国防科技大学学报
Journal of National University of Defense Technology
ISSN 1001-2486,CN 43-1067/T

《国防科技大学学报》网络首发论文

题目：大模型智能体驱动的在线设计成像卫星应急任务调度算法
作者：陈盈果，陈佳威，徐世龙，林翔，王俊琦，周庆瑞
收稿日期：2025-07-27
网络首发日期：2025-12-19
引用格式：陈盈果，陈佳威，徐世龙，林翔，王俊琦，周庆瑞．大模型智能体驱动的在线设计成像卫星应急任务调度算法[J/OL]．国防科技大学学报.
https://link.cnki.net/urlid/43.1067.t.20251218.1743.002

doi: 10.11887//j.issn.1001-2486.25070050

大模型智能体驱动的在线设计成像卫星应急任务调度算法

陈盈果[1]，陈佳威[1*]，徐世龙[1]，林翔[1]，王俊琦[1]，周庆瑞[2]

（1. 国防科技大学 系统工程学院, 湖南 长沙 410073；2. 中国空间技术研究院，北京，100094）

摘　要：为提升成像卫星应对常态化应急任务的调度效率，解决固定调度算法在复杂多变临时场景下性能不足及人工设计算法响应滞后问题，本研究提出一种大模型驱动的智能体协同框架。该框架将复杂调度问题分解为任务分配与单星规划两个子问题，并将算法设计过程拆分为初始算法生成与算法演化优化两个阶段。分解后形成的可执行元任务由多个智能体通过协同沟通自主完成最优算法设计。整个框架实现全自动、智能化运行，支持离线训练与在线无间断部署，保障应急任务的实时响应需求。实验验证表明，该框架自动生成的算法在求解质量与计算效率上均显著优于人类专家设计的算法。

关键词：成像卫星任务规划；大语言模型；优化算法；大模型智能体

中图分类号：C931.9　　**文献标志码：**A

Online design algorithm for emergency mission scheduling of imaging satellites driven by large language model agents

Chen Yingguo[1], Chen Jiawei[1], Xu Shilong[1], Lin Xiang[1], Wang Junqi[1], Zhou Qingrui[2]*

(1. College of Systems Engineering, National University of Defense Technology, Changsha 410073 , China; 2. China Academy of Space Technology, Beijing 100094, China)

Abstract: To enhance the scheduling efficiency of imaging satellites for regular emergency scenarios and address the performance limitations of fixed scheduling algorithms in complex, dynamic situations and the lag associated with manual algorithm design, an LLM (large language model)-driven agent collaboration framework was proposed. The framework decomposed the complex scheduling problem into two subproblems: task allocation and single-satellite planning. The algorithm design process was further divided into two phases: initial algorithm generation and evolutionary algorithm optimization. The resulting executable meta-tasks were autonomously completed by multiple agents through collaborative communication to design the optimal algorithm. The entire framework operated fully automatically and intelligently, supporting offline training and seamless online deployment to ensure real-time response capabilities for emergency tasks. Experimental results demonstrate that algorithms automatically generated by this framework significantly outperform those designed by human experts in both solution quality and computational efficiency.

Keywords: imaging satellite task scheduling; large language models; optimization algorithm; large language model agents

成像卫星任务调度问题作为一个非确定性多项式难度(non-deterministic polynomial-hard，NP-Hard)的组合优化难题，其核心是在复杂约束条件下为观测任务分配最优的资源与时间窗口，以最大化系统综合效益[1,2]。然而，传统离线调度模式假设所有任务信息完全已知，这与实际航天任务环境不符。观测数据显示，现代对地观测系统中约 70%的任务是动态产生的应急需求（如灾害监测、军事侦察等）[3,4]，其显著的时效敏感性与资源冲突性推动了在线调度新范式的发展。在线调度面临三重核心挑战：需在有限计算时间内快速做出决策、应急任务带来的场景多变性，以及新任务与已规划任务间的资源冲突。

该问题的求解主要包括仅适用于小规模问题

收稿日期：2025-07-27

基金项目：国家自然科学基金资助项目（U23B2039）；中国科协青年人才托举工程资助项目（2022QNRC001）

第一作者：陈盈果（1988—），男，重庆人，副教授，博士，硕士生导师，E-mail：ygchen @nudt.edu.cn

* **通信作者：**陈佳威（1996—），男，湖南益阳人，博士研究生，E-mail：cjw@ nudt.edu.cn

引用格式：陈盈果，陈佳威，徐世龙，等. 大模型智能体驱动的在线设计成像卫星应急任务调度算法[J]. 国防科技大学学报,

Citation: CHEN Y G, CHEN J W, SHILONG X L, et al. Online design algorithm for emergency mission scheduling of imaging satellites driven by large language model agents[J]. Journal of National University of Defense Technology,

的精确算法、能在合理时间内提供高质量近似解的元启发式算法[5]，以及易陷入局部最优的机器学习类算法[6-8]。其中，自适应大领域搜索算法(adaptive large neighborhood search，ALNS)通过破坏与修复算子进行高效搜索，并已被扩展至多星协同调度[7,8]。应急任务规划因对时效性、突发性及资源冲突处理的高要求而尤为重要。现有方法主要包括启发式方法[9-11]和基于学习的方法[12-18]。然而，启发式方法常以牺牲常规任务完成度来保障响应速度，且难以应对大规模协同；基于学习的方法则依赖大量训练、耗时长，且对突发场景的泛化能力有限[9-18]。现有方法本质上缺乏在动态环境中实时分析、推理与决策的能力。

为突破这些局限，本文创新性地提出了一种大语言模型(large language model, LLM)驱动的智能体协同框架。该框架充分利用大语言模型强大的问题理解与代码生成能力，实现了算法自主设计。与现有工作相比，其核心优势在于：既保持了传统启发式方法的高效性，又融入了大语言模型的创造性推理能力；既避免了基于学习方法对历史数据的依赖[9-18]，又通过多智能体协同机制实现了在线自适应优化。

具体而言，本文框架采用分层优化策略，将复杂多星调度问题分解为任务分配和单星调度两个子问题，并采用算法设计与演化的两阶段流程。通过智能体间的实时协商与知识共享，实现了分布式协同的全局最优算法设计。此框架实现了从传统程序化调度到认知式调度的根本性转变，为“用自然语言指挥卫星任务”的新型操作模式提供了可行的技术路径。

1 大模型驱动的智能体技术

1.1 大语言模型与智能体技术

大语言模型（LLM）基于 Transformer 架构，经大规模训练，在多项任务中表现出色[19]。为将其通用能力用于解决特定问题，智能体（Agent）技术成为关键路径。智能体通常由感知端、控制端和行动端三部分组成，实现从环境感知到决策执行的闭环[20]。例如，FunSearch[21]将 LLM 与演化算法结合，通过迭代优化展示智能体在算法设计中的协同能力。

