面向磁悬浮传输线动子调度的大语言模型辅助进化算法研究

吴 斌[1†]，刘新宇[1]，王 华[2]

(1. 南京工业大学 经济与管理学院，南京 211816；2. 南京工业大学 机械与动力工程学院，南京 211816)

摘 要: 磁悬浮传输系统在半导体、精密电子等对精度与洁净度要求高的制造产线中逐渐普及, 其动子调度问题是制约产线效率的核心问题. 鉴于此, 以最小化最大完工时间、总能耗和负载不均衡度为目标, 建立磁悬浮传输系统多目标动子调度的混合整数规划模型. 提出一种 LLM 驱动的演化算子自动设计框架 —— LLM-EvoOp. 该框架采用“思想-代码”二元表征与演化机制: 在算子演化阶段, LLM 基于问题描述与演化提示, 通过交叉与变异型提示驱动, 自动生成并迭代优化适用于调度问题结构特征的高性能遗传算子; 在实例求解阶段, 将演化所得算子嵌入标准 NSGA-II 流程, 实现高效的调度求解. 仿真实验表明, LLM-EvoOp 是求解此类问题的高效算法.

关键词: 大语言模型；演化算法；多目标调度；磁悬浮传输系统；NSGA-II；算子设计

中图分类号: TP183 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2026.0011

引用格式: 吴斌, 刘新宇, 王华. 面向磁悬浮传输线动子调度的大语言模型辅助进化算法研究 [J]. 控制与决策.

A large language model-driven framework for automated evolutionary operator design in multi-objective actuator scheduling of maglev transportation systems

WU Bin[1†]，*LIU Xin-yu*[1]，*WANG Hua*[2]

(1. College of Economics and Management，Nanjing Tech University，Nanjing 211816，China；2. College of Mechanical and Power Engineering，Nanjing Tech University，Nanjing 211816，China)

Abstract: Maglev transmission systems are gaining widespread application in manufacturing lines with stringent requirements for precision and cleanliness, such as those in the semiconductor and precision electronics industries. However, the scheduling of movers constitutes a core bottleneck restricting production line efficiency. To address this, this study aims to minimize makespan, total energy consumption, and load balance. It constructs a multi-objective mixed-integer programming model for the scheduling of maglev movers. Subsequently, an LLM-driven automated evolutionary operator design framework, named LLM-EvoOp, is proposed. This framework adopts a "Thought-Code" dual representation and evolution mechanism. In the operator evolution phase, the Large Language Model (LLM), driven by crossover and mutation-type prompts based on problem descriptions, automatically generates and iteratively optimizes high-performance genetic operators tailored to the structural characteristics of the scheduling problem. In the instance solving phase, the evolved operators are embedded into the standard NSGA-II workflow to achieve efficient scheduling solutions. Finally, simulation experiments demonstrate that LLM-EvoOp is an effective algorithm for solving this class of problems.

Keywords: large language model； evolutionary algorithm； multi-objective scheduling； maglev transportation system；NSGA-II；operator design

0 引 言

在制造业智能化转型的进程中, 高柔性、高效率的生产系统成为关键发展方向[1]. 磁悬浮传输系统(Maglev Transportation System, MTS) 作为一种非接触、高速、低振动的物料搬运设备, 以其动子独立控制与高精度定位的优势, 正逐步应用于半导体、精密

收稿日期: 2026-01-05；**录用日期:** 2026-03-06.

基金项目: 苏州市关键核心技术“揭榜挂帅”项目 (SYG2024014)；国家重点研发计划项目 (2022YFB3805201).

责任编委: 巩敦卫.

†通信作者. E-mail: wubin@njtech.edu.cn.

电子与新能源电池等对洁净度与精度要求极高的制造领域[2], 为实现“边输送边加工”的柔性生产模式提供了重要的技术支撑.

目前有关 MTS 的研究, 在机电控制、动子同步、路径切换及防碰撞等底层控制技术方面已取得了一定成果[3]. 然而, 在系统运行层面的动子调度问题却尚未得到充分重视. 该问题是实现磁悬浮传输系统高柔性、高效率目标的核心关键, 其本质是一个高度复杂、多约束且实时演化的动态优化问题. 与 AGV 等传统运输设备的调度问题相比, MTS 动子调度问题呈现出显著不同的复杂性特征: 1) 运行时空的强耦合与冲突动态传播. 大量动子在二维/三维空间网络中高密度、高速并行, 其路径时空交织, 冲突点随运行状态瞬时演变, 要求调度系统具备动态避碰与冲突消解机制[4]. 2) 与生产流程的深度嵌入与协同. 动子常作为移动式加工或装配平台, 深度嵌入生产节拍. 因此, 调度不仅关乎物流效率, 更需与严格的工艺时间、工序顺序实现毫秒级协同, 这对调度的精确性与鲁棒性提出了极致要求. 3) 优化目标的多维性与综合权衡需求[5]. 调度目标已从单一追求效率 (如最小化最大完工时间) 拓展为多维度性能的综合权衡. 决策者需在提升吞吐效率的同时, 兼顾动子间的负载均衡以保障系统长期运行稳定, 并优化能耗与设备损耗以控制成本[6]. 这些因素共同将 MTS 调度推向一个兼具多目标性与强耦合性的 NP-hard 难题.

虽然关于 MTS 动子调度问题的研究还未开展, 但与生产调度相关的研究已取得丰硕成果, 为本文的研究提供了良好的借鉴. 传统精确算法, 如分支定界法、列生成算法, 虽能保证全局最优, 但其计算复杂度随问题规模呈指数增长, 仅适用于小规模静态问题[7]; 基于规则或局部搜索的启发式方法, 虽计算高效, 却高度依赖人工经验, 缺乏通用性与自适应性[8]. 以 NSGA-II 为代表的多目标演化算法 (MOEA)[9], 因其全局搜索能力与 Pareto 前沿特性, 被广泛应用于制造系统与运输系统的调度问题[10]. 然而, MOEA 的效果高度依赖核心遗传算子 (交叉/变异) 的设计. 标准算子如 SBX 等虽具有普适性, 但缺乏领域知识嵌入, 难以充分利用问题领域知识, 导致求解质量提升不明显[11]. 而定制算子的设计又往往耗时费力, 依赖专家经验且难以迁移[12], 形成 MOEA 应用中的“算子设计瓶颈”. 因此, 实现算子设计的自动化与自适应化, 已成为制约多目标演化算法在复杂工业调度问题中进一步发展的核心瓶颈.

