该框架与3.4节所述的进化知识反思框架融合，充分利用经验树中萃取的跨代历史经验，引导LLM生成高质量的子代个体。

**3. 5. 1元进化算子类型与语义设计**

元进化算子框架包含四个分别对应不同进化策略方向的元算子，旨在通过结构化语义引导LLM实现算子的自我迭代。

1. **结构重构元算子。**该算子的优化目标为“探索未被覆盖的算法搜索空间，通过激进式的结构变革发现全新的优化策略”。算法选取种群中性能最优的3个个体作为参考，结合进化知识反思框架提供的成功经验指引，引导LLM进行大幅度代码重写。例如，若参考算子主要依赖“随机交换两个节点”的逻辑，该算子可能指导LLM将其重构为“基于邻域结构的块移动”逻辑，从而从根本上改变搜索模式，而非仅微调细节。
2. **机制融合元算子。**该算子的优化目标为“整合不同层级个体的优势特征，平衡探索与开发”。算法分别采样最优、中位及末位个体作为参考，提示LLM融合各层级代码以形成一个全新代码。其中，最优个体提供高性能的策略基线，中位个体代表稳健的通用解法，而末位个体虽然性能较差，但可能蕴含未被发现的创新性机制或特殊边界处理逻辑。例如，将精英个体的“贪心选择”与末位个体的“随机扰动”相结合，生成兼具高收敛性与强跳出能力的混合算子。
3. **参数调优元算子。**该算子的优化目标为“在逻辑结构固定前提下，优化控制算法行为的数值参数”。该算子锁定当代种群中的最优个体为参考个体，严格限制LLM仅调整阈值、权重等数值参数，确保性能提升纯粹源于参数空间的局部寻优。此处的“数值参数”指的是算子中的超参数，而非路径节点坐标。例如，在变异算子中，数值参数可以是变异操作的执行概率（如0.1至0.5）或发生作用的节点范围（如前50%节点）。
4. **自适应增强元算子。**该算子的优化目标为“为个体代码逻辑增加自适应机制，实现根据求解过程状态动态调整策略”。算法随机选取一个个体为参考，引导LLM对其代码逻辑添加一套自适应控制流，使得算子能够根据解质量或迭代次数等即时反馈进行不同策略的切换。例如，在优化交叉算子时，可引入逻辑：“如果两个父代个体的相似度超过90%，则采用多点交叉以增加多样性；否则，采用单点交叉以保留优良模式”。

**3. 5. 2规划制定**

规划制定阶段作为连接进化知识反思框架与具体代码实现的知识转化枢纽，其核心在于将模糊的进化意图转化为精确的执行规划，避免直接向LLM下达模糊的生成指令。在该阶段。针对每类语义进化算子，系统整合三类关键输入：算子特定的优化目标（{operation\_goal}）、从当前种群选取的参考个体代码（{reference\_codes}），以及由3.4节反思机制生成的策略修正指令（{reflection\_lessons}）。

这些多源信息被嵌入预设的规划指令模板（图3）中，引导LLM进行深度语义分析与策略综合。例如，对于结构重构算子，结合反思指令指出的“避免深层循环导致超时”的教训，生成的规划可能包含：“重构节点访问顺序决策逻辑，引入基于景观质量的自适应权重机制，避免超过三层嵌套循环”等具体指导。LLM基于此综合信息，凝练生成一份结构化的生成规划，明确指定新个体应具备的核心特征、需规避的缺陷模式及关键实现细节。

|  |
| --- |
| 规划 |
| **图****3. 规划制定指令模板** |

**3. 5. 3代码实现**

在代码实现阶段，四种元算子统一采用结构化的代码生成模板（见图4），将规划制定阶段产生的策略规划转化为符合特定函数签名规范的可执行代码。该模板包含四个组成部分：{seed}部分提供基础函数框架；{construction\_idea}位置用于填入规划阶段生成的详细指导思路，从而将具体的改进意图直接传递给LLM；{reference\_codes}部分则复用与规划阶段相同的参考个体代码，为LLM提供具体的代码修改上下文和风格基准；{function\_signature}部分明确规定了生成代码必须遵循的接口规范，确保新个体能够无缝集成至评估系统中。

|  |
| --- |
| 生成 |
| **图4. 代码实现指令模板** |

**3. 6种群更新**

在算法初始化阶段，通过3.3节的方法生成初始种群，该种群作为当前种群启动后续的LrEA迭代进化流程。在每一轮迭代中，个体进化阶段综合运用3.4节定义的四种元进化算子，每类算子各生成个子代个体，共同构成规模为的子代种群。这些子代个体完成性能评估后，该子代种群将直接替换当前种群，进入下一次迭代循环。当算法运行过程中生成的个体总数达到预设的时，迭代终止。

4 求解算法

LrEA算法采用Python语言实现，所有测试均在搭载Apple M4处理器（10核CPU+10核GPU）及24GB统一内存的macOS Sequoia 15.5系统环境下完成。算法核心配置包括：选用DeepSeek-V3-0324大语言模型（默认参数），种群规模设定为20；每轮迭代中，四种元进化算子分别生成5个子代个体，当累计生成的启发式算子个体总数达到160时，算法终止运行。为全面评估算法性能，实验构建了包含48个实例的综合测试集，涵盖20个小规模实例（S集）、23个大规模实例（L集）以及5个专为评估LrEA不同进化阶段性能设计的评估集（E集）。所有实例均采用“A\_B”格式命名（如S\_10表示节点数为10的小规模实例），数据集已公开发布于：https://github.com/Go-own-way/AOP-NP-instances。

**4. 1 核心组件贡献的消融实验**

为量化LrEA算法框架中各核心组件对性能的贡献，本节通过移除关键模块开展消融实验。构建了变体LrEA\O1至O4（分别移除结构重构、机制融合、参数调优、自适应增强元算子）、LrEA\R（移除反思机制）及LrEA\P（移除规划制定）。各变体在评估集上独立运行十次，实验结果如表1所示（Avg、Best、Std、CPU分别为平均目标函数值、最佳目标函数值、标准差、平均运行时间），其中“↑”表示指标越高越好，“↓”表示越低越好，括号内为相对LrEA的变化率。