1.2 多智能体协同与任务分解

智能体技术指能感知环境、自主决策并完成复杂任务的智能系统[22]。面对复杂任务，多智能体协同更为高效，例如 ChatDev[23]、Hong[24]和 AgentCoder[25]等所示，通过角色分工可构建鲁棒的协同工作流。这种分而治之策略与卫星调度可分解为任务分配加单星调度的双层结构高度契合。因此，本文提出一种新的算法范式，即构建由两个智能体协同驱动的双层优化框架。上层分配智能体动态生成任务分配启发式算子，下层调度智能体生成或优化单星排序的邻域搜索算子。二者通过交互将大语言模型的创造性推理注入算子生成核心，实现与经典双层优化模型的深度融合，形成 LLM 驱动算子生成与双层优化相融合的范式。

该方法与其他现有相关研究相比展现出不同特性。与自适应大领域搜索、FunSearch 等方法的对比，本文的 LLM 驱动算子生成加双层优化范式具备独特的核心机制，由大语言模型理解问题并生成策略，创造出符合逻辑的全新启发式算子，其双层分工与算子直接调用保障了高效率，并能实时动态调整策略与算子，显示出极强的自适应性。

而传统的自适应大领域搜索 ALNS 方法，其核心机制是从固定算子库中依据表现自适应挑选[6]，创新性依赖专家预设而自身无法创造新算子，运行效率上选择机制快速但灵活性受限、效率中等[7]，自适应方面则只能调整选择频率。LLM 算法设计方法通过大语言模型生成代码并迭代进化[26]，能设计出人类未知的、性能更优的数学算法[21]，但其运行需要通过大量演示训练，整体设计任务复杂且大语言模型难以胜任，自适应能力为中等。算法演化方法基于进化策略自动调整算法[27]，虽然可生成新算子但质量随机[28]，搜索过程开销大、效率低，且通常离线生成，自适应过程较弱。自动机器学习 AutoML 方法在预定义空间中进行超参数与结构搜索[29]，创新能力仅限于预定义模型或算子空间的组合，其大规模网格或贝叶斯搜索的计算成本极高、训练周期长，且算法在离线优化后固定，不具备在线调整能力[30]。

综上，本文提出的新范式兼具大语言模型的创新自适应性以及双层分工的高效性，为实现动态复杂的卫星应急调度提供了自动化程度更高的强大工具。

2 问题描述

卫星任务规划问题指在多颗卫星与多个地面任务间安排观测时间与资源，以最大化任务完成效益。该问题涉及可见时间窗口（visible time windows, VTW）及卫星能力、姿态角等约束，是典型的 NP-hard 组合优化问题。本文将问题分解为任务分配和单星调度两个子问题，二者存在层级交互关系。基于 BLO 模型[31]，上层决策任务分配，下层决策单星调度，形成嵌套优化架构。因此，建立 BLO 模型并融合智能体技术与嵌套进化策略以实现协同优化。为应对动态应急场景，构建分批次任务实例：任务分批输入上层分配模块，

再传递至下层调度模块，后者优化每颗卫星的执行方案。

2.1 任务分配

在卫星应急任务规划中，任务分配是第一步，其目标是将任务分配给卫星，为后续单星调度奠定基础。在上层涉及的变量符号如表 1 所示：

表 1 上层模型符号

Tab.1 Notations in the upper-level model

符号	描述
r	任务 $r \in R$
s	卫星 $s \in S$
W_{sr}	卫星 s 与任务 r 之间的可见时间窗口
x_{sr}^{u}	当 $x_{sr}^{u}=1$ 时，表示任务 r 在上层（upper，简称 u）被分配给卫星 s，否则为 0
U_s	表示卫星 s 的总功率。
V_s	表示卫星 s 的存储总量。
u_r	表示观测任务 r 所消耗的卫星功率
v_r	表示观测任务 r 所消耗的存储容量。

2.1.1 决策变量

定义任务分配层的决策变量 x_{sr}^{u}：

$$x_{sr}^{u} \in \{0,1\}, \forall s \in S, \forall r \in R \quad (1)$$

$$x_{sr}^{u}=0, \quad if\ W_{sr}=\varnothing, \forall s \in S, \forall r \in R \quad (2)$$

式（1）中，当 $x_{sr}^{u}=1$ 时，表示任务 r 在上层（upper, 简称 u）被分配给卫星 s。针对每个卫星，存在 $X_s^{u}=\left[x_{s1}^{u},...,x_{sn}^{u}\right]$

式（2）表示若卫星 s 和任务 r 之间不存在可见时间窗口（VTW），则 x_{sr}^{u} 必须为 0。

2.1.2 目标函数

$$\max \sum_{s \in S} f_s \quad (3)$$

式（3）中 f_s 代表下层目标函数返回的值

2.1.3 约束条件

$$\sum_{s \in S} x_{sr}^{u} \le 1, \forall r \in R \quad (4)$$

$$\sum_{t \in T} u_r \cdot x_{sr}^{u} \le U_s, \forall s \in S \quad (5)$$

$$\sum_{t \in T} v_r \cdot x_{sr}^{u} \le V_s, \forall s \in S \quad (6)$$

式（4）表示每个任务至多被分配一次

式（5）（6）表示分配给某颗卫星的任务，其累计功率和存储需求不能超过该卫星的最大容量。

2.2 单星调度

在任务分配完成后，每颗卫星需要根据所分配的任务集合，优化其任务执行顺序与观测时间，确保在满足各类约束的前提下，最大化收益函数。在下层涉及的变量符号如表 2 所示：

表 2 下层模型符号

Tab.2 Notations in the lower-level model

符号	描述
R_s	分配给卫星 s 的任务集合
x_{rw}^{l}	$x_{rw}^{l}=1$ 表示与任务 r 相匹配的卫星 s 在下层（lower，简称 l）选择可见窗口 w 来执行，否则为 0
ot_r	表示观测任务 r 的开始时间
p_r	表示任务 r 的优先级
$\alpha_s^{i_r}$、$\beta_s^{i_r}$、$\gamma_s^{i_r}$	分别表示卫星 s 上的任务 r 在时刻 i 的滚动角、俯仰角和偏航角
b_{rw}	表示任务 r 的第 w 个可见时间窗口的开始时间
e_{rw}	表示任务 r 的第 w 个可见时间窗口的结束时间；
t_{rw}^{b}	表示任务 r 在可见时间窗口 w 内的观测开始时间
t_{rw}^{e}	表示任务 r 在可见时间窗口 w 内的观测结束时间
d_r	表示完成任务 r 所需的成像时间
q_r	表示任务 r 的成像质量，是关于观测开始时间 t_{rw}^{b} 的函数

q_r^{least}	表示任务 r 所需的最低成像质量，可由用户定义
$\delta_{rr'}$	表示两个连续任务 r 和 r' 之间的转换时间
l_{rw}	表示任务 r 的可见时间窗口 w 的时间长度
$y_{rr'}$	表示卫星是否连续观测任务 r 和 r'。当 $y_{rr'}=1$ 时，表明卫星观测完任务 r 后接着观测任务 r'
t_{rw}^{best}	表示在可见时间窗 w 内，任务 r 可获得最佳成像质量的时间