近年来, 大语言模型 (Large Language Model, LLM) 在代码生成与逻辑推理方面的突破, 为自动化算法设计提供了新途径[13]. 当前研究关于大语言模型在算法自动化设计方面的研究大致分为两种范式[14]. 范式一将 LLM 作为在线优化器, 嵌入优化求解的循环中, 使其在每一代演化中生成或修改候选解. Yang 等 [15] 提出的 OPRO(Optimization by PROmpting) 框架将优化问题以自然语言形式嵌入提示中, 让 LLM 基于历史信息直接生成和评估解; Liu 等 [16] 提出的 LMEA(LLM-driven EA) 框架, 使 LLM 在演化算法中实时生成子代解, 替代传统的交叉与变异操作; Wang 等 [17] 提出的 MAEF(Multi-Agent LLMs-driven Evolutionary Framework) 框架, 构建了多智能体协同系统, 由多个 LLM 在求解过程中分工完成问题定义、解生成与评估等任务. 这类方法虽灵活, 但存在 LLM 调用成本高、响应延迟大、解的可行性难以保证等缺陷, 在资源受限的工业实时调度中实用性较低[18]. 范式二将 LLM 从实时决策环节解耦, 转而作为离线的启发式算子生成器或算法设计师. Google 提出的 FunSearch 是此方向的代表, 它在演化框架中利用 LLM 生成针对特定组合优化问题的 Python 函数, 实现了对卡普雷卡尔常数等问题的创新求解[19]. Liu 等[20] 提出的 EoH(Evolution of Heuristics) 框架采用“思想–代码”双层表征机制, 通过同时演化自然语言思想与代码实现, 大幅降低了计算成本并提升了生成算法的求解质量. 此外, Ye 等 [21] 提出的 LLM-LNS(Large Language Model-driven Large Neighborhood Search) 框架利用 LLM 离线设计特定算法组件, 验证了大规模邻域搜索中的破坏与修复算子的可行性. Li 等[22] 提出的 ARS(Automatic Routing Solver) 是一个基于大语言模型的自动路由求解系统, 能够根据自然语言描述自动生成约束感知的启发式代码, 在复杂 VRP 问题上表现出色. 与在线方法相比, 该范式具备成本可控、结果可验证、可复现性强等优势, 特别适合用于构建高质量、领域特化的算法组件.

尽管第二种范式取得了进展, 但如何高效、自动生成演化算法中的核心算子这一细粒度组件, 仍是一个研究空白. 与生成完整算法相比, 算子级演化能显著降低搜索空间与 LLM 调用成本, 提升工程可行性, 同时保持与 MOEA 框架 (如 NSGA-II) 的兼容. 演化完整算法需要同时搜索算法结构、参数及所有组件的实现, 其搜索空间是指数级组合; 而算子级演化将搜索目标聚焦于预定义接口下的交叉、变异函数, 其搜索空间被严格限制在有限、可控的代码生成任务内, 复杂度显著降低. 为突破传统 MOEA 的“算

子设计瓶颈”并规避 LLM 在线应用的高成本问题, 本文提出一种新颖的混合智能优化框架——LLM-EvoOp (large language model-driven evolutionary operator framework). 该框架的核心思想是通过角色重塑与任务解耦, 将 LLM 从“在线求解者”转变为“离线遗传算子设计师”, 构建“算子演化—实例求解”的双阶段协同机制. 在算子演化阶段, 我们设计了一种“思想-代码”二元演化机制, 驱动 LLM 自动生成面向 MTS 调度问题特性的高性能交叉与变异算子; 在实例求解阶段, 将这些高性能算子嵌入标准 NSGA-II 框架, 实现高效、稳定的调度求解. 相较于依赖人工经验的传统算子设计, 这种自动设计范式能够在大规模搜索空间中挖掘出人类直觉难以发现的非直观逻辑, 有效突破了人工设计在处理强时空耦合约束时的认知局限, 实现了以低成本、零样本的方式快速适应不同产线场景的调度需求.

本文的主要贡献包括: (1) 针对磁悬浮传输系统多动子协同作业场景, 构建了一个考虑连续时空耦合与防冲突约束的多目标调度模型. (2) 提出了 LLM-EvoOp 优化求解框架, 基于问题特征的启发式演化机制, 实现了遗传算子的自动化生成.

1　问题描述与建模

1.1　前提假设

本文研究的问题描述如下: 在由V个节点和E条轨道段构成的磁悬浮传输网络$(G=V,E)$上, T个运输任务需由M个同质磁悬浮动子协同完成. 每个任务具有取货节点、送货节点、处理时间以及可选的时间窗约束. 同时, 每台动子具有确定的物理参数, 且动子始终沿最短路径移动. 根据问题描述做出以下假设: 1. 动子同构: 所有动子具有相同的物理特性. 2. 单任务约束: 每个动子在任意时刻最多承载一个运输任务, 任务不可分割且必须由同一动子完成全程运输. 3. 确定性运动模型: 动子运动遵循梯形速度曲线, 行驶时间和能耗可由距离精确计算, 不考虑随机扰动. 4. 静态全知信息: 所有任务信息 (位置、时间窗、前序约束) 在调度决策前已知, 属于静态调度问题. 5. 无边冲突: 仅考虑节点处的时空冲突检测, 该冲突可通过调度规则避免, 暂不考虑轨道段容量约束.