**表1. LrEA及其消融变体的性能对比**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 消融变体 | 移除组件 | Avg ↑ | Best ↑ | Std ↓ | CPU/h ↓ |
| LrEA（完整） | 无 | 5651.06 | 5695.45 | 19.13 | 2.69 |
| LrEA\O1 | 结构重构 | 5633.97  (-0.30%) | 5668.13  (-0.48%) | 29.78  (+55.70%) | 2.62  (-2.55%) |
| LrEA\O2 | 机制融合 | 5633.77  (-0.27%) | 5669.64  (-0.50%) | 21.05  (+10.03%) | 2.16  (-19.78%) |
| LrEA\O3 | 参数调优 | 5633.77  (-0.31%) | 5669.64  (-0.45%) | 19.89  (+3.99%) | 2.52  (-6.36%) |
| LrEA\O4 | 自适应增强 | 5641.66  (-0.17%) | 5672.71  (-0.40%) | 20.05  (+4.81%) | 2.42  (-10.12%) |
| LrEA\R | 反思机制 | 5627.02  (-0.43%) | 5686.43  (-0.16%) | 27.67  (+44.64%) | 2.67  (-0.66%) |
| LrEA\P | 规划制定 | 5632.97  (-0.32%) | 5669.77  (-0.45%) | 21.41  (+11.96%) | 2.35  (-12.39%) |

实验结果揭示了以下主要发现：（1）剔除任一模块后，Avg与Best指标呈现系统性下降，同时Std指标普遍上升，表明各模块对维持算法整体性能均具有实质性贡献。（2）反思机制是性价比最高的增益杠杆，以最少的时间增幅，获得最大的Avg增幅和排名第二的标准差降幅，凸显了跨代经验学习的显著价值。（3）规划制定模块带来了各项指标的显著增益（均排名前三），证明了其将模糊的进化意图转化为精确执行规划的能力。（4）四种元进化算子表现出了差异性。例如，结构重构元算子的移除带来了最大的Std增幅；机制融合元算子的移除导致了最大的Best降幅；而参数调优、自适应增强等精细化元算子的剔除，则产生了相对于其他算子更为温和的性能衰退。

值得强调的是，上述各模块引入的时间开销仅作用于离线算子生成阶段，属于一次性预处理成本。LrMA 作为任务是自动化产出可复用的优质启发式算子，而非直接参与实时路径规划。一旦算子生成完毕，最终固化的求解器可在大量问题实例求解中重复调用，其一次性计算投入将被显著分摊。从实际应用价值来看，这些模块带来的性能提升，在路径规划场景中可能转化为更优风景路段，由此带来的用户体验增益，远超过算子生成阶段的额外耗时。

**4. 2 跨LLM后端的鲁棒性分析**

为评估LrEA框架对后端LLM的依赖程度，并区分框架自身贡献与模型固有能力的边界，本节选取四个在公开代码生成基准测试中表现相近的开源LLM作为后端：DeepSeek-V3-0324（默认配置）、Doubao-seed-1-6-flash-250828、Moonshot-Kimi-K2-Instruct及Qwen-flash-2025-07-28。每个模型驱动的LrEA变体均在完整测试集上独立运行10次，以总吸引力得分为评价指标。

|  |
| --- |
| llm 泛化 |
| **图5. 不同后端LLM下LrEA的收敛曲线** |

图5展示了不同后端LLM下LrEA的收敛曲线，横轴表示评估次数（即生成算子的总数），纵轴为目标函数值。所有曲线均表现出快速上升并迅速收敛至[5656, 5670]区间的特征，表明LrEA在不同后端LLM上的优化过程具有高度一致性。尽管最终性能存在细微差异，但其收敛趋势和稳定性保持一致，体现了框架在不同LLM支持下的稳健行为。值得注意的是，各模型变体的最终性能排序与这些模型在公开代码生成基准测试中的表现吻合：DeepSeek-V3-0324表现最优，其次为Qwen-flash，Doubao-seed与Moonshot-Kimi-K2-Instruct相对略低。这一对应关系说明，虽然LrEA的性能上限仍由后端LLM的固有能力决定，但其核心优化机制能够有效利用不同模型的能力，在保证算法稳定性的前提下实现性能最大化。

上述结果为LrEA具备后端无关性提供了经验证据：无论选用何种后端模型，均可观察到相似的优化路径与收敛特性。这一特性赋予框架“即插即用”的自适应能力——在快速原型开发阶段，可选择性能达标且易于获取的轻量级模型；当追求最优求解质量时，则可通过更换高性能模型实现进一步提升，且无需修改框架核心逻辑。

**4. 4 AONP算法基准测试**

为验证LrEA生成算法在求解AONP问题中的性能，本研究将LrEA生成的算子集成至3.1节所述的简单进化框架中，替换手工设计算子，构建增强型算法并命名为LrEA-EA。实验选取四种方法作为基准开展对比测试：以Gurobi作为小规模算例的精确求解器；Simple-EA（3.1节提出的简单进化算法）；以及当前最优的LLM增强启发式算法自动生成框架EoH[14]和ReEvo[15]生成的EoH-EA、ReEvo-EA。各算法在每个算例独立运行10次，结果汇总于表2（小规模算例）与表3（大规模算例），报告最优值（Best）、平均值（Avg）、标准差（Std）及计算耗时（CPU/s），并通过成对统计对比（胜场数#Wins、平局数#Ties、负场数#Losses）和Wilcoxon符号秩检验（P值）分析显著性。