2.2.1 决策变量

单星调度层包含两个决策变量：x_{rw}^{l} 和 ot_r。

$$x_{rw}^{l}\in\{0,1\},\forall w\in W,\forall r\in R \tag{7}$$

式（7）中，当 $x_{rw}^{l}=1$ 表示与任务 r 相匹配的卫星 s 在下层（lower，简称 l）选择可见窗口 w 来执行；

ot_r 代表观测任务 r 的开始时间。

2.2.2 目标函数

单星调度的目标旨在最大化所有被观测任务的总优先级。只有当 $x_{st}^{u}=1$ 时，该任务的优先级才会被计入，即在该双层优化模型中，上层通过完成任务分配后，下层仅能对上层已分配的任务开展具体调度，无法选择未分配任务 。

$$\max f_s=\sum_{r\in R_s}\sum_{w\in W_{sr}}p_r x_{rw}^{l} \tag{8}$$

2.2.3 约束条件

$$\sum_{w\in W_{sr}}x_{rw}^{l}\le 1,\forall r\in R_s,\forall w\in W_{sr} \tag{9}$$

$$b_{rw}\le t_{rw}^{b}\le t_{rw}^{e}\le e_{rw},\forall r\in R_s,\forall w\in W_{sr} \tag{10}$$

$$t_{rw}^{b}+d_r=t_{rw}^{e},\forall r\in R_s,\forall w\in W_{sr} \tag{11}$$

$$t_{rw}^{b}+\delta_{rr'}\le t_{r'w'}^{e},if\ \ y_{rr'}=1,\ \forall w,w'\in W_{sr} \tag{12}$$

$$q_r\ge q_r^{least},\forall r\in R_s \tag{13}$$

$$y_{rr'}\in\{0,1\},\forall r,r'\in R_s \tag{14}$$

$$\delta_{rr'}\ge 0,\forall r,r'\in R_s \tag{15}$$

式（9）确保任务在其对应卫星的所有可见时间窗口中，最多选一个窗口。

式（10）表明任务观测时间应在所选可见窗口内。

式（11）体现了任务观测时间、持续时间和结束时间之间的逻辑关系。

式（12）表明当前任务观测开始时间和上一个任务结束时间的差值必须大于卫星姿态转换时间。

式（13）表明最终成像质量必须满足用户最低成像质量要求。其中图像质量可通过式（16）计算，最佳成像质量的时间可通过式（17）计算：

$$q_r(t_{rw}^{b})=10-9\frac{\left|2t_{rw}^{b}+d_r-2t_{rw}^{best}\right|}{l_{rw}-d_r},\quad if\ \ x_{rw}^{l}=1 \tag{16}$$

$$t_{rw}^{best}=\frac{b_{rw}+e_{rw}}{2} \tag{17}$$

式（14）中 $y_{rr'}$ 是一个二元决策变量，用来描述卫星是否连续观测任务 r 和 r'。当 $y_{rr'}=1$ 时，表明卫星观测完任务 r 后接着观测任务 r'。

式（15）中 $\delta_{rr'}$ 是两个连续任务 r 和 r' 之间的转换时间，$\delta_{rr'}\ge 0$ 确保了转换时间的非负性。转化时间 $\delta_{rr'}$ 具体定义为：

$$\delta_{rr'}=\begin{cases}11.66 & \Delta r\le 10\\ 5+\dfrac{\Delta r}{a_1} & 10<\Delta r\le 30\\ 10+\dfrac{\Delta r}{a_2} & 30<\Delta r\le 60\\ 16+\dfrac{\Delta r}{a_3} & 60<\Delta r\le 90\\ 22+\dfrac{\Delta r}{a_4} & 90<\Delta r\end{cases} \tag{18}$$

其中，$a_1\sim a_4$ 代表四个不同的姿态转换速度其中，$a_1=1.5$，$a_2=2$，$a_3=2.5$，$a_4=3$。Δr 用于衡量卫星 s 的姿态转换角度[7]，可通过式（19）计算：

(19)

$$\Delta r = \left|\alpha_s^{j_{r'}} - \alpha_s^{i_r}\right| + \left|\beta_s^{j_{r'}} - \beta_s^{i_r}\right| + \left|\gamma_s^{j_{r'}} - \gamma_s^{i_r}\right|,$$
$$\forall r, r' \in R, y_{rr'} = 1$$

3 算子进化智能体框架

为解决成像卫星任务规划问题，本研究提出一种层次化分阶段求解框架，将复杂的多星多任务调度分解为任务分配与单星规划两个子问题。算法设计相应地分为初始方案生成与演化优化两个阶段。后者利用演化计算与智能体协同技术对方案进行迭代改进，通过智能体间的协商与信息共享确保全局适应性。

该框架以自适应大领域搜索 ALNS 算法为核心，其已被证明在该领域具有优越性[8]。ALNS 算法的关键在于其算子设计，包括宏观协同的任务分配算子以及专注于单星优化的破坏与修复算子，这些算子的协同能有效处理联合优化问题。

鉴于设计高效算子至关重要，本研究进一步提出了智能体的算子演化框架以实现智能化设计。其核心创新在于离线使用大语言模型 LLM 进行算法开发并固化为算子库，在线调度时直接调用，从而通过离线训练实现快速在线部署。为降低复杂度，开发任务被分解为设计与演化两步。整个演化流程包含两个主要循环，先固定下层算法优化上层算法至收敛，再固定上层算法优化下层算法，最终通过智能体协同生成全局最优的双层算法，如图 1 和算法 1 所示。

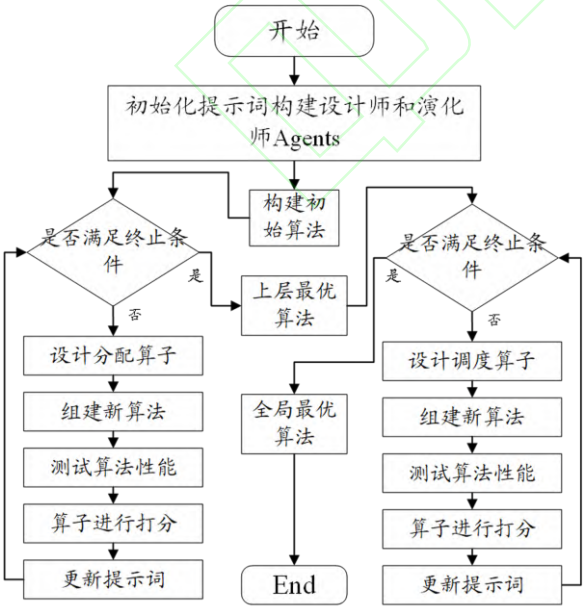


图 1 大语言模型智能体驱动框架流程图

Fig.1 Flowchart of LLM agent driving framework

算法 1 大语言模型智能体驱动的卫星调度算法设计框架

Alg.1 Design framework for satellite scheduling algorithm driven by large model agents