1.2　参数与变量

模型中所使用的符号变量含义如下: 动子索引号$j\in\{1,2,\ldots,M\}$; 任务索引号$i,k\in\{1,2,\ldots,T\}$; 传输线拓扑网络$G=(V,E)$, 其中物理节点集合V, 拓扑网络中节点间的有向边 (轨道) 集合E; $Path(j,t)$表示动子j在时刻t所占据的网络节点; E_j^{move}和E_j^{wait}分别表示动子单位距离的行驶能耗与动子单位时间的待机能耗; W_j和W_{avg}分别表示动子工作负载和动子平均负载; P_i表示任务i的标准加工或装卸时间; Loc_k^{start}和Loc_i^{end}分别表示任务k的起始节点和任务i的结束节点; τ表示动子从一点移动到另一点所需行驶时间; $[ES_i,LS_i]$表示任务i的硬时间窗; λ表示惩罚系数; B为足够大的正常数. 定义如下决策变量:

C_i, 连续变量, 任务i的实际完成时间; S_i, 连续变量, 任务i的实际开始执行时间; X_{ij}, 二元变量, 决定任务i是否分配给动子j, 若分配则为 1, 否则为 0; Y_{ikj}, 二元变量, 决定动子j是否在执行完任务i后紧接着执行任务k, 紧接着执行任务则为 1, 否则为 0.

1.3　数学模型

目标函数为:

$$minf_1=\max_{i\in T}\{C_i\},\tag{1}$$

$$minf_2=\sum_{j\in M}(E_j^{move}+E_j^{wait}),\tag{2}$$

$$minf_3=\sqrt{\frac{1}{M}\sum_{j\in M}(W_j-W_{avg})^2}.\tag{3}$$

约束条件为:

$$\sum_{j\in M}X_{ij}=1,\forall i\in T,X_{ij}\in\{0,1\},\ \forall i\in T,j\in M,\tag{4}$$

$$\sum_{i\in T,i\neq k}Y_{ikj}\leqslant X_{kj},\forall k\in T,j\in M,\tag{5}$$

$$\sum_{k\in T,k\neq i}Y_{ikj}\leqslant X_{ij},\forall i\in T,j\in M,\tag{6}$$

$$C_i=S_i+P_i,\forall i\in T,\tag{7}$$

$$S_i\geqslant ES_i and C_i\leqslant LS_i,\forall i\in T,\tag{8}$$

$$S_k\geqslant C_i+\tau(Loc_i^{end},Loc_k^{start})-B(1-Y_{ikj}),\quad\forall i,k\in T,i\neq k,\forall j\in M,\tag{9}$$

$$(j_1,t)\cap Path(j_2,t)=\varnothing,\quad\forall j_1,j_2\in M,j_1\neq j_2,\forall t.\tag{10}$$

公式 (1)-(3) 分别表示最小化最大完工时间, 总能耗和负载不均衡度的目标函数. 约束 (4) 确保每个任务i必须且仅被分配给一个动子j. 约束 (5) 和约束 (6) 定义了分配给同一个动子的任务之间的先后顺序关系. 约束 (5) 确保对于分配给动子j的任何任务k(非j的第一个任务), 最多只有一个任务i紧邻其前. 约束 (6) 确保对于分配给动子j的任何任务i(非j的最后一个任务), 最多只有一个任务k紧邻其后. 约束 (7) 定义了任务i的完成时间与其开始时间和加

工时间之间的关系. 约束 (8) 确保任务i的执行必须在其硬时间窗内. 约束 (9) 确保了动子的时间连续性, 规定如果动子j需在执行任务i后立即执行任务k(Y_{ikj}=1), 则任务k的开始时间S_k必须在任务i完成(C_i)并且动子j具有足够的行驶时间T从i的结束位置移动到k的起始位置之后. 约束 (10) 是确保系统安全运行的物理防冲突约束. 该约束要求任意两个动子在任何时刻都不能占用相同的物理空间, 或必须保持最小安全时距. 在解码策略中, 该约束通过基于时空资源的冲突检测与解析规则来保证.

2 LLM-EvoOp

为有效求解上述复杂的多目标调度问题, 本文基于 NSGA-II 算法并结合大语言模型, 提出了一个混合智能优化算法框架 LLM-EvoOp. LLM-EvoOp 的核心思想是: 大语言模型不直接参与调度问题求解, 而是扮演一个“元优化器”或“启发式设计师”的角色. 在算子演化阶段, 通过模拟生物进化过程, LLM 自动生成一套针对当前调度问题特征的高效的遗传算子. 随后在实例求解阶段, 这些遗传算子被无缝集成到 NSGA-II 算法框架中, 实现对具体调度问题的高效求解. 该框架将 LLM 的创造性、启发式推理能力与演化算法的结构化、并行化搜索能力进行有效深度融合. 图 1 展示了该框架的整体结构.

