控制与决策
Control and Decision

基于分布式元强化学习的多敏捷卫星多目标调度算法

张广辉, 魏晨轩, 冯彦翔, 李晓玲

引用本文:
张广辉, 魏晨轩, 冯彦翔, 等. 基于分布式元强化学习的多敏捷卫星多目标调度算法[J]. 控制与决策, 2026, 41(4): 1065-1076.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2025.0112

基于分布式元强化学习的多敏捷卫星多目标调度算法

张广辉[1†]，魏晨轩[1]，冯彦翔[2]，李晓玲[3]

(1. 河北农业大学 信息科学与技术学院，河北 保定 071001；2. 西安交通大学 自动化科学与工程学院，西安 710049；3. 长安大学 电子与控制工程学院，西安 710064)

摘　要: 随着地球观测领域相关技术的高速发展, 近年来具有先进姿态调整能力的敏捷地球观测卫星已经引起了广泛的关注. 敏捷卫星任务调度具有时间依赖性切换时间, 在多星、多轨道、多需求的卫星观测场景下, 产生了复杂的时间依赖性多敏捷卫星多目标调度问题. 针对该问题, 首先, 基于问题特征和优化目标建立问题的数学规划模型; 其次, 提出一种分布式元Q学习协同进化框架, 包括预训练和进化搜索两个阶段, 预训练阶段通过分布式Q学习提高训练效率, 进化搜索阶段通过训练好的分布式Q学习模型实现多种群进化算子的自适应选择; 然后, 基于所提出的进化框架和问题特征, 设计多样化的进化算子和动态种群划分选择策略, 建立一种分布式元Q学习协同进化算法 (DMCEA); 最后通过实验验证 DMCEA 求解问题的有效性.

关键词: 敏捷卫星；时间依赖性；任务调度；多目标调度；协同进化算法；分布式强化学习

中图分类号: V19　　**文献标志码:** A

DOI: 10.13195/j.kzyjc.2025.0112

引用格式: 张广辉, 魏晨轩, 冯彦翔, 等. 基于分布式元强化学习的多敏捷卫星多目标调度算法 [J]. 控制与决策, 2026, 41(4): 1065-1076.

Multiple agile satellites multi-objective scheduling algorithm based on distributed meta-reinforcement learning

ZHANG Guang-hui[1†]，*WEI Chen-xuan*[1]，*FENG Yan-xiang*[2]，*LI Xiao-ling*[3]

(1. School of Information Science and Technology，Hebei Agricultural University，Baoding 071001，China；2. School of Automation Science and Engineering，Xi'an Jiaotong University，Xi'an 710049，China；3. School of Electronics and Control Engineering，Chang'an University，Xi'an 710064，China)

Abstract: With the rapid advancement of earth observation technologies in recent years, agile earth observation satellites (AEOS) equipped with advanced attitude adjustment capabilities have attracted widespread attention. The scheduling of AEOS involves time-dependent transition times, which generates a complex time-dependent multi-objective scheduling problem in multi-satellite, multi-orbit, and multi-demand observation scenarios. To solve this problem, we first formulate a mathematical programming model based on the problem characteristics and optimization objectives. Subsequently, a distributed meta-Q-learning co-evolutionary framework is proposed, which consists of a pretraining phase and an evolutionary search phase. In the pretraining phase, distributed Q-learning is used to enhance the pretraining efficiency, while the evolutionary search phase leverages the pretrained distributed Q-learning model to adaptively select the evolutionary operators. Based on the proposed evolutionary framework and problem characteristics, different evolutionary operators and a dynamic population division strategy are designed. Further, a distributed meta-Q-learning co-evolutionary algorithm (DMCEA) is constructed. Finally, computational experiments validate the effectiveness of the DMCEA in solving the problem under consideration.