输入：场景实例数据 D_s，初始算子群 $P_0=(P_0^u, P_0^l)$ 和算法 A_0，设计提示词 I_a 与算子设计师 AD，演化提示词 I_e 与算子演化师 AE。

输出：最佳算子群 P^*，最佳算法 A^*，最优调度方案的收益 S^*。

设计初始算子群 P_0 和算法 A_0；

使用提示词训练 LLM 驱动的智能体，$AD \leftarrow I_a$，$AE \leftarrow I_e$；

测试初始算子群 $P_0=(P_0^u, P_0^l)$ 和算法 A_0 并获得方案收益 S_0；

设置两端的最优算子群 P_0^u

定义最优算子群 $P^*=P_0$（$P_0^{u*}=P_0^u$，$P_0^{l*}=P_0^l$），$A^*=A_0$，$S^*=S_0$，

设置最大分配算子的演化次数 $t_{\max}^u$；

for (i=0; i<$t_{\max}^u$; i++) **do**

使用 AD 针对任务分配问题设计算子群 P_i^u；

使用算子群更新算法 $A_i^u \leftarrow P_i^u$

测试算法性能并获取收益 $S_i \leftarrow A_i^u$

使用 AE 对各算子进行打分 P_i^u 并重构提示词 I_a 和 I_e；

if $S_i > S^*$ **do**

$P^*=P_i$（$P_i^{u*}=P_i^u, P_i^{l*}$），$A^*=A_i$，$S^*=S_i$，

endif

重新训练智能体，$AD \leftarrow I_a$，$AE \leftarrow I_e$；

endfor

设置最大调度算子的演化次数 $t_{\max}^l$；

for (j=0; j<$t_{\max}^l$; j++) **do**

使用 AD 针对单星调度问题设计算子群 P_j^l；

使用算子群更新算法 $A_i^l \leftarrow P_i^l$

测试算法性能并获取收益 $S_i \leftarrow A_i^l$

使用 AE 对各算子进行打分 P_j^l 并重构提示词 I_a 和 I_e；

if $S_j > S^*$ **do**

$P^*=P_j$（$P_j^{u*}=P_j^u, P_j^{l*}$），$A^*=A_j$，$S^*=S_j$，

endif

重新训练智能体，$AD \leftarrow I_a$，$AE \leftarrow I_e$；

endfor

return 最佳算子群 P^*，最佳算法 A^*，最优调度方案的收益 S^*；

在进入演化之前，需要对基于 LLM 构建的两个智能体进行提示词训练。在设置了最大演化次数之后，分别对任务分配和单星调度算子进行演化，其演化过程由两个智能体共同完成。当算法的收益获得提升时，更新最优化算法，直至达到

最大演化次数。

3.1 算子演化智能体

本文提出了两个大语言模型驱动的智能体：算子设计师与算子演化师。设计师负责理解问题与初始算子，进而创新设计高效的新算子；演化师则负责评估已有算子的性能，并对其进行微调与优化。

为明确二者职责并确保任务高效执行，本研究采用了角色分化的提示词工程方法。具体而言，为设计算子设计师，构建了一种结构化的提示词框架。该框架如图 2 所示，包含问题描述、算子代码、代数解释、支持类以及具体的设计任务与要求。通过这一完整的提示结构，算子设计师能够有效依据已有算子，创新生成性能更优的新算子。

算子设计师

 # 任务描述：
我的主要目标是高效地解决多敏捷地球观测卫星（多AEOS）调度问题。我的想法是首先将任务调度到卫星，然后使用单敏捷地球观测卫星（单AEOS）调度算法来规划每颗卫星的调度任务。最后，将每个单AEOS的调度方案组合起来，形成多AEOS调度问题的解决方案。当前的挑战是设计一种高效的任务调度算子（启发式算法），以便将任务合理地分配给每颗卫星。

 # 算子代码
```
def dispatch_heuristic(self):
    task_dispatchings = [[] for _ in range(self.sat_num)]
    undispatched_tasks = list(self.candidate_task_ids)
    satellite_load = [0] * self.sat_num

    while undispatched_tasks:
        task_id = undispatched_tasks.pop(0)
        task_info = self.task_lists[0][task_id]
        best_sat_id = None
        best_score = -1
    … …
```

 # 代数解释：
- self.task_lists (List[List[Task]]):此列表以两层结构化形式表示不同的卫星及其关联的任务实例。外层列表的每个元素对应一个卫星，而内层列表包含该特定卫星的所有任务实例。例如，考虑以下列表：[[Task instance 1 of Sat 1, Task instance 2 of Sat 1],[Task instance 1 of Sat 2, Task instance 2 of Sat 2, Task instance 3 of Sat 2]].这些任务实例由"支持类"部分中定义的 Task 类封装。
- self.candidate_task_ids (List(int)):待调度任务的索引列表。
- self.sat_num (int):卫星数量。
- ……

 # 支持类：
- ```
```
class Task:
    index: int = index  # ID of a task
    vtw_list: List[VTW] = vtw_list  # List of VTWs for a task
    priority: int = priority  # Task priority
    task_dtime: float = task_dtime  # Task duration
```
- ```
```
class VTW:
    begin_time: List = bt  # Begin time: [year, month, day, hour, minute, second]
    end_time: List = et  # End time
    self.vtw_dtime: int = dt  # Duration of the VTW
    self.attitude_angle_list:        List[AttitudeAngle]        = attitude_angle_list  # A list of 'AttitudeAngle' objects
```
-……

 # 设计任务：
- 您必须逐步阅读给定的问题描述、变量解释和支持类部分，并彻底了解问题背景和变量的类型。
- 您的任务是协助算子演化师充分利用我提供的信息，特别是任务、卫星、可见时间窗的信息，创新地设计新的分配和调度算子。

 # 任务要求：
- 你的 Python 函数必须保留与代码例子相同的风格方法、函数名称、变量名称、参数和返回值。
- 新引入的参数、变量、第三方库必须及时赋值适当的值，认真定义，并正确导入。
- 如果你定义了辅助函数，请确保它们在"dispatching_heuristic"中定义。
- 只需反馈新设计的调度启发式代码，无需任何其他附加输出。