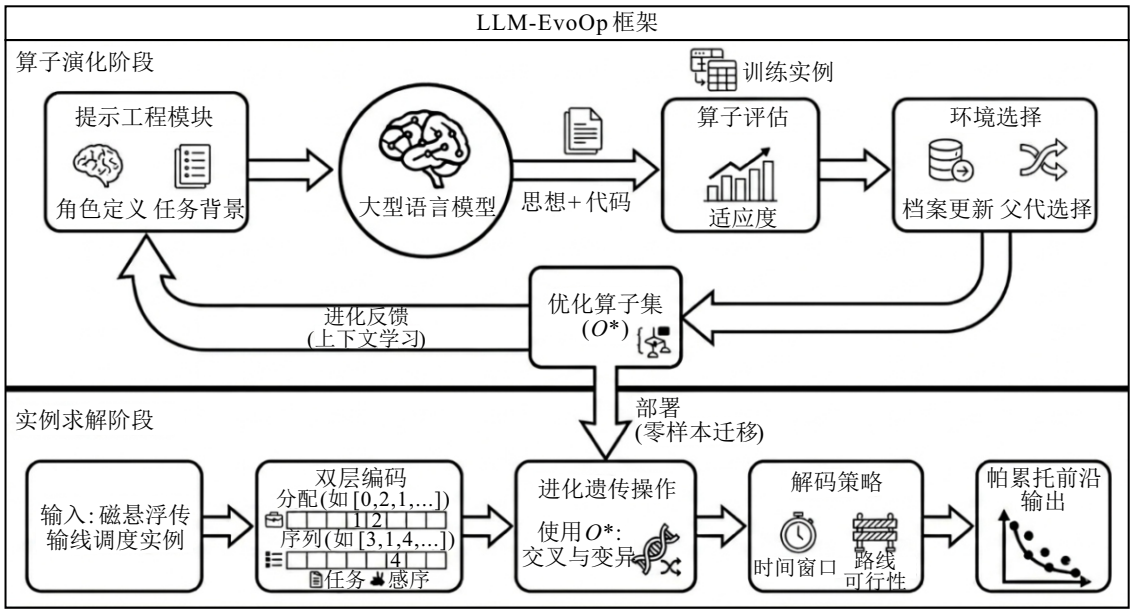


图1 LLM-EvoOp 整体框架结构

2.1 LLM 驱动的遗传算子演化框架

LLM-EvoOp 框架旨在通过大语言模型的生成与推理能力, 自动化地演化出高性能的遗传算子. 该框架将演化过程分解为四个核心模块: 问题定义与编码/解码, 初始算子生成, 演化优化与评估选择. 这些模块通过结构化的提示词进行驱动, 形成了一个自动化设计闭环.

2.1.1 问题定义与编码/解码模块

该模块作为整个框架的输入接口, 负责将复杂的磁悬浮调度问题转化为 LLM 可理解的结构化表述. 模块接收用户提供的关于系统动子, 任务时间窗与网络拓扑的自然语言描述, 并利用预定义的提示词模板, 引导 LLM 从中提取关键的优化要素. 这些要素包括优化目标, 决策变量以及复杂的硬性约束.

在此基础上, 模块进一步规定了用于 NSGA-II 算法的双层染色体编码方案, 如图 2 所示. 第一层表示任务的分配方案, 每个整数基因对应一个动子编号, 用于指定每个任务由哪台动子执行, 如任务 1 由动子 2 执行; 第二层表示任务的调度优先级, 该层基因位的索引对应任务编号, 而基因位上的整数数值代表该任务的优先级. 数值越小, 代表该任务在调度队列中的优先级越高, 将被优先处理. 解码过程如下:

任务编号	任务1	任务2	任务3	任务4	任务5
任务分配	2	1	3	1	2
任务排序	3	0	4	1	2

图2 双层染色体编码

Step1: 初始化系统时钟 t=0, 加载路网拓扑 $G(V,E)$, 将所有动子置于初始位置, 并将所有待处理任务装载至待调度队列.

Step2: 读取染色体第一层的任务分配向量, 将每

个任务绑定至指定的动子, 形成初步的“动子-任务”集合.

Step3: 读取染色体第二层的任务排序向量, 将所有任务按照该序列定义的优先级重排.

Step4: 按照 *Step3* 确定的调度序列, 依次对每个任务进行调度. 每完成一个任务, 更新该动子的位置、累计能耗及工作时间. 全部任务调度完成后, 对生成的调度方案进行冲突检测, 检查是否存在同一时刻多个动子占用同一节点的情况, 将所有检测到的冲突记录至约束违规列表.

Step5: 若过程中检测到约束违规, 则赋予该个体与违规次数成正比的惩罚值 λ, 并将该惩罚值同时叠加至三个目标函数上, 确保不可行解在演化过程中被有效区分和淘汰.

2.1.2 初始算子生成模块

初始算子生成模块负责创建一组高质量, 多样化的遗传算子, 为后续演化奠定基础. 该模块利用精心设计的初始化提示词, 引导 LLM 扮演“智能算法专家”的角色, 生成具备不同搜索倾向的初始算子. 提示词明确要求 LLM 创建两个逻辑迥异的交叉算子: 其一是收敛导向型算子, 侧重于利用已知的有效模式, 如在重组子代时优先继承父代中能带来更低能耗或更短路径的基因片段, 以快速提升解的质量; 其二是多样性导向型算子, 侧重于引入受控的随机性, 如以一定概率从不同质量的父代中引入基因, 以维持种群的广泛探索能力. 每个生成的算子均以“思想-代码”二元结构进行封装, 其中“思想”部分以自然语言阐述其启发式逻辑与设计原理, 保证了设计的可解释性; “代码”部分则是符合预定接口的可执行 Python 函数, 确保了功能的可执行性. 为保证 LLM 生成算子的语义一致性, 本框架采用结构化输出绑定机制: 要求 LLM 以 JSON 格式同时输出"思想"和"代码"两个字段, 形成显式关联. 提示词设计遵循"分析-思想-代码"三步法, 强制 LLM 先形成完整的启发式思想, 再根据该思想编写代码实现. 在演化过程中, 父代的思想与代码同时传递给子代生成提示, 确保思想演化链的可追溯性. 这种设计使得每个算子的设计意图与实现逻辑保持一致, 便于后续分析和改进.

2.1.3 演化优化模块

演化优化模块是驱动算子性能持续提升的核心. 它通过两类专门的提示词——交叉型提示词与变异型提示词, 在 LLM 的驱动下模拟生物进化中的交叉与变异操作. 交叉型提示词旨在融合两个优秀父代算子的优点. 它会将两个父代的性能得分、启发式“思想”和实现“代码”一同输入给 LLM, 并要求其执行一个严谨的三步流程: 首先, 批判性分析各父代在收敛性, 多样性或特定目标优化上的优劣; 其次, 提出一个能协同增效的新启发式思想; 最后, 根据新思想生成融合后的代码. 变异型提示词则侧重于对单个优秀父代算子进行精准改进. 它同样遵循三步流程: 诊断父代算子的潜在局限性; 提出一个针对性的小范围修正思想; 最终在父代代码基础上实现该定向变异. 所有新生成的子代代码都会经过一个自动化的修复与约束合规性检查流程, 以保障演化过程的稳健性.

2.1.4 算子评估与选择模块

算子评估与选择模块为演化过程提供关键的性能反馈, 驱动种群向适应度高的区域演进. 该模块采用一种高效的代理评估策略: 将每一个待评估的算子与一个基准变异算子配对, 形成一个临时的 NSGA-II 求解器. 随后, 该求解器在一组具有代表性但规模较小的训练实例集上进行一轮轻量级的快速优化. 优化结束后, 计算所得帕累托前沿的超体积指标, 该指标在所有训练实例上的平均值即被定义为该算子的适应度. 这种基于代理性能的评估方式, 虽不及在完整问题上测试精确, 但能显著降低计算成本, 使得在有限资源下对大量候选算子进行快速、频繁的筛选成为可能. 基于适应度分数, 模块随后采用标准的演化算法选择机制, 保留精英算子, 淘汰劣质算子, 并更新存档中的历史最优算子, 从而确保算法质量在迭代中持续提升.