以Gurobi求解的精确解为基准，LrEA-EA的平均求解结果相对前者偏差仅0.58%，表明其在小规模问题上具备逼近理论最优的能力。在启发式算法横向对比中，LrEA-EA展现出解质量与稳定性的双重优势：相较于Simple-EA，其平均解质量提升0.42%，标准差由2.6显著收敛至0.4，证实搜索过程鲁棒性增强；与EoH-EA相比，平均解质量提升0.23%，且Wilcoxon检验P值远小于0.05，性能优势具有统计显著性；与表现最优的ReEvo-EA相当，但在成对比较中仍以微弱优势体现特定算例的求解潜力。

大规模算例测试进一步验证了LrEA-EA在求解高维复杂问题时的优越性。在解质量方面，LrEA-EA的平均求解结果（Avg）达10127.2，相较于Simple-EA和EoH-EA分别提升2.18%和6.78%（EoH-EA在节点规模较小时表现良好，但规模增大时求解质量急剧下滑）；与ReEvo-EA相比，Avg提升0.53%，Wilcoxon检验P=0.03，且成对比较中LrEA-EA以16胜7负的优势，证实大规模搜索空间中更强的寻优能力。在稳定性层面，LrEA-EA的标准差仅117.3，较EoH-EA和ReEvo-EA分别降低10%和29%，收敛一致性最优。计算效率方面，LrEA-EA与ReEvo-EA耗时相当，表明其在维持可接受计算成本的前提下，通过改进启发式算子设计实现了求解精度与鲁棒性的双重突破。

综上，LrEA的生成算法在求解AONP问题时，无论是小规模场景下的精确逼近能力，还是大规模场景下的稳定寻优表现，均展现出持续且稳定的竞争力。

**表2. 小规模算例下的性能比较**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 算例 | Gurobi | | Simple-EA | | | | EoH-EA | | | | ReEvo-EA | | | | LrMA-EA | | | |
| opt | CPU/s | Best | Avg | Std | CPU/s | Best | Avg | Std | CPU/s | Best | Avg | Std | CPU/s | Best | Avg | Std | CPU/s |
| S\_32 | 954.0 | 5.9 | 946.0 | 945.0 | 1.1 | 0.3 | 945.0 | 945.0 | 0.0 | 4.5 | 945.0 | 945.0 | 0.0 | 19.9 | 945.0 | 945.0 | 0.0 | 20.0 |
| S\_33 | 1060.0 | 4.4 | 1059.0 | 1056.3 | 2.3 | 0.3 | 1056.0 | 1055.7 | 0.5 | 7.5 | 1056.0 | 1056.0 | 0.0 | 25.7 | 1056.0 | 1056.0 | 0.0 | 25.1 |
| S\_34 | 1266.0 | 0.7 | 1257.0 | 1253.6 | 2.7 | 0.3 | 1255.0 | 1254.6 | 0.5 | 7.8 | 1255.0 | 1255.0 | 0.0 | 24.5 | 1255.0 | 1255.0 | 0.0 | 23.2 |
| S\_36 | 1165.0 | 8.4 | 1158.0 | 1156.1 | 1.4 | 0.3 | 1161.0 | 1160.8 | 0.4 | 7.8 | 1161.0 | 1161.0 | 0.0 | 23.8 | 1161.0 | 1161.0 | 0.0 | 23.7 |
| S\_37 | 1365.0 | 2.8 | 1360.0 | 1357.7 | 1.8 | 0.3 | 1361.0 | 1360.2 | 0.4 | 9.0 | 1361.0 | 1361.0 | 0.0 | 35.0 | 1361.0 | 1361.0 | 0.0 | 31.7 |
| S\_38 | 1079.0 | 2.0 | 1071.0 | 1067.3 | 2.1 | 0.4 | 1070.0 | 1069.8 | 0.4 | 10.4 | 1070.0 | 1070.0 | 0.0 | 35.7 | 1070.0 | 1070.0 | 0.0 | 34.8 |
| S\_39 | 677.0 | 7.3 | 673.0 | 672.5 | 0.5 | 0.3 | 673.0 | 673.0 | 0.0 | 6.9 | 673.0 | 673.0 | 0.0 | 39.0 | 673.0 | 673.0 | 0.0 | 39.4 |
| S\_44 | 1282.0 | 3.0 | 1281.0 | 1276.3 | 2.8 | 0.4 | 1278.0 | 1276.6 | 1.1 | 14.0 | 1279.0 | 1278.1 | 0.3 | 38.9 | 1278.0 | 1278.0 | 0.0 | 41.0 |
| S\_45 | 1389.0 | 1.6 | 1380.0 | 1377.8 | 1.8 | 0.5 | 1384.0 | 1381.6 | 1.5 | 15.3 | 1384.0 | 1383.9 | 0.3 | 44.3 | 1384.0 | 1384.0 | 0.0 | 44.8 |
| S\_46 | 1692.0 | 0.8 | 1678.0 | 1675.8 | 1.4 | 0.6 | 1683.0 | 1681.4 | 1.0 | 14.1 | 1684.0 | 1684.0 | 0.0 | 37.4 | 1684.0 | 1684.0 | 0.0 | 35.1 |
| S\_48 | 1502.0 | 1.4 | 1494.0 | 1490.4 | 2.9 | 0.5 | 1494.0 | 1491.7 | 1.6 | 19.1 | 1494.0 | 1493.9 | 0.3 | 59.9 | 1494.0 | 1493.9 | 0.3 | 53.1 |
| S\_53 | 1609.0 | 14.3 | 1602.0 | 1597.6 | 2.5 | 0.6 | 1601.0 | 1598.1 | 1.8 | 19.4 | 1603.0 | 1602.0 | 0.7 | 73.7 | 1603.0 | 1602.0 | 0.7 | 86.4 |
| S\_54 | 2009.0 | 279.1 | 2000.0 | 1993.4 | 4.1 | 0.6 | 2000.0 | 1997.4 | 1.8 | 19.6 | 2001.0 | 2001.0 | 0.0 | 47.6 | 2001.0 | 2001.0 | 0.0 | 48.5 |
| S\_55 | 1330.0 | 2.5 | 1315.0 | 1308.0 | 3.4 | 0.7 | 1318.0 | 1315.0 | 2.7 | 27.4 | 1319.0 | 1318.9 | 0.3 | 84.3 | 1319.0 | 1319.0 | 0.0 | 82.5 |
| S\_60 | 2141.0 | 494.5 | 2116.0 | 2111.6 | 2.8 | 0.8 | 2124.0 | 2119.8 | 3.0 | 25.5 | 2126.0 | 2125.2 | 0.4 | 82.3 | 2127.0 | 2125.2 | 0.6 | 90.0 |
| S\_61 | 2136.0 | 384.4 | 2125.0 | 2117.3 | 3.9 | 0.7 | 2129.0 | 2124.4 | 2.5 | 35.7 | 2132.0 | 2131.0 | 0.9 | 114.3 | 2132.0 | 2131.0 | 0.7 | 134.4 |
| S\_62 | 1344.0 | 62.9 | 1321.0 | 1317.7 | 2.5 | 0.5 | 1329.0 | 1323.3 | 2.7 | 28.6 | 1335.0 | 1333.8 | 1.0 | 116.1 | 1335.0 | 1334.0 | 0.7 | 109.3 |
| S\_63 | 1942.0 | 36.3 | 1924.0 | 1920.1 | 2.2 | 0.8 | 1923.0 | 1918.5 | 2.7 | 23.2 | 1928.0 | 1926.3 | 1.2 | 89.3 | 1929.0 | 1927.3 | 1.0 | 88.7 |
| S\_64 | 1948.0 | 7.2 | 1927.0 | 1920.9 | 3.6 | 0.9 | 1924.0 | 1921.0 | 2.1 | 29.5 | 1931.0 | 1928.4 | 1.3 | 97.8 | 1933.0 | 1929.4 | 1.9 | 91.3 |
| S\_65 | 2148.0 | 4.4 | 2131.0 | 2121.8 | 5.4 | 0.8 | 2130.0 | 2125.8 | 2.1 | 32.8 | 2135.0 | 2133.3 | 1.0 | 106.2 | 2136.0 | 2133.9 | 1.8 | 104.2 |
| 均值 | 1501.9 | 66.2 | 1490.9 | 1486.9 | 2.6 | 0.5 | 1491.9 | 1489.7 | 1.4 | 17.9 | 1493.6 | 1493.0 | 0.4 | 59.8 | 1493.8 | 1493.2 | 0.4 | 60.4 |
| #Wins | 20 | 17 | 5 | 1 | 0 | 20 | 0 | 0 | 0 | 20 | 1 | 1 | 3 | 8 | - | - | - | - |
| #Ties | 0 | 0 | 1 | 1 | 0 | 0 | 10 | 2 | 0 | 0 | 15 | 13 | 11 | 0 | - | - | - | - |
| #Losses | 0 | 3 | 14 | 18 | 20 | 0 | 10 | 18 | 20 | 0 | 4 | 6 | 6 | 12 | - | - | - | - |
| P值 | 8.5E-05 | 0.08 | 0.01 | 1.5E-04 | 1.9E-06 | 1.9E-06 | 4.8E-03 | 1.9E-04 | 1.9E-04 | 1.9E-06 | 0.16 | 0.04 | 0.54 | 0.50 | - | - | - | - |