Keywords: agile satellites；time-dependence；task scheduling；multi-objective scheduling；co-evolutionary algorithm；distributed reinforcement learning

0　引　言

地球观测卫星 (EOS) 是利用星载遥感器对地球表面和低层大气进行探测以获取有关信息的一类卫星, 其主要任务是对地面特定目标进行成像, 并将所

收稿日期: 2025-01-22；**录用日期:** 2025-04-17.

基金项目: 河北省自然科学基金项目 (F2024204007)；西安交通大学机械制造系统工程国家重点实验室开放课题 (sklms2023002).

责任编委: 王凌.

[†]通信作者. E-mail: ghzhang@hebau.edu.cn.

获得的数据实时传输至地面站[1]. EOS 由于具有覆盖范围广、观测持续时间长、不受空域国界限制等优点, 在天气预报、灾害监测、自然资源勘探和军事侦察等领域应用非常广泛[2]. 随着卫星机动和成像能力的不断提升, 敏捷地球观测卫星 (AEOS) 作为新一代地球观测工具, 其在俯仰、侧摆、偏航 3 个轴向具有先进姿态调整能力. 与仅具有侧摆轴向的非敏捷地球观测卫星相比, AEOS 能够在卫星经过目标点之前或之后进行观测, 具有更长的可见时间窗口 (VTW) 执行观测任务, 可以满足更多的观测需求, 如图 1 所示.

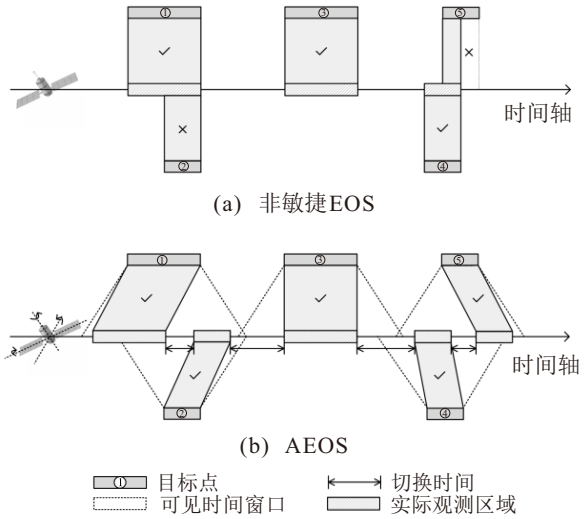

图1 非敏捷 EOS 与 AEOS

随着观测任务需求的不断增加, 现有的观测卫星数量难以满足实际应用需求. AEOS 具有的灵活姿态调整能力能够大幅提高工作效能, 也导致了更复杂的任务 VTW 冲突, 形成了敏捷地球观测卫星调度问题 (AEOSSP). 在此背景下, 开发高效的调度算法是提高卫星观测效率的关键, 对于充分发挥新型卫星平台的观测优势具有重要的理论价值与工程实践意义. 针对 AEOSSP, 国内外学者已经开展了广泛的研究. Lemaître 等[3] 首先证明了 AEOSSP 是一个 NP 难问题, 建立了相应的整数规划模型, 并设计贪婪算法、局部搜索算法、动态规划算法和约束规划算法共 4 种启发式方法处理该问题, 然而这项工作忽视了 AEOSSP 中切换时间的时间依赖特性.

AEOS 对相邻任务进行连续观测时, 需要一个切换时间来完成不同目标点的过渡, 切换时间的长短取决于卫星 3 个轴向的角度变化幅度大小, 变化幅度由两次观测的开始时间决定, 从而产生了时间依赖性的切换时间约束, 许多学者对带有时间依赖性的 AEOSSP 进行了研究. Liu 等[4] 在单星场景下设计了一种自适应大邻域搜索算法处理此问题; Peng 等[5] 在单星和多星场景下设计了贪婪随机迭代求解算法求解此问题; Wu 等[6] 提出基于频繁模式的并行搜索算法对 AEOSSP 进行求解. 上述工作考虑了 AEOSSP 的时间依赖性, 并将 AEOSSP 视为最大化观测收益的单目标问题, 但忽视了决策者的不同偏好和 AEOSSP 的其他相关约束与实际需求, 难以反映卫星实际调度中的动态复杂性.