图 2 算子设计师提示词

Fig.2 Prompts for algorithm designer

为明确智能体职责，本研究采用分化的提示词工程方法。如图 2 所示，算子设计师的提示词结构包含完整的问题描述、算子代码、代数解释与支持类，旨在使其基于已有算子创新设计出更优的新算子。

对于算子演化师，其提示词结构如图 3 所示，核心在于演化任务与要求。与设计师不同，演化师无需深刻理解整个问题，其职责是理解算子、调用程序测试算法性能，并依据结果改进算子群，具体操作为淘汰低适应度算子并保留高适应度算子。

算子演化师

 # 任务描述：
我的主要目标是测试由新算子组合而成的新算法，并根据测试的收益结果，并设计新一轮的算子，将新算法提供给算子设计师，共同构建新算法。

 # 算子代码
```
def dispatch_heuristic(self):
"""
Functionality:此功能通过整合任务优先级、卫星负载均衡和姿态兼容性，向卫星分配任务。它首先根据优先级对任务进行排序，然后将每个任务分配给负载均衡程度最低且姿态兼容性最高的卫星。如果未找到兼容卫星，则恢复随机分配。
"""
{code…}
… …
```

 # 演化任务：
- 你必须阅读所有的算子解释与代码，并根据算法的计算结果与迭代的自适应值对算子进行打分。
- 你的任务是调用程序进行代码运行，并打分进行算子的优化，进而设计出新的算子，为在下一轮演化中给算子设计师提供指导。

 # 任务要求：
- 在算子打分中，自适应度越高，评分最高。
- 只能在已有规则和代码上改进算子，不能完全设计新算子。
- 可以将两种高评分的算子进行结合，并放弃低评分的算子。
- 只需反馈新设计的调度启发式代码，无需任何其他附加输出。

图 3 算子演化师提示词

Fig.3 Prompts for algorithm evolutionist

3.2 智能体框架工作流

为实现算子在线协同演化，本文设计了智能体协作框架。如图 4 所示，该框架包含完整的自适应大邻域搜索流程与智能体协同演化机制。

图 4 上半部分描绘算法执行流程。在分配层，算法从初始解开始，通过轮盘赌机制动态选择分配算子以调整任务分配。随后在调度层，以相同机制选取调度算子进行局部优化，生成新解。算法根据算子性能自适应调整其选择概率，奖励优异算子，削弱低效算子。

图 4 下半部分阐释了智能体协同演化过程。算子设计智能体基于对问题与现有算子的理解，创新设计新算子并注入种群。新算子经实例化测试后，其性能被精确评估。算子演化智能体则依据适应度得分等指标综合评价算子，并将结果反馈至下一轮提示中。通过多轮迭代，该框架能自主演化出针对特定场景的高性能算法配置。

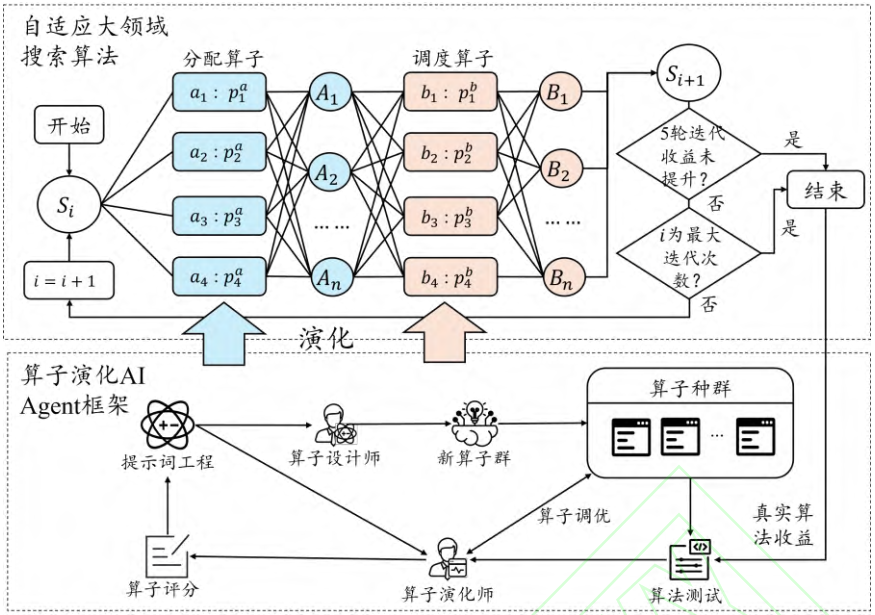


图 4 大语言模型智能体协同算法设计框架展示

Fig.4 LLM agents collaborative algorithm design framework display

4 仿真场景实验

为验证所提算子进化智能体框架的有效性，本文设计了不同任务规模下的对比实验。实验数据来源于“天智杯人工智能挑战赛”开放平台(https://tianzhibei.com/regularHomepage)，其数据公开可下载。为保障结果准确与公平，所有方案的最终收益均使用官方工具，依据第三节所述的优化目标进行计算。

4.1 仿真场景构建

为全面验证智能体演化算法在不同规模应急场景下的性能，本文构建了共计 9 个仿真场景。所有场景规划周期均为 24 小时（0 时至 24 时），期间每隔 2 小时到达一批应急任务，每个任务收益设定在 5 至 10 之间。场景依据静态与应急任务的数量及卫星资源规模，划分为低、中、高三个难度等级，每个等级包含 3 个具体场景。

低难度场景（编号 1-3）包含 180 个静态点目标与 10 个静态区域目标，并需处理 25 个应急点目标与 5 个应急区域目标，调度卫星数量为 30 颗。

中难度场景（编号 4-6）的任务量显著增加，包含 540 个静态点目标与 20 个静态区域目标，应急任务增至 80 个点目标与 10 个区域目标，用于调度的卫星数量为 80 颗。

高难度场景（编号 7-9）具有最大的任务与资源规模，包含 1620 个静态点目标与 50 个静态区域目标，需处理 150 个应急点目标与 15 个应急区域目标，参与调度的卫星数量达到 200 颗。

4.2 实验设置

为检验智能体演化算法的性能，本文选取大邻域搜索算法、模拟退火算法、遗传算法及贪婪爬山算法作为对比。所有算法均采用 Java 1.8 实现，使用的大语言模型包括 DeepSeek-V3（671B）以及用于对比的 GPT-4o（300B）、GPT-3.5 Turbo（175B）与文言一星（260B）。各模型均采用官方基础版本，温度设为 0.7 并禁用 Top-p 采样。实验在统一环境中进行：Intel Core I9-13900H CPU 2.0 GHz、Windows 11 操作系统及 32GB 内存。算法在达到指定迭代次数或最大未改进次数后终止。