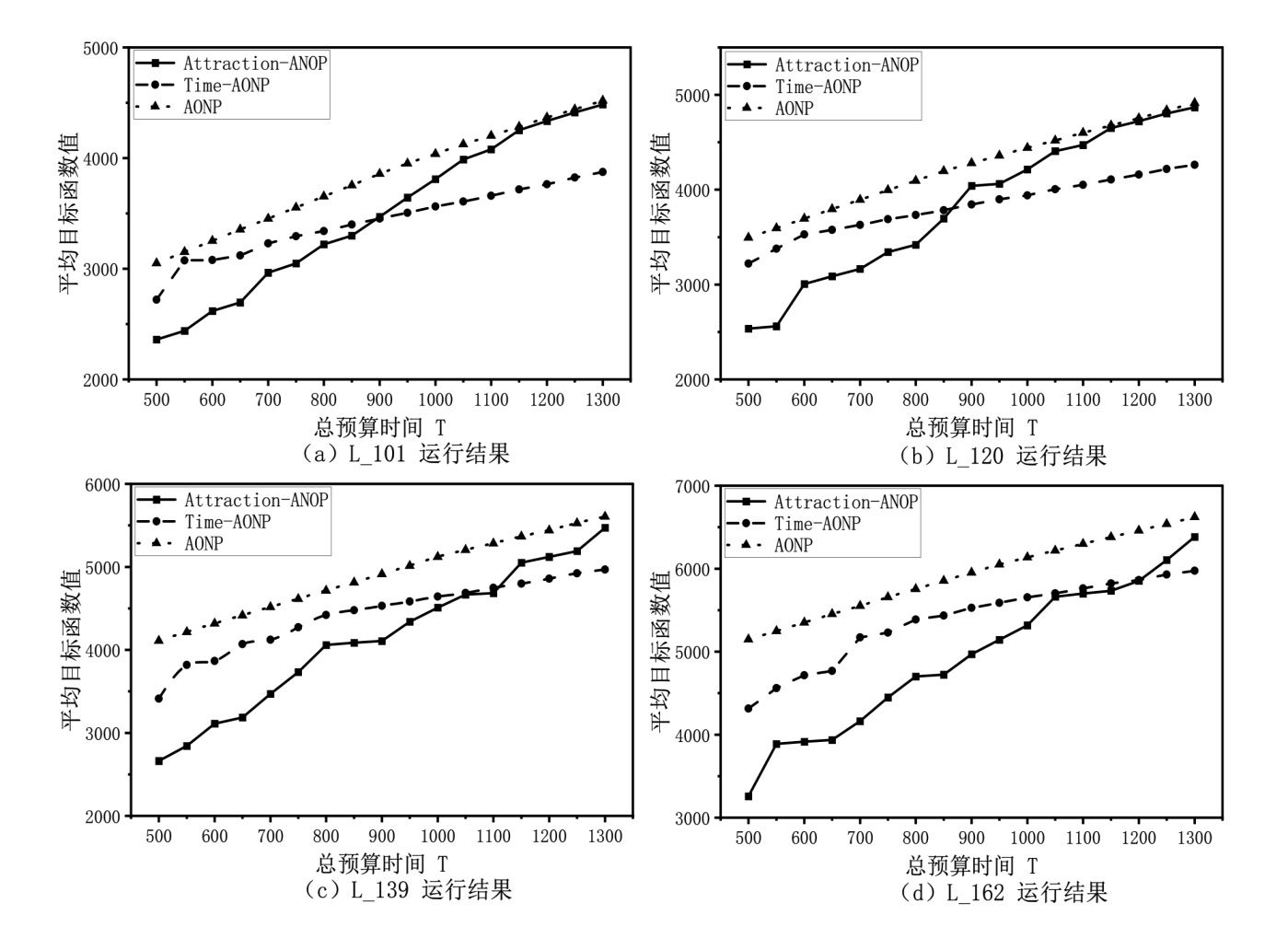
**表3. 大规模算例下的算法性能对比**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 算例 | Simple-EA | | | | EoH-EA | | | | ReEvo-EA | | | | LrMA-EA | | | |
| Best | Avg | Std | CPU/s | Best | Avg | Std | CPU/s | Best | Avg | Std | CPU/s | Best | Avg | Std | CPU/s |
| L\_101 | 4615.0 | 4581.1 | 26.6 | 0.8 | 4621.0 | 4617.1 | 2.5 | 68.7 | 4650.0 | 4642.5 | 4.5 | 257.0 | 4652.0 | 4644.1 | 3.3 | 261.4 |
| L\_120 | 4779.0 | 4737.9 | 21.5 | 0.9 | 4814.0 | 4807.4 | 5.6 | 74.2 | 4825.0 | 4822.8 | 1.4 | 190.9 | 4830.0 | 4824.5 | 2.9 | 214.7 |
| L\_139 | 5702.0 | 5586.1 | 59.6 | 1.0 | 5766.0 | 5733.1 | 22.0 | 88.5 | 5791.0 | 5788.0 | 2.2 | 257.0 | 5793.0 | 5788.8 | 2.1 | 312.7 |
| L\_162 | 6138.0 | 6053.3 | 37.9 | 1.2 | 6198.0 | 6152.1 | 24.8 | 131.8 | 6238.0 | 6232.0 | 4.3 | 274.9 | 6241.0 | 6231.5 | 4.7 | 292.9 |
| L\_181 | 6445.0 | 6360.4 | 47.8 | 1.9 | 6441.0 | 6427.8 | 7.3 | 158.9 | 6469.0 | 6461.6 | 4.7 | 375.9 | 6466.0 | 6460.8 | 3.6 | 348.8 |
| L\_200 | 9081.0 | 9040.9 | 45.6 | 3.2 | 9163.0 | 9111.1 | 26.0 | 225.4 | 9254.0 | 9241.9 | 5.2 | 398.5 | 9254.0 | 9242.0 | 6.0 | 369.9 |
| L\_219 | 6815.0 | 6765.0 | 45.7 | 1.5 | 6908.0 | 6826.8 | 34.9 | 86.0 | 6997.0 | 6987.3 | 6.0 | 363.1 | 6999.0 | 6990.8 | 7.6 | 385.8 |
| L\_242 | 8613.0 | 8456.4 | 101.2 | 2.0 | 8580.0 | 8425.9 | 78.5 | 95.1 | 8785.0 | 8779.1 | 10.5 | 236.3 | 8785.0 | 8777.9 | 5.7 | 253.0 |
| L\_261 | 8905.0 | 8799.5 | 106.7 | 2.3 | 8956.0 | 8794.9 | 110.4 | 108.4 | 9122.0 | 9110.8 | 6.4 | 327.7 | 9122.0 | 9113.5 | 6.4 | 366.6 |
| L\_280 | 9929.0 | 9635.3 | 208.2 | 2.6 | 9608.0 | 9274.5 | 127.6 | 99.1 | 10055.0 | 10040.4 | 19.7 | 258.5 | 10054.0 | 10042.1 | 7.1 | 272.9 |
| L\_303 | 10239.0 | 10167.8 | 41.1 | 3.8 | 10244.0 | 10194.3 | 49.5 | 197.0 | 10393.0 | 10386.6 | 3.7 | 534.1 | 10396.0 | 10389.0 | 3.6 | 538.5 |
| L\_322 | 2162.0 | 2150.8 | 7.1 | 0.5 | 2171.0 | 2167.1 | 2.3 | 70.6 | 2185.0 | 2182.9 | 1.8 | 155.2 | 2181.0 | 2174.6 | 3.6 | 167.1 |
| L\_344 | 10942.0 | 10475.6 | 305.9 | 3.6 | 10740.0 | 10290.8 | 238.3 | 118.7 | 11345.0 | 11323.0 | 16.9 | 414.8 | 11351.0 | 11337.0 | 8.2 | 425.9 |