为了更加切合卫星的应用场景, 许多学者对多目标敏捷地球观测卫星调度问题 (MO-AEOSSP) 展开了进一步研究. Wei 等[7] 以最小化观测任务失败率和负载均衡为目标, 对单星场景下的 MO-AEOSSP 构造了多目标模因算法, 并设计了不同的搜索算子; Wang 等[8] 为多星场景下的 MO-AEOSSP 提出了一种多目标模因离散 Jaya 算法, 并设计不同的目标改进算子用于求解该问题; Chang 等[9] 提出了一种基于自适应大领域搜索算法和非支配排序遗传算法 II 的多目标优化算法, 用来解决具有综合任务聚类的 MO-AEOSSP; Wang 等[10] 构造了以总任务收益、卫星能耗和负载均衡为目标的 MO-AEOSSP, 并采用策略融合的多目标蜣螂优化算法求解该问题; Huang 等[11] 提出了基于强化学习的蜣螂优化算法, 通过自适应优化方法解决 MO-AEOSSP. 上述算法能够有效处理 MO-AEOSSP, 但并未充分利用任务属性、可见时间窗属性和卫星能耗等启发式信息, 算法求解性能仍存在提升空间.

本文针对多星场景下带有时间依赖性的 MO-AEOSSP, 提出一种新的分布式元Q学习协同进化算法 (DMCEA), 主要创新点如下:

1) 针对多星场景下的 MO-AEOSSP 问题, 构建 MO-AEOSSP 数学模型, 并对该模型采用 Cplex 求解.

2) 提出一种分布式元Q学习协同进化框架, 包括预训练和进化搜索两个阶段. 预训练阶段采用分布式Q学习机制, 显著提升训练效率. 进化搜索阶段由训练好的分布式Q学习模型驱动, 实现多种群进化算子的自适应选择, 动态平衡种群的收敛性和多样性.

3) 基于所提出的框架和 MO-AEOSSP 特征, 设计多种高效的进化算子和一种动态种群划分与选择策略, 建立 DMCEA 算法. 实验验证了 DMCEA 求解 MO-AEOSSP 的有效性.

1 问题描述与模型

1.1 问题描述

根据 Li 等[12] 的研究, 在确保任务调度收益最大化的基础上, 考虑卫星能耗资源的均衡利用, 可以有

效延长 AEOS 的使用寿命, 具有重要的工程意义. 因此, MO-AEOSSP 以最小化观测任务失败率和最小化卫星负载均衡为目标. 首先, 需要确定可行任务调度序列, 明确每个任务在不同卫星、轨道的调度位置, 确保相邻任务之间的时间间隔符合卫星的姿态切换时间约束. 同时, 需要考虑卫星能耗和存储空间的资源分配, 任务调度序列既要符合各卫星的实际资源承载能力, 又要合理化多星系统的资源利用率. 基于帕累托最优理论, 帕累托最优解是一组不可被改进的解, 这些解构成了帕累托前沿 (PF). 对 MO-AEOSSP 求解过程就是找到具有更佳决策方案的 PF, 并采用性能指标对 PF 进行评估.

考虑到带有时间依赖性的 MO-AEOSSP 具有复杂目标和约束, 本文做出以下合理假设:

1) 只考虑一次观测即可完成观测任务的点目标或小区域目标;

2) 一颗卫星一次只能观测到一个任务, 且任何在执行中的任务均不能被中断或抢占;

3) 本文研究的卫星型号 AS-01 为一颗半敏捷卫星, 其观测角度在观测过程中保持不变.

1.2 数学模型

1.2.1 符号定义

首先给出 MO-AEOSSP 的符号定义.

$M=\{m|m\in|M|\}$, M表示任务集合, $|M|$表示任务数量, m表示对应的任务索引.