4.3 性能对比实验

为全面评估 BLO-ALNS 算法的有效性，本文将其与四种基线算法进行对比。各算法参数设置如下：ALNS 采用 5 个破坏与 5 个修复算子，迭代 1000 次；模拟退火初始温度为 1000，降温系数 0.95；遗传算法种群规模为 100，交叉概率 0.8；贪婪爬山最大未改进次数为 100；BLO-ALNS 则设置上层迭代 200 次，下层迭代 500 次。

评估从运行时间与算法收益两个维度进行，结果汇总于表 3。表中外部数值为 24 小时总收益，括号内为每时域平均计算时间。对比显示，在不同规模与难度的场景中，BLO-ALNS 均取得最优性能。相较于基础 ALNS，它在收益上显著提升而计算时间增长有限，验证了 BLO 机制的有效性。

与各类基线算法相比，BLO-ALNS 也始终表现更优。

图 5 进一步展示了不同规模场景下任务收益随时间演变趋势。在所有滚动规划周期内，BLO-ALNS 的累计收益均优于其他算法，且随周期推进持续增长。与各算例最优算法相比，本框架平均提升收益 10.48%。这得益于新应急任务被高效纳入调度并完成，体现了算法在动态环境中的优势。双层优化框架支持上层利用大模型进行高质量全局决策，进一步验证了算法的有效性与鲁棒性。

表 3　算法收益对比结果

Tab.3 Algorithm profits comparison results

算例	BLO—ALNS	ALNS	SA	GA	Greedy
1	**1420.20（6.3s）**	1245.45**(6.3s)**	1231.25(10.3s)	1194.38 (90s)	831.12(8.3s)
2	**1415.57（8.4s）**	1158.34 (8.5s)	1108.44 (11.8s)	1181.90 (120s)	808.64 (10.8s)
3	**1390.77（6.7s）**	1235.12(8.8s)	1235.12(10.9s)	1269.26 (100s)	935.16(11.9s)
4	**4220.98（45.4s）**	4012.15 (45.6s)	3901.15 (60.6s)	4157.18 (260s)	2901.19 (35.6s)
5	**4190.12（54.8s）**	4088.78 (44.9s)	3845.89 (64.5s)	4044.56 (380s)	2845.38 (37.5s)
6	**4201.34（49.1s）**	4165.72(43.6s)	3965.72(65.4s)	3931.34 (520s)	2965.62(50.4s)
7	**10009.56**（85.8s）	8542.90 **(85.4s)**	7742.90(100.4s)	9618.47(700s)	6742.97(89.4s)
8	**11879.55（87.5s）**	9618.43(88.3s)	8619.63(110.4s)	9705.29(920s)	7619.67(90.4s)
9	**11425.19**（91.9s）	10695.76**(89.5s)**	8665.46(130.5s)	8792.50(1100s)	7665.46(95.5s)