| L\_359 | 11162.0 | 10947.9 | 196.0 | 4.2 | 10859.0 | 10396.6 | 195.2 | 153.2 | 11374.0 | 11306.0 | 150.5 | 222.8 | 11435.0 | 11409.1 | 11.0 | 329.2 |
| L\_384 | 12346.0 | 11959.4 | 318.6 | 4.2 | 11010.0 | 10644.6 | 166.8 | 113.9 | 12659.0 | 12162.6 | 426.1 | 211.3 | 13028.0 | 12655.0 | 329.2 | 209.4 |
| L\_401 | 12671.0 | 12511.0 | 120.2 | 6.4 | 12449.0 | 12279.3 | 131.7 | 226.4 | 12852.0 | 12822.3 | 13.9 | 589.2 | 12848.0 | 12835.3 | 7.9 | 559.9 |
| L\_420 | 13372.0 | 13116.5 | 255.1 | 6.8 | 13253.0 | 12239.1 | 380.9 | 189.3 | 14717.0 | 14478.8 | 189.6 | 385.5 | 14612.0 | 14256.8 | 416.6 | 472.8 |
| L\_439 | 14242.0 | 13720.4 | 536.8 | 5.9 | 13050.0 | 12166.1 | 342.4 | 165.8 | 15448.0 | 14120.0 | 781.7 | 261.6 | 15025.0 | 14122.8 | 588.1 | 294.8 |
| L\_459 | 14036.0 | 13984.0 | 71.9 | 9.5 | 13895.0 | 13811.4 | 63.4 | 277.9 | 14187.0 | 14165.9 | 17.3 | 977.4 | 14174.0 | 14162.8 | 8.8 | 928.2 |
| L\_480 | 14959.0 | 14387.0 | 407.3 | 7.1 | 13836.0 | 13138.0 | 356.6 | 179.7 | 15500.0 | 15462.6 | 79.9 | 574.3 | 15521.0 | 15428.1 | 88.0 | 673.7 |
| L\_502 | 15145.0 | 15074.6 | 62.4 | 10.6 | 15046.0 | 14797.3 | 159.7 | 343.3 | 15319.0 | 15302.3 | 13.5 | 933.3 | 15336.0 | 15324.3 | 6.3 | 952.0 |
| L\_599 | 14042.0 | 12747.0 | 634.3 | 5.9 | 11611.0 | 11129.8 | 232.9 | 191.0 | 13119.0 | 11800.3 | 652.4 | 196.4 | 13575.0 | 12389.5 | 467.7 | 231.3 |
| L\_701 | 17247.0 | 16597.4 | 682.8 | 9.7 | 14120.0 | 13703.9 | 211.1 | 263.6 | 15828.0 | 14073.3 | 1067.8 | 337.6 | 15054.0 | 14325.6 | 710.0 | 332.2 |
| 均值 | 10156.0 | 9906.7 | 188.7 | 4.2 | 9710.4 | 9440.4 | 129.1 | 157.7 | 10309.2 | 10073.6 | 151.3 | 379.7 | 10292.7 | 10127.2 | 117.3 | 399.7 |
| #Wins | 2 | 2 | 4 | 23 | 0 | 0 | 7 | 23 | 8 | 7 | 7 | 17 | - | - | - | - |
| #Ties | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 1 | 0 | - | - | - | - |
| #Losses | 21 | 21 | 19 | 0 | 23 | 23 | 16 | 0 | 12 | 16 | 15 | 6 | - | - | - | - |
| p 值 | 0.81 | 7.3E-04 | 0.76 | 1 | 0.53 | 1.2E-07 | 0.96 | 1 | 0.98 | 0.03 | 0.82 | 0.99 | - | - | - | - |