$\mathrm{AS}=\{a|a\in|\mathrm{AS}|\}$, AS表示卫星集合, $|\mathrm{AS}|$表示卫星数量, a表示对应的卫星索引.

$S=\{O_{a,k}|a\in\mathrm{AS},\ k\in|S|\}$, S表示轨道集合, $|S|$表示轨道数量, $O_{a,k}$表示卫星a轨道k的索引.

$\mathrm{VTW}=\{w_m^{a,k}|m\in M,\ a\in\mathrm{AS},\ k\in S\}$,VTW表示所有任务的可见时间窗口集合, $w_m^{a,k}$表示任务m在卫星a轨道k上的可见时间窗口.

$w_m^{a,k}=\{\mathrm{st}_m^{a,k},\ \mathrm{ed}_m^{a,k},O_{a,k}|m\in M,a\in\mathrm{AS},\ k\in S\}$, 对于每个可见时间窗口$w_m^{a,k}$, $\mathrm{st}_m^{a,k}$和$\mathrm{ed}_m^{a,k}$分别表示该时间窗口的开始时间和结束时间, $O_{a,k}$表示该时间窗口所在的卫星和轨道索引.

$A_t^{w_m^{a,k}}=\{(\alpha_t^{w_m^{a,k}},\beta_t^{w_m^{a,k}},\gamma_t^{w_m^{a,k}})|m\in M,\ k\in S,\ t\in[\mathrm{st}_m^{a,k},\mathrm{ed}_m^{a,k}]\}$, $A_t^{w_m^{a,k}}$表示在某一时间窗口下, t时刻卫星观测任务m时的角度, 该角度由$\{\alpha_t^{w_m^{a,k}},\beta_t^{w_m^{a,k}},\gamma_t^{w_m^{a,k}}\}$组成, 分别对应$t$时刻下卫星俯仰、侧摆、偏航的角度分量.

D_m表示执行任务m需要花费的时间, E_m表示完成任务m后取得的收益, $C_{w_m^{a,k}}$表示调度时间窗口$w_m^{a,k}$的数据存储消耗, P_s、P_t和P_r分别表示卫星在准备、切换、执行任务时的能源消耗速率, W_a表示卫星执行任务前需要准备的时间, $\mathrm{MC}_{\max}^{a,k}$和$\mathrm{EC}_{\max}^{a,k}$分别表示卫星a中轨道k下的数据存储和能源储备量.

1.2.2 决策变量

在 MO-AEOSSP 数学模型中, 有 3 个决策变量用于表示一个可行的任务调度方案π, 包括:

1) $X_m^{a,k}$, 0-1 变量, 表示任务m是否在卫星a的轨道k上进行调度. 若调度, 则为 1, 否则为 0.

2) $Y_{m,n}^{a,k}$, 0-1 变量, 表示相邻任务m和n是否在相同卫星a的轨道k上进行调度. 若为 1, 则表示m作为n的前驱任务进行调度, 否则为 0.

3) $\mathrm{ST}_m^{a,k}$, 整数变量, 表示任务m在卫星a轨道k上的实际调度开始时间. 若为 0, 则表示该任务没有被调度.

1.2.3 时间依赖性切换时间

在带有时间依赖性切换时间的 MO-AEOSSP 中, 相邻任务的姿态切换时间计算取决于两个任务的实际调度开始时间. 假设前驱任务m和后继任务n在同一卫星a下的相同轨道k进行调度, 已知两个任务的实际调度开始时间$\mathrm{ST}_m^{a,k}$和$\mathrm{ST}_n^{a,k}$, 切换时间$\mathrm{Tran}(\mathrm{ST}_m^{a,k},\ \mathrm{ST}_n^{a,k})$由以下分段函数定义:

$$\mathrm{Tran}(\mathrm{ST}_m^{a,k},\mathrm{ST}_n^{a,k})=\begin{cases}11.6,\ \Delta\theta\leqslant 10;\\ a_1+\dfrac{\Delta\theta}{v_1},\ 10<\Delta\theta\leqslant 30;\\ a_2+\dfrac{\Delta\theta}{v_2},\ 30<\Delta\theta\leqslant 60;\\ a_3+\dfrac{\Delta\theta}{v_3},\ 60<\Delta\theta\leqslant 90;\\ a_4+\dfrac{\Delta\theta}{v_4},\ \Delta\theta>90.\end{cases}\tag{1}$$