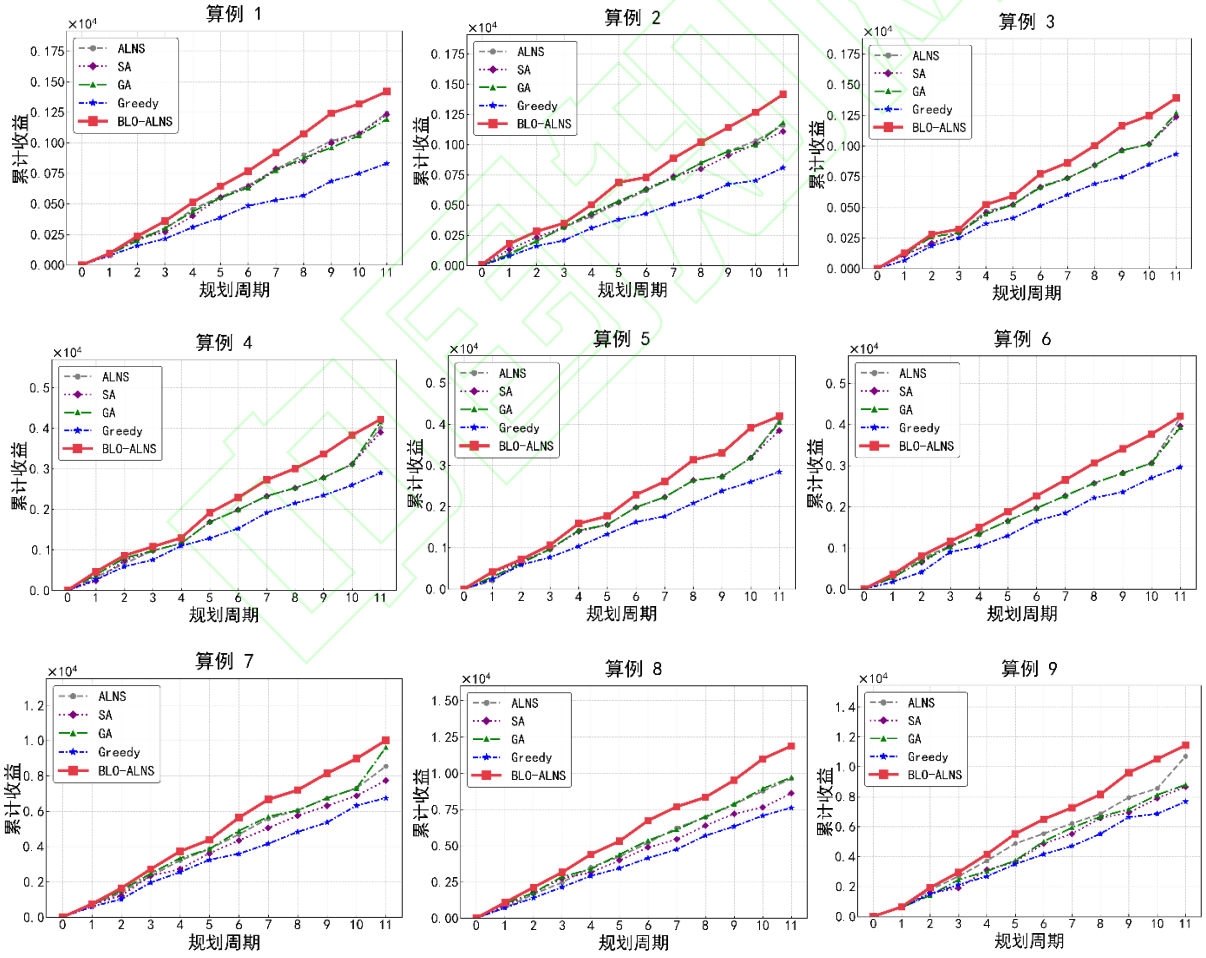


图 5 算法迭代图

Fig. 5 Algorithm iteration chart

4.4 消融实验

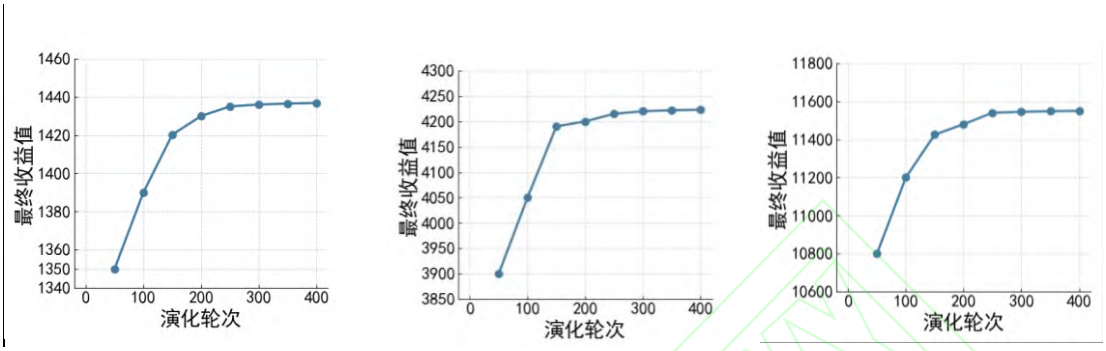
为分析各组件影响，本文通过消融实验探讨了 BLO 组件、智能体拆分、算子库规模与演化轮

次等因素对系统性能的作用，同时评估了评分策略与不同大语言模型替换的效果。

图 6 显示，随着演化轮次增加，各算例收益均先快速上升，在轮次达 200 左右后提升减缓并趋于平稳，表明此前增加轮次能有效提升收益。

图 7 表明，算子库规模扩大初期能显著提高收益，当规模达到 20 左右时，收益增长进入平台期，后续扩大作用有限。

图 8 展示了计算时间与收益的关系。整体上，收益随计算时间增加而上升，大规模场景收益增长更为明显。例如在算例 9 中，计算时间由 0 增至约 175 秒时，收益从 1000 快速上升至 12000 以上。这表明延长计算时间能有效提升收益，且大规模场景更具增长潜力。

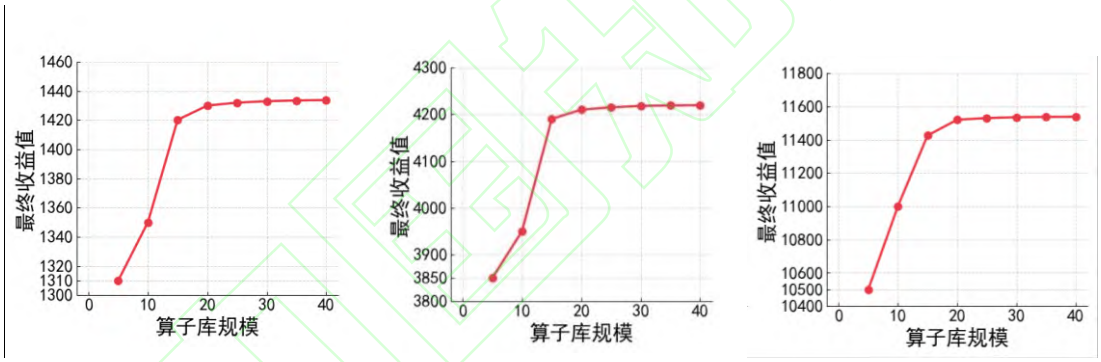


（a）算例 1　　（b）算例 5　　（c）算例 9

（a）Example 1　　（b）Example 5　　（c）Example 9

图 6 不同轮次与收益的影响图

Fig. 6 Impact diagram of different rounds and profits



（a）算例 1　　（b）算例 5　　（c）算例 9

（a）Example 1　　（b）Example 5　　（c）Example 9

图 7 不同算子库规模与收益的影响图

Fig. 7 Impact diagram of different operator library sizes and profits

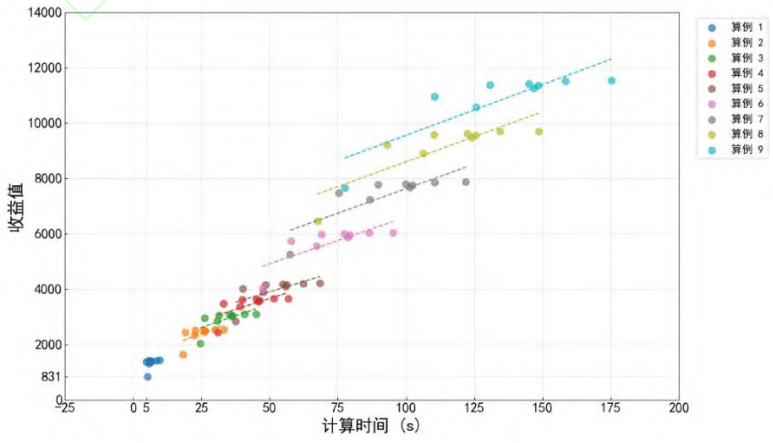


图 8 时间与收益的关系图

Fig. 8 Graph of the relationship between time and profits

5 结论

本研究针对成像卫星常态化应急调度场景，创新性提出大模型驱动的多智能体协同决策框架，实现全流程智能化算法设计与实时响应。核心突破在于：首创“问题分解-算法进化”双循环机制——将复杂调度自主拆解为任务分配与单星规划子问题，并通过两阶段优化流程（初始生成与演化微调）动态生成高性能调度算法。

基于智能体间实时协商与知识共享，构建分布式协同决策网络。该框架成功实现三大能力跃升：1）全链路无人干预的调度方案自动生成；2）离线训练-在线部署的协同演化机制；3）对突发任务的秒级响应能力。实验证实，本框架所生成算法在求解质量（+10.48%）与计算效率上均显著超越人类专家设计基准，为智能卫星自主任务规划提供全自动化智能化开发的解决方案。

参考文献（References）

[1] WU J, CHEN Y N, HE Y M, et al. Survey on autonomous task scheduling technology for Earth observation satellites[J]. Journal of Systems Engineering and Electronics, 2022, 33(6): 1176-1189.