**4. 5 平行弧机制有效性分析**

为验证AONP模型中平行弧机制的实际价值，本节设计对比实验以量化其在时间成本与路径吸引力之间的权衡能力。具体而言，将AONP与两类极端策略基准模型进行比较：Time-AONP（在每组节点对间强制选择通行时间最短的弧段）和Attraction-AONP（强制选择吸引力评分最高的弧段）。上述三类模型在四组大算例（L\_101、L\_120、L\_139及L\_162）上各独立运行10次，结果呈现于图6与图7。

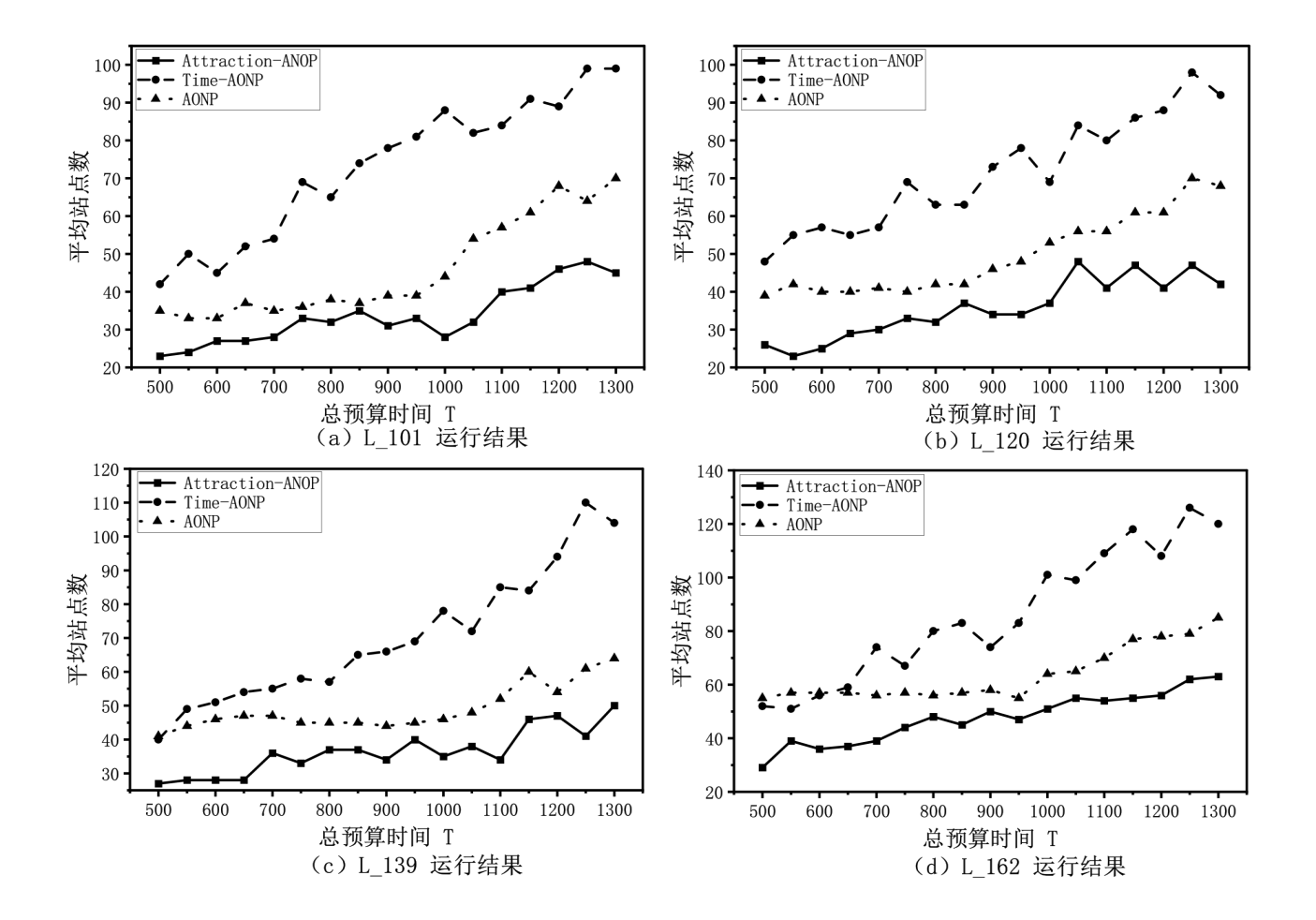
从图6的平均目标函数值变化趋势可见，AONP在总预算时间的整个区间内，其平均目标函数值普遍优于或至少与两个基准模型持平。这表明平行弧的引入赋予算法灵活选择的能力，使其能根据剩余时间动态调整路径属性，避免陷入单一优化维度的局部最优。