2.1.5 算子演化阶段的算法流程

LLM-EvoOp 框架下算子演化阶段的流程如算法 1 所示. 算法初始化阶段首先通过初始化提示生成包含 A 个算子个体的初始种群, 并对种群中所有算子进行编译与单元测试以保障代码可执行性; 随后, 对每个算子进行适应度评估, 通过将其与基准算子组合, 构成轻量级 NSGA-II 算法 (与标准 NSGA-II 使用相同的算法逻辑, 仅通过缩减参数规模实现快速评估: 种群规模 N=20, 迭代次数 Gen=15, 以支撑算子演化阶段的大规模算子适应度计算). 在训练实例集上计算平均超体积指标作为适应度分数. 演化主循环从第 1 代至最大代数 Gen 依次执行, 每代开始时初始化空子代集合, 并通过循环结构依次生成子代个体: 每次迭代中采用锦标赛选择从当前种群选取两个父代算子, 利用交叉型提示驱动 LLM 融合父代思想与代码生成子代算子的自然语言描述与可执行代码, 针对 LLM 存在幻觉问题, 框架会对生成代码进行语法检查与约束合规性验证, 无法运行的

代码会被直接丢弃; 交叉算子与变异算子分别独立演化, 若子代算子通过可行性检验则将其加入子代集合. 每代演化结束后, 通过环境选择机制将父代与子代种群合并, 基于适应度与多样性指标筛选出新一代种群, 同时更新存档以记录当前最优算子个体. 算法最终输出存档中性能最优的算子集合作为算子演化阶段的成果, 为后续实例求解阶段提供可复用的高性能遗传操作组件.

算法1 算子演化阶段

Input: LLM, $prompts\ P_{init}$, P_{cross}, P_{mut}, $population\ A$, $generationsGen$

1 $O_{ex} \leftarrow \langle GenerateOperator(\text{LLM}, P_{init})\rangle\ repeated\ M\ times\}$

2 for $g = 1\ to\ Gen$ do

3 $Off \leftarrow \emptyset$

4 for $i = 1\ to\ \lceil A/2 \rceil$ do //生成子代

5 $p1, p2 \leftarrow \text{TournamentSelect}(O_{cx})$

6 $(t_{child}, code_{child}) \leftarrow \text{LLM}(P_{cross}, p1, p2)$ //交叉算子演化

7 $code_{child} \leftarrow \text{Repair\_ConstraintCheck}(code_{child})$

8 if $Feasible(code_{child})$ then $Of \leftarrow Off \cup \{\text{MakeOperator}(t_{child}, code_{child})\}$ end //可行性验证

9 end for

10 $O_{ex} \leftarrow \text{EnvironmentalSelection}(O_{ex} \cup Off)$ //环境选择

11 Archive_cx.update(TopK(O_{ex})) //存档更新

12 end for

13 $O\_mx \leftarrow \langle \text{GenerateOperator}(\text{LLM}, P_{intmx})\rangle\ repeated\ M\ times\}$

14 for $g = 1$ to Gen do... //变异算子演化, 逻辑同上

15 end for

Output: $O^* = \{Cx_{best}, Mx_{best}\}$ //输出冠军算子

2.2 实例求解阶段

在实例求解阶段, 将算子演化阶段得到的最优算子集嵌入标准 NSGA-II 框架, 用于高效求解具体调度实例, 流程如算法 2 所示. 该过程以问题实例、演化算子集、种群规模、最大迭代代数及多目标函数作为输入. 首先, 算法采用双层编码方案初始化种群, 并通过解码策略评估每个个体的目标值, 同时基于离散化时空资源的冲突检测机制, 将时间轴以 0.01 秒精度离散化, 以网络节点为空间单元, 检测任意时刻-节点对上是否存在多动子占用. 为保证所得解的可行性, 将检测到的冲突作为软约束处理, 通过向目标函数添加与冲突数量成正比的惩罚项, 使 NSGA-II 在进化过程中自然淘汰冲突解. 随后进入主迭代循环, 在每一代中初始化空子代集合, 通过循环结构生成新个体直至满足规模要求. 每次迭代采用锦标赛策略选择父代个体, 依次应用算子演化阶段得到的最优交叉算子与最优变异算子执行遗传操作, 为确保遗传操作后个体的编码合法性, 算法在交叉和变异之后执行可行性修复: 对于任务分配段, 将越界的动子编号随机映射到有效范围 [0, M-1] 内; 对于任务排序段, 若检测到重复或缺失, 则重新生成随机排列. 所有子代经解码评估后加入子代集合. 每代结束时通过非支配排序与拥挤度距离从混合种群中筛选新一代个体, 若满足收敛条件则提前终止. 最终输出高质量的非支配解集作为帕累托最优调度方案.

算法2 实例求解阶段

Input: $problem\ instance\ D$, $operator\ set\ O* = \{Cx_{best}, Mx_{best}\}$, $population\ size\ M$, $max\ generations\ Gen$, $objectives\ f1, f2, f3$

1 $P \leftarrow \text{InitializePopulation}(D, M)$

2 $Evaluate\ (P, \text{using SimulationDecoder})$ //初始评估

3 for $g = 1$ to Gen do

4 $P_{off} \leftarrow \emptyset$

5 while $|P_{off}| < M$ do

6 $(p1, p2) \leftarrow TournamentSelect(P)$

7 $c\_child \leftarrow Cx_{best}(p1, p2)$ //加载冠军交叉算子

8 $c\_child \leftarrow Mx_{best}(c\_child)$ //加载冠军变异算子

9 $c\_child \leftarrow RepairFeasibility(c\_child)$

10 $(f, \text{violations}) \leftarrow SimulationDecode(c\_child)$ //解码

11 $f \leftarrow f + \lambda \cdot |violations|$ //约束处理

12 $P_{off} \leftarrow P_{off} \cup \{(c\_child, f)\}$

13 end while

14 $P \leftarrow EnvironmentalSelection(P \cup P_{off}, by\ NonDominatedSort + CrowdingDistance)$ //环境选择

15 if $Converged(P)\ then\ break$

16 end for

Output: $ParetoFront \leftarrow NonDominated(P)$ //输出非支配解集

3 实验结果与分析

为了验证 LLM-EvoOp 框架的有效性, 本文设计了一系列的实验. 围绕以下三个核心问题展开: 1. LLM-EvoOp 演化生成的遗传算子是否比通用标准

算子更有效. 2. 集成 LLM 演化算子的 LLM-EvoOp 算法, 性能是否优于当前主流的多目标优化算法. 3. LLM-EvoOp 框架中的关键组件各自对最终性能的贡献度是多少.