[2] 贺仁杰, 高鹏, 白保存, 等. 成像卫星任务规划模型、算法及其应用[J]. 系统工程理论与实践, 2011, 31(3): 411-422.
HE R J, GAO P, BAI B C, et al. Models, algorithms and applications to the mission planning system of imaging satellites[J]. Systems Engineering—Theory & Practice, 2011, 31(3): 411-422.(in Chinese)

[3] 王静巧, 杨磊, 庄超然, 等. 面向应急事件的卫星任务规划技术[J]. 航天返回与遥感, 2022, 43(3): 105-112.
WANG J Q, YANG L, ZHUANG C R, et al. Emergency-oriented satellite mission planning[J]. Spacecraft Recovery & Remote Sensing, 2022, 43(3): 105-112. (in Chinese)

[4] 杜永浩, 邢立宁, 姚锋, 等. 航天器任务调度模型、算法与通用求解技术综述[J]. 自动化学报, 2021, 47(12): 2715-2741.
DU Y H, XING L N, YAO F, et al. Survey on models, algorithms and general techniques for spacecraft mission scheduling[J]. Acta Automatica Sinica, 2021, 47(12): 2715-2741. (in Chinese)

[5] NIU X N, TANG H, WU L X. Satellite scheduling of large areal tasks for rapid response to natural disaster using a multi-objective genetic algorithm[J]. International Journal of Disaster Risk Reduction, 2018, 28: 813-825.

[6] LIU X L, LAPORTE G, CHEN Y W, et al. An adaptive large neighborhood search metaheuristic for agile satellite scheduling with time-dependent transition time[J]. Computers & Operations Research, 2017, 86: 41-53.

[7] HE L, LIU X L, LAPORTE G, et al. An improved adaptive large neighborhood search algorithm for multiple agile satellites scheduling[J]. Computers & Operations Research, 2018, 100: 12-25.

[8] LI L, DU Y H, YAO F, et al. Learning memetic algorithm based on variable population and neighborhood for multi-complex target scheduling of large-scale imaging satellites[J]. Swarm and Evolutionary Computation, 2025, 92: 101789.

[9] WANG J J, ZHU X M, YANG L T, et al. Towards dynamic real-time scheduling for multiple earth observation satellites[J]. Journal of Computer and System Sciences, 2015, 81(1): 110-124.

[10] WANG J J, ZHU X M, QIU D S, et al. Dynamic scheduling for emergency tasks on distributed imaging satellites with task merging[J]. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(9): 2275-2285.

[11] 李超. 数据驱动的敏捷卫星自主调度方法研究[D]. 长沙: 国防科学技术大学, 2017: 1-25.
LI C. Data-driven autonomous scheduling of agile satellites[D]. Changsha: National University of Defense Technology, 2017: 1-25.(in Chinese)

[12] CHUN J, YANG W Y, LIU X L, et al. Deep reinforcement learning for the agile earth observation satellite scheduling problem[J]. Mathematics, 2023, 11(19): 4059.

[13] HERRMANN A, SCHAUB H. Reinforcement learning for the agile earth-observing satellite scheduling problem[J]. IEEE Transactions on Aerospace and Electronic Systems, 2023, 59(5): 5235-5247.

[14] 陈名. 基于深度强化学习的敏捷卫星调度方法研究[D]. 长沙: 国防科技大学, 2020.
CHEN M. Research on agile satellite scheduling method based on deep reinforcement learning[D]. Changsha: National University of Defense Technology, 2020. (in Chinese)

[15] 王海蛟. 基于强化学习的卫星规模化在线调度方法研究[D]. 北京: 中国科学院国家空间科学中心, 2018.
WANG H J. Massive scheduling method under online situation for satellites based on reinforcement learning[D]. Beijing: National Space Science Center, Chinese Academy of Sciences, 2018. (in Chinese)

[16] 邢立宁, 王原, 何永明, 等. 基于 BP 神经网络的星上任务可调度性预测方法[J]. 中国管理科学, 2015, 23(增刊 1): 117-124.
XING L N, WANG Y, HE Y M, et al. An earth observation satellite task schedulability prediction method based on BP artificial network[J]. Chinese Journal of Management Science, 2015, 23(Suppl 1): 117-

124. (in Chinese)

[17] 刘嵩, 白国庆, 陈英武. 地球观测网络成像任务可调度性预测方法[J]. 宇航学报, 2015, 36(5): 583-588.
LIU S, BAI G Q, CHEN Y W. Prediction method for imaging task schedulability of earth observation network[J]. Journal of Astronautics, 2015, 36(5): 583-588. (in Chinese)

[18] WU J, YAO F, SONG Y J, et al. Frequent pattern-based parallel search approach for time-dependent agile earth observation satellite scheduling[J]. Information Sciences, 2023, 636: 118924.

[19] 秦小林, 古徐, 李弟诚, 等. 大语言模型综述与展望[J]. 计算机应用, 2025, 45(3): 685-696.
QIN X L, GU X, LI D C, et al. Survey and prospect of large language models[J]. Journal of Computer Applications, 2025, 45(3): 685-696.

[20] WANG L, MA C, FENG X Y, et al. A survey on large language model based autonomous agents[J]. Frontiers of Computer Science, 2024, 18(6): 186345.

[21] ROMERA-PAREDES B, BAREKATAIN M, NOVIKOV A, et al. Mathematical discoveries from program search with large language models[J]. Nature, 2024, 625: 468-475.

[22] CASTELFRANCHI C. Modelling social action for AI agents[J]. Artificial Intelligence, 1998, 103(1/2): 157-182.

[23] QIAN C, CONG X, LIU W, et al. Communicative agents for software development[EB/OL]. [2025-05-01]. 2023.https://openreview.net/pdf?id=yW0AZ5wPji.

[24] HONG S R, ZHUGE M C, CHEN J Q, et al. MetaGPT: meta programming for a multi-agent collaborative framework[EB/OL]. (2024-11-01)[2025-05-01]. https://arxiv.org/abs/2308.00352.

[25] HUANG D, ZHANG J M, LUCK M, et al. AgentCoder: multi-agent-based code generation with iterative testing and optimisation[EB/OL]. (2024-05-24)[2025-05-01]. https://arxiv.org/abs/2312.13010.

[26] LIU F, TONG X L, YUAN M X, et al. Evolution of heuristics: towards efficient automatic algorithm design using large language model[EB/OL]. (2024-06-01)[2025-05-01]. https://arxiv.org/abs/2401.02051.

[27] TANG K, LIU S C, YANG P, et al. Few-shots parallel algorithm portfolio construction via co-evolution[J]. IEEE Transactions on Evolutionary Computation, 2021, 25(3): 595-607.

[28] LIU S C, TANG K, YAO X. Generative adversarial construction of parallel portfolios[J]. IEEE Transactions on Cybernetics, 2022, 52(2): 784-795.

[29] SHEN Z Q, ZHANG Y Q, WEI L N, et al. Automated machine learning: from principles to practices[EB/OL]. (2024-02-27)[2025-05-01]. https://arxiv.org/abs/1810.13306.

[30] HE X, ZHAO K Y, CHU X W. AutoML: a survey of the state-of-the-art[J]. Knowledge-Based Systems, 2021, 212: 106622.

[31] BRACKEN J, MCGILL J T. Mathematical programs with optimization problems in the constraints[J]. Operations Research, 1973, 21(1): 37-44.