**图6. 不同弧段选择策略下平均目标函数值随预算时间的变化趋势**

图7进一步揭示了策略差异的本质。Time-AONP虽通过最短路径实现了最高的节点访问量，但其边际收益递减现象明显：当T增加后，目标函数值未随访问节点数同步提升，说明快速通行的低质量路段无法累积足够吸引力。相反，Attraction-AONP虽每段路径收益较高，却因高吸引力弧段通常伴随长通行时间，导致在有限预算内可访问节点数锐减，整体目标值受限。而AONP则有效避免了上述两种极端情况，它不盲目追求站点数量，也不单纯追求高吸引力，而是在通行时间和路径收益之间寻求平衡，通过平行弧机制动态调整路径选择策略，使其能够最充分地利用有限的预算时间，实现目标函数值的最大化。

综上，平行弧机制通过赋予AONP灵活权衡时间与吸引力的能力，显著提升了算法在复杂场景下的优化性能。



**图7. 不同弧段选择策略下平均访问节点数随预算时间的变化趋势**

5 结束语

针对个性化骑行路线规划中现有方法难以权衡平行路径且低估节点价值的关键局限，本文提出AONP模型。实验结果清晰地展示了AONP相较于传统单策略模型的优势，特别是在优化路径吸引力和时间效率之间的权衡方面表现突出。

在算法求解层面，进一步设计LrEA，将LLM的语义理解与代码生成能力深度嵌入进化框架，通过经验树反思机制与四类差异化元进化算子（结构重构、机制融合、参数调优、自适应增强）实现启发式策略的自动化迭代与自我优化。跨LLM后端测试验证LrEA在不同模型上均保持稳定优化趋势，展现出良好的部署灵活性。大规模基准对比显示，LrEA生成的求解器在小规模算例中逼近理论最优，在高维复杂场景下较传统元启发式算法与当前先进LLM增强方法（EoH、ReEvo）在求解质量方面实现超越。

未来的研究可以着眼于以下几个方向：首先，进一步优化AONP模型，使其能够更好地适应动态变化的环境条件，如天气、交通状况等；其次，将LrEA框架应用于更多实际场景，如城市物流路径规划、旅游路线推荐等，进一步验证其通用性与实用性。

参考文献

1. 央视新闻客户端. 从代步工具到生活方式，1.3亿骑行用户支撑起三千亿大产业[EB/OL]. (2025-08-31) [2024-09-01]. <https://news.bjd.com.cn/2025/08/31/11290484.shtml>.
2. Souffriau W, Vansteenwegen P, Vanden Berghe G, et al. The planning of cycle trips in the province of east flanders[J]. Omega, 2011, 39(2): 209-213.
3. Verbeeck C, Vansteenwegen P, Aghezzaf E H. An extension of the arc orienteering problem and its application to cycle trip planning[J]. Transportation Research Part E: Logistics and Transportation Review, 2014, 68: 64-78.
4. Kaur R, Goyal V, Gunturi V M V. Finding the most navigable path in road networks [J]. Geoinformatica, 2021, 25(1): 207-40.
5. Tricoire F, Parragh S N, Gansterer M. The hiking tourist problem [J]. Central European Journal of Operations Research, 2024: 25.
6. Dhein G, De Araújo O C B, Cardoso G. Genetic local search algorithm for a new bi-objective arc routing problem with profit collection and dispersion of vehicles [J]. Expert Systems with Applications, 2018, 92: 276-88.
7. Chen C, Chen X, Wang L Y, et al. MA-SSR: A Memetic Algorithm for Skyline Scenic Routes Planning Leveraging Heterogeneous User-Generated Digital Footprints [J]. IEEE Transactions on Vehicular Technology, 2017, 66(7): 5723-36.
8. Chen C, Gao L P, Xie X F, et al. Enjoy the most beautiful scene now: a memetic algorithm to solve two-fold time-dependent arc orienteering problem [J]. Frontiers of Computer Science, 2020, 14(2): 364-77.
9. Piedra-De-La-Cuadra R, Ortega F A. Designing Ecotourism Routes with Time-Dependent Benefits along Arcs and Waiting Times at Nodes [J]. Mathematics, 2024, 12(5): 624.
10. Zhang G W, Jia N, Zhu N, et al. Robust drone selective routing in humanitarian transportation network assessment [J]. European Journal of Operational Research, 2023, 305(1): 400-428.
11. Romera-Paredes B, Barekatain M, Novikov A, et al. Mathematical discoveries from program search with large language models[J]. Nature, 2024, 625(7995): 468-475.
12. Wu X, Wu S H, Wu J, et al. Evolutionary computation in the era of large language model: Survey and roadmap[J]. IEEE Transactions on Evolutionary Computation, 2025, 29(2): 534-554.
13. Stein N V, Bäck T. LLaMEA: A large language model evolutionary algorithm for automatically generating metaheuristics[J]. IEEE Transactions on Evolutionary Computation, 2025, 29(2): 331-345.
14. Liu F, Tong X, Yuan M, et al. Evolution of heuristics: Towards efficient automatic algorithm design using large language model[J/OL]. arXiv preprint, 2024 [2025-11-02]. arXiv:2401.02051.
15. Ye H, Wang J, Cao Z, et al. Reevo: Large language models as hyper-heuristics with reflective evolution[J]. Advances in neural information processing systems, 2024, 37: 43571-43608.