所有实验均运行在 Intel Core i7-13620H CPU, 16GB 内存, Windows 11 操作系统的计算机平台上, 编程环境为 Python 3.12.10, LLM 接口调用通用大模型 Deepseek-V3.2-Exp API, temperature 设置为 0.5, max tokens 设置为 8192, API 请求超时时间设置为 180s.

3.1　实验设置

为全面评估算法的性能与可扩展性, 本文采用参数化方法生成了不同规模的测试算例, 构建了两个算例集:

训练算例集: 用于算子演化阶段的算子适应度评估以及算子有效性验证实验. 该集合包含 10 个随机独立生成的算例, 运行次数设置为在每个固定规模算例上各运行 1 次, 规模设置如表 1 所示.

表1　训练算例集配置表 (Training-Suite)

算例集	动子数量	任务数量	节点数量
Training-Suite	3	15	20

基准测试算例集: 用于最终算法性能的综合对比与消融实验. 该集合包含小、中、大三种规模, 规模划分依据磁悬浮传输线的实际运行负荷设定: 小规模对应产线试运行或低冲突场景; 中规模对应正常生产时的标准负荷; 大规模对应高峰期或高冲突、高饱和场景, 这种分级旨在全面评估算法在不同拥堵程度下的鲁棒性. 每种规模包含 10 个随机独立生成的算例, 具体参数见表 2. 所有算例的任务处理时间, 取/送货点位置均在预设范围内随机生成.

表2　基准测试算例集配置表 (Benchmark-Suite)

规模	动子数量	任务数量	节点数量	重复次数
小规模	5	20	30	30
中规模	10	50	60	30
大规模	15	100	100	30

为全面评估 LLM-EvoOp 框架的性能, 本文选取 4 种代表性的多目标优化算法 S-NSGA-II (标准 NSGA-II)、MOPSO、MOEA/D 和 GA 作为对比基准. 这些算法涵盖了基于 Pareto 支配、分解以及聚合的不同优化范式, 确保了比较的广泛性和公正性. 算法各自参数设置如下: S-NSGA-II 交叉概率为 0.8, 变异概率为 0.2; MOPSO 与 MOEA/D 的参数设置具体参见文献 [23], GA 具体参考文献 [24] 设置自适应参数.

所有对比算法均采用与 LLM-EvoOp 相同的种群规模 (N=30(实验 1)/50(实验 2/3)) 和最大迭代次数 (Gen=20(实验 1)/100(实验 2/3)), 以保障比较的公平性.

采用 HV 和反转世代距离 (Inverted Generational Distance, IGD) 衡量算法的收敛性与多样性.其中, HV 值越大, 表明解集综合性能越好. HV 的参考点设置为在所有算法运行结果中各目标函数最大值的 1.1 倍. IGD 计算真实 Pareto 前沿上的最优解集到已知非支配解集之间的最小平均距离, IGD 值越小, 表明解集收敛性和分布性越好. 本文选取五个算法产生的解集合中的非支配解作为近似 Pareto 前沿 (Approximate Pareto Frontier, APF) 以代替真实 PF.

3.2　算子有效性验证

为验证 LLM 演化生成的遗传算子在磁悬浮调度问题中的有效性, 首先对演化过程中表现最优的交叉与变异算子进行了系统评估. 从表 3 可以看出, 交叉算子与变异算子的 HV 值分别稳定在 1.133 6 与 1.162 1, 均优于标准交叉与变异算子.

表3　算子演化性能指标对比表

指标	交叉算子	变异算子
初始HV值	1.003 5	1.123 6
最终HV值	**1.133 6**	**1.162 1**
提升率	+12.97%	+3.43%

本文在 10 个独立测试实例上对比了 LLM 演化算子与标准算子在轻量级 NSGA-II 框架下的性能. 表 4 展示了 LLM-EvoOp 在绝大多数实例上优于标准 NSGA-II. LLM 生成的演化算子在提升解集收敛性与分布性方面具有显著优势, 证明 LLM 引导的算子演化, 能够产生优于标准算子的遗传算子.

3.3　消融实验

为深入探索 LLM-EvoOp 框架中各核心组件的贡献, 本文设计 5 项消融实验变体, 包括: 完整模型 (Full Model)、仅演化代码 (Code-Only)、仅使用 LLM 交叉算子搭配一个标准的变异算子 (LLM-EvoOp-C)、仅使用 LLM 变异算子搭配一个标准的交叉算子 (LLM-EvoOp-M) 以及标准 NSGA-II(S-NSGA-II). 五个算子在对大、中、小所有实例进行了测试, 统计结果如表 5 所示. 消融实验表明, 完整模型的性能均显著优于其他消融变体, 验证了框架整体设计的有效性. 具体而言, LLM-EvoOp-M(仅使用 LLM 变异) 和 LLM-EvoOp-C(仅使用 LLM 交叉) 的性能均显著优于 S-NSGA-II, 证明了单独引入 LLM 设计的