其中: v_1、v_2、v_3和v_4表示相对于不同观测角度变化值下的不同角速度, 对应卫星相机的旋转能力参数, a_1、a_2、a_3和a_4对应角速度的常数项. 对于卫星 AS-01, 其每个值分别为$\{v_1,v_2,v_3,v_4\}=\{1.5,2,2.5,3\}$, $\{a_1,a_2,a_3,a_4\}=\{5,10,16,22\}$. $\Delta\theta$表示相邻任务的观测角度变化值, 其计算方法如下所示:

$$\begin{aligned}&\Delta\theta=|A_{\mathrm{ST}_m^{a,k}}^{w_m^{a,k}}-A_{\mathrm{ST}_n^{a,k}}^{w_n^{a,k}}|=\\&|\alpha_{\mathrm{ST}_m^{a,k}}^{w_m^{a,k}}-\alpha_{\mathrm{ST}_n^{a,k}}^{w_n^{a,k}}|+|\beta_{\mathrm{ST}_m^{a,k}}^{w_m^{a,k}}-\beta_{\mathrm{ST}_n^{a,k}}^{w_n^{a,k}}|+|\gamma_{\mathrm{ST}_m^{a,k}}^{w_m^{a,k}}-\gamma_{\mathrm{ST}_n^{a,k}}^{w_n^{a,k}}|.\end{aligned}\tag{2}$$

1.2.4 目标函数

本文研究的 MO-AEOSSP 具有两个目标函数, 分别为最小化观测任务失败率F_1和最小化卫星负载均衡F_2, 对于一个可行解π, 其两个目标函数的计算方法如下所示:

$$F_1(\pi)=1-\frac{\sum\limits_{a\in \mathrm{AS}}\sum\limits_{k\in S}\sum\limits_{m\in M}E_m\cdot X_m^{a,k}}{\sum\limits_{m\in M}E_m},\tag{3}$$

$$F_2(\pi)=\sqrt{\sum_{a\in \mathrm{AS}}\frac{\left(\mathrm{TEC}_a-\overline{\mathrm{TEC}}\right)^2}{(|\mathrm{AS}|-1)}}\Big/\overline{\mathrm{TEC}}.\tag{4}$$

其中: TEC_a表示卫星a的实际能源消耗总量, 其计算公式为

$$\mathrm{TEC}_a=\sum_{k\in S}\mathrm{EC}_{a,k},\tag{5}$$

$$\mathrm{EC}_{a,k}=\sum_{m\in M}X_m^{a,k}\cdot(P_s\cdot W_a+P_r\cdot D_m)+\sum_{m,n\in M}Y_{m,n}^{a,k}\cdot\mathrm{Tran}(\mathrm{ST}_m^{a,k},\mathrm{ST}_n^{a,k})\cdot P_t.\tag{6}$$

$\mathrm{EC}_{a,k}$表示卫星a轨道k的实际能源消耗量, $\overline{\mathrm{TEC}}$表示多颗卫星下能源消耗量的平均值, 其计算方法为

$$\overline{\mathrm{TEC}}=\sum_{a\in \mathrm{AS}}\mathrm{TEC}_a\Big/|\mathrm{AS}|.\tag{7}$$