**Personalized Cycling Route Planning Driven by Large Language Models: A Reflective Evolutionary Algorithm Framework**

**Abstract:** Cycling has steadily grown and evolved into a highly popular and widely adopted mode of transportation that not only offers numerous health benefits but also showcases remarkable environmental friendliness. In today's modern era, the cycling market has expanded significantly and now serves hundreds of millions of users through a diverse range of various applications. These applications have made cycling more accessible and convenient for people from all walks of life. However, despite this remarkable growth, mainstream tools in the cycling domain continue to primarily focus on simple shortest - path calculations or offer predefined route suggestions. These rather basic approaches are far from sufficient as they fail to address several critical user requirements. For example, they do not take into account the scenic quality of the routes, which is important for cyclists who want to enjoy beautiful views during their rides. Road safety is another crucial aspect that these tools overlook, as they do not consider factors like traffic density, presence of bike lanes, and road conditions. Physical exertion management is also neglected, leaving cyclists without proper guidance on how to pace themselves according to their fitness levels. Additionally, the opportunities for meaningful stops, such as visiting local attractions or having a rest at nice cafes, are not provided. This technological gap directly constrains the quality of the user experience, as cyclists are left with less - than - optimal routes that do not meet their diverse needs. Moreover, it also hinders the further development of the cycling industry, as it fails to attract more users and expand the market potential.

Previous research applied the Arc Orienteering Problem (AOP) model to cycling route optimization. However, existing AOP formulations suffer from two fundamental limitations. The first involves network representation simplification where only a single path connects any two locations. Real-world cycling networks contain multiple parallel routes between nodes, such as fast but monotonous arterial roads versus slower but scenic waterfront paths. Current models cannot capture these alternatives, preventing personalized trade-off decisions. The second concerns undervaluation of node importance. Traditional AOP treats intermediate points as simple connectors while ignoring the intrinsic value of viewpoints, landmarks, and rest areas that cyclists actually seek out.

This work proposes the Arc Orienteering Problem with Node Profits (AONP) to address both deficiencies simultaneously. The model employs a multigraph structure that permits multiple parallel arcs between node pairs, each defined by distinct travel times and attraction scores. This enables the algorithm to make fine-grained adjustments based on user preferences and remaining time budgets. Additionally, AONP assigns profit values and stop durations to each location, incorporating nodes as independent optimization elements. This dual optimization framework accurately captures the ride-and-stop behavior where both journey quality and destination value matter equally.

Solving this more realistic model presents substantially greater difficulty. Conventional approaches rely on hand-designed heuristic rules requiring extensive expert knowledge and offering limited adaptability. Recent efforts to employ Large Language Models (LLMs) for automatic heuristic generation suffer from code hallucinations, prompt sensitivity, and weak constraint reasoning. This research introduces the LLM-Enhanced Reflective Evolutionary Algorithm (LrEA), which shifts the optimization focus from problem solutions to the heuristic operators that generate them. The framework incorporates three key innovations. An evolutionary knowledge reflection system uses an experience tree to store successful and failed operators from past generations, allowing the LLM to learn effective patterns. A meta-evolutionary operator framework includes four distinct operator types for structural exploration, mechanism combination, parameter refinement, and adaptive enhancement. A two-stage generation process separates high-level planning from code implementation, improving both quality and reliability.

Comprehensive experiments validate the proposed approach. Ablation studies demonstrate positive contributions from every component. The reflection mechanism proves most valuable, as its removal causes the largest average quality drop 0.43 percent and increases variability by 29 percent while reducing runtime by only 1 percent, confirming that learning from past errors provides excellent return on investment. Testing with four different LLM backends DeepSeek-V3, Doubao, Moonshot-Kimi, and Qwen shows consistent performance across all models, proving the framework's robustness and independence from any specific LLM implementation.

Comparative evaluation against established benchmarks further confirms effectiveness. For small problems where exact optimal solutions are available through Gurobi, LrEA-generated solvers achieve results within 0.6 percent of optimal. For large-scale problems involving hundreds of locations, LrEA significantly outperforms traditional metaheuristic algorithms and state-of-the-art LLM-based methods including EoH and ReEvo. It delivers 2 to 7 percent better average solution quality while demonstrating substantially more stable performance with comparable computing time. Statistical tests confirm these improvements are highly significant (p less than 0.05).

The contributions of this work are following. First, the AONP model introduces parallel arcs and node profit mechanisms that better represent real cycling experiences. Second, the LrEA framework enables automatic generation and self-optimization of heuristic strategies for complex combination problems. Third, extensive experimental evidence demonstrates that LrEA-generated solvers surpass both manually designed heuristics and existing LLM-enhanced methods in solution quality, stability, and scalability.

**Keywords:** cycling route; arc orienteering; large language model; evolutionary algorithm

1. [↑](#footnote-ref-0)