表4 算子性能测试实例对比表

实例编号	指标	LLM-EvoOp	S-NSGA-II
1	HV	0.3884	0.3576
	IGD	0	0.0623
2	HV	0.1844	0.1782
	IGD	0	0.0142
3	HV	0.5565	0.5246
	IGD	0.0015	0.0230
4	HV	0.5632	0.5365
	IGD	0	0.0267
5	HV	0.1978	0.1577
	IGD	0	0.0368
6	HV	0.9993	0.8795
	IGD	0	0.0250
7	HV	0.6949	0.5506
	IGD	0.0165	0.0288
8	HV	0.7240	0.7231
	IGD	0.0048	0.0099
9	HV	0.8153	0.8263
	IGD	0.0009	0.0195
10	HV	0.4127	0.3579
	IGD	0.0011	0.0596

交叉或变异算子都是有效的, 能够带来性能提升. Full Model(完整模型) 的性能一致优于 Code-Only (仅演化代码) 模型. 这一关键对比有力地证明了“思想-代码”二元演化机制的优越性. 它表明, 让 LLM 首先在自然语言层面进行逻辑推理和启发式“思想”的生成与融合, 能够引导其产生结构更合理、逻辑更优越的代码实现, 从而获得比直接在代码层面进行“黑盒”式演化更好的效果. 这激活了 LLM 更深层次的结构化推理能力.

表5 消融实验统计结果

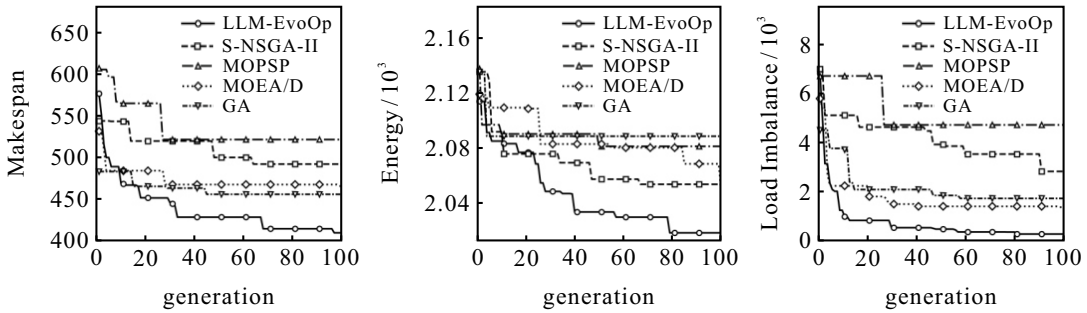
变体	指标	均值	vsFull	vsS-NSGA-II	组件贡献
FullModel	HV	**0.8821**	-	**+35.62%**	–
	IGD	**0.0369**	-	**-74.98%**	–
Code-Only	HV	0.7698	−12.73%	+18.36%	Thought:12.73%
	IGD	0.0927	+151.22%	−37.15%	Thought:151.22%
LLM-EvoOp-C	HV	0.8065	−8.57%	+24.00%	Mutation:8.57%
	IGD	0.0583	+58.00%	−60.47%	Mutation:58.00%
LLM-EvoOp-M	HV	0.7851	−11.00%	+20.71%	Crossover:11.00%
	IGD	0.0626	+69.65%	−57.56%	Crossover:69.65%
S-NSGA-II	HV	0.6504	−26.27%	–	Both:26.27%
	IGD	0.1475	+299.73%	–	Both:299.73%

3.4 与其他算法的对比分析

为进一步全面评估 LLM-EvoOp 框架的性能, 本研究选取了四种代表性的多目标优化算法 S-NSGA-II(标准 NSGA-II)、MOPSO、MOEA/D 和 GA 进行对比分析. 结果如表 6 所示. 从表中数据可以看出, LLM-EvoOp 在所有测试场景下均取得最优性能, 同时其标准差相对较小, 表现稳定. 值得一提的是, 随着问题规模的扩大, LLM-EvoOp 相对于其他基准算法的性能优势, 即 HV 值的相对差距, 呈现出扩大趋势. 图 3 为五个算法在求解大规模算例时三个目标值的收敛曲线, 可以看出 LLM-EvoOp 得到的非支配解的结果在收敛性上也显著优于其他算法. 这表明 LLM 生成的算子所包含的领域知识在处理更复杂、更大规模问题时能发挥更关键的作用, 证明了 LLM-EvoOp 框架具有良好的可扩展性.

表6 各算法在不同规模测试集上的平均 HV/IGD 性能

规模	指标	LLM-EvoOp	S-NSGA-II	MOEA/D	MOPSO	GA
小	HV	**0.8196 ± 0.0720**	0.6957 ± 0.0483	0.7149 ± 0.0749	0.6684 ± 0.0718	0.4147 ± 0.1242
	IGD	**0.0482 ± 0.0177**	0.0910 ± 0.0237	0.0788 ± 0.0273	0.1128 ± 0.0341	0.2009 ± 0.0764
中	HV	**0.9091 ± 0.0459**	0.7332 ± 0.0475	0.7732 ± 0.0489	0.6804 ± 0.0528	0.7017 ± 0.0560
	IGD	**0.0259 ± 0.0187**	0.1126 ± 0.0292	0.0909 ± 0.0276	0.1402 ± 0.0330	0.1112 ± 0.0324
大	HV	**0.9229 ± 0.0354**	0.7174 ± 0.0438	0.7687 ± 0.0454	0.6640 ± 0.0519	0.7103 ± 0.0545
	IGD	**0.0224 ± 0.0112**	0.1209 ± 0.0210	0.0982 ± 0.0212	0.1474 ± 0.0260	0.1123 ± 0.0238



(a) 不同算法的Makespan收敛曲线 (b) 不同算法的Energy收敛曲线 (c) 不同算法的Load Imbalance收敛曲线

图3 不同算法收敛曲线对比

4 结 论

针对磁悬浮传输系统的动子调度问题, 本文考虑最小化最大完工时间、总能耗和负载不均衡度, 建立了 MTS 动子多目标调度模型. 提出了 LLM-EvoOp 算法框架求解此模型. 该框架将 LLM 从在线求解器转变为离线遗传算子设计器; 基于“思想—代码”二元演化机制, 驱动 LLM 自动生成并优化面向调度问题特性的交叉与变异算子, 将演化生成的算子嵌入 NSGA-II 求解 MTS 动子调度问题. 实验结果表明, 本文提出的 LLM-EvoOp 框架性能优越, 是求解磁悬浮传输系统动子调度问题的有效方法. 研究证实, LLM 驱动的自动设计机制不仅克服了人工设计算子的专家偏差, 更展现出卓越的领域知识捕获能力, 能够生成兼顾效率提升与跳出局部最优的遗传算子, 为解决复杂约束下的多目标调度问题提供了比传统人工设计更具优势的通用范式. 未来将在提示工程、算子泛化能力提升与跨场景迁移等方面继续深入研究.

参考文献 (References)

[1] Xu H S, Liu X, Yu W, et al. Reinforcement learning-based control and networking co-design for industrial Internet of Things[J]. IEEE Journal on Selected Areas in Communications, 2020, 38(5): 885-898.

[2] Nithin J J, Vibin R, Govardhan M A, et al. Magnetic levitation based industrial material handling systems: A futuristic review[C]. Proceedings of the 2nd International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation. Coimbatore, 2023: 223-228.

[3] Xiang B, Wen T, Liu H. Cooperative control of multiple PM actuators in PMLSM with segmented windings[J]. IEEE Transactions on Transportation Electrification, 2024, 10(3): 5570-5580.

[4] Zhao H M, Gu M X, Qiu S P, et al. Dynamic path planning for space-time optimization cooperative tasks of multiple unmanned aerial vehicles in uncertain environment[J]. IEEE Transactions on Consumer Electronics, 2025, 71(3): 7673-7682.

[5] Li Y C, Jing C J. Multi-objective optimization of counter-flow dew-point evaporative coolers for multi-scenario applications using non-dominated sorting genetic algorithm II[J]. Applied Thermal Engineering, 2025, 262: 125292.

[6] Somwong P, Somchit Y. Energy-aware controller load distribution in software-defined networking[C]. International Conference on Electronics, Information, and Communication. Taipei, 2024: 1-4.

[7] Kurtz V, Lin H. Mixed-integer programming for signal temporal logic with fewer binary variables[J]. IEEE Control Systems Letters, 2022, 6: 2635-2640.

[8] 蚁文洁, 刘婷婷, 牛奔. 智能优化算法自动设计综述[J]. 信息与控制, 2025, 54(2): 206-225.
(Yi W J, Liu T T, Niu B. Review of automated design of intelligent optimization algorithm[J]. Information and Control, 2025, 54(2): 206-225.)

[9] Verma S, Pant M, Snasel V. A comprehensive review on NSGA-II for multi-objective combinatorial optimization problems[J]. IEEE Access, 2021, 9: 57757-57791.

[10] Chen D, Zhang J, Wu L H, et al. Optimisation method for semiconductor wafer manufacturing system scheduling: Reinforcement learning with decision graph guiding[J]. Journal of Manufacturing Systems, 2025, 82: 1158-1170.

[11] Li R, Gong W Y, Wang L, et al. Surprisingly popular-based adaptive memetic algorithm for energy-efficient distributed flexible job shop scheduling[J]. IEEE Transactions on Cybernetics, 2023, 53(12): 8013-8023.

[12] Lu C, Zhang B, Gao L, et al. A knowledge-based multiobjective memetic algorithm for green job shop scheduling with variable machining speeds[J]. IEEE Systems Journal, 2022, 16(1): 844-855.

[13] Li R, Wang L, Sang H Y, et al. LLM-assisted automatic memetic algorithm for lot-streaming hybrid job shop scheduling with variable sublots[J]. IEEE Transactions on Evolutionary Computation, 2025: 1.

[14] Wu X Y, Wu S H, Wu J B, 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.

[15] Yang C, Wang X, Lu Y, et al. Large language models as optimizers[C]. Proceedings of the 12th International Conference on Learning Representations. Vienna, 2024: 1-42.

[16] Liu S C, Chen C S, Qu X H, et al. Large language models as evolutionary optimizers[C]. IEEE Congress on Evolutionary Computation. Yokohama, 2024: 1-8.

[17] Wang Y D, Wang J Y, Chu Z W. Multi-agent large language models as evolutionary optimizers for scheduling optimization[J]. Computers & Industrial Engineering, 2025, 206: 111197.

[18] Mahmud D, Hajmohamed H, Almentheri S, et al. Integrating LLMs with ITS: Recent advances, potentials, challenges, and future directions[J]. IEEE Transactions on Intelligent Transportation Systems, 2025, 26(5): 5674-5709.

[19] 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.

[20] Liu F, Tong X L, Yuan M X, et al. Evolution of heuristics: Towards efficient automatic algorithm design using large language model[C]. Proceedings of the 41st International Conference on Machine Learning. Vienna, 2024: 32201-32223.

[21] Ye H G, Xu H, Yan A, et al. Large language model-driven large neighborhood search for large-scale MILP problems[C]. Proceedings of the 42nd International Conference on Machine Learning. Vancouver, Vancouver, 2025: 1-23.

[22] Li K, Liu F, Wang Z K, et al. ARS: Automatic routing solver with large language models[EB/OL]. (2025-02-21)[2025-05-19]. https://arxiv.org/abs/2502.15359.

[23] 陈仁胜, 吴斌, 闫飞一. 基于混合学习策略的可变速AGV与机器绿色集成调度[J]. 控制与决策, 2024, 39(12): 3955-3963.
(Chen R S, Wu B, Yan F Y. Hybrid learning strategy for green integrated scheduling with variable speed AGV[J]. Control and Decision, 2024, 39(12): 3955-3963.)

[24] Xiao K, Dong A F, Mei Z X, et al. Genetic algorithm-based joint task offloading and resource allocation in UAV-enabled vehicular edge computing[C]. Proceedings of the 20th International Conference on Mobility, Sensing and Networking. Harbin, 2024: 346-353.

作者简介

吴斌 (1979−), 男, 教授, 博士, 主要研究方向为智能优化算法、系统建模与优化, E-mail: wubin@njtech.edu.cn;

刘新宇 (2002−), 男, 硕士生, 主要研究方向为生产调度、智能优化算法, E-mail: 202461213198@njtech.edu.cn;

王华 (1978−), 男, 教授, 博士, 博士生导师, 主要研究方向为智能制造、智能运维, E-mail: wanghua@njtech.edu.cn.