1.2.5 问题约束

基于上述内容, 时间依赖性 MO-AEOSSP 存在以下约束:

$$\min\ (F_1,\ F_2).\tag{8}$$

$$\text{s.t.}\sum_{\substack{a\in \mathrm{AS}\\ k\in S}}X_m^{a,k}\leqslant 1,\ \forall m\in M;\tag{9}$$

$$\mathrm{st}_m^{a,k}\cdot X_m^{a,k}\leqslant \mathrm{ST}_m^{a,k},\\ \mathrm{ST}_m^{a,k}\leqslant(\mathrm{ed}_m^{a,k}-D_m)+L\cdot(1-X_m^{a,k}),\\ \mathrm{ST}_m^{a,k}\leqslant L\cdot\sum_{a\in \mathrm{AS}}\sum_{k\in S}X_m^{a,k},\\ \forall m\in M,\ a\in \mathrm{AS},\ k\in S;\tag{10}$$

$$\sum_{\substack{n\in M\cup\{\mathrm{ve}\}\\ n\neq m}}Y_{m,n}^{a,k}=X_m^{a,k},\quad\sum_{\substack{n\in M\cup\{\mathrm{vs}\}\\ n\neq m}}Y_{n,m}^{a,k}=X_m^{a,k},\\ \forall m\in M,\ a\in \mathrm{AS},\ k\in S;\tag{11}$$

$$\sum_{m\in M\cup\{\mathrm{vs}\}}Y_{m,\mathrm{ve}}^{a,k}=1,\quad\sum_{m\in M\cup\{\mathrm{ve}\}}Y_{\mathrm{vs},m}^{a,k}=1,\\ \forall a\in \mathrm{AS},\ k\in S;\tag{12}$$

$$\mathrm{ST}_m^{a,k}+D_m+W_a+\mathrm{Tran}(\mathrm{ST}_m^{a,k},\mathrm{ST}_n^{a,k})-\mathrm{ST}_n^{a,k}\leqslant L\cdot(1-Y_{m,n}^{a,k}),\\ \forall m,n\in M,\ m\neq n,\ a\in \mathrm{AS},\ k\in S;\tag{13}$$

$$\sum_{m\in M}X_m^{a,k}\cdot(P_s\cdot W_a+P_r\cdot D_m)+\sum_{\substack{m,n\in M\\ n\neq m}}Y_{m,n}^{a,k}\cdot\mathrm{Tran}(\mathrm{ST}_m^{a,k},\mathrm{ST}_n^{a,k})\cdot P_t\leqslant\mathrm{EC}_{\max}^{a,k},\\ \forall a\in \mathrm{AS},\ k\in S;\tag{14}$$

$$\sum_{m\in M}X_m^{a,k}\cdot C_{w_m^{a,k}}\leqslant\mathrm{MC}_{\max}^{a,k},\\ \forall a\in \mathrm{AS},\ k\in S;\tag{15}$$

$$X_m^{a,k}=\{0,1\},\ Y_{m,n}^{a,k}=\{0,1\},\ 0\leqslant\mathrm{ST}_m^{a,k},\\ \forall m,n\in M,\ a\in \mathrm{AS},\ k\in S.\tag{16}$$

其中: 式 (8) 表示 MO-AEOSSP 是一个最小化问题; 式 (9) 表示每个任务最多被调度一次; 式 (10) 表示时间窗口约束, 被调度任务的实际开始时间必须在时间窗口范围内, 且能完整执行, L为非常大的正整数; 式 (11) 和 (12) 表示若某个任务的 VTW 被调度, 则该 VTW 所在卫星和轨道上只存在一个前驱任务和一个后继任务, vs和ve分别表示每颗卫星每个轨道上的虚拟开始任务和虚拟结束任务; 式 (13) 表示切换时间约束, 同一卫星和轨道上的相邻任务调度必须具有足够的切换时间、前一任务的执行时间和后一任务的准备时间; 式 (14) 表示每颗卫星每个轨道的能源消耗约束, 每颗卫星的每个轨道能源消耗总量不能超过给定的能源储备量; 式 (15) 表示每颗卫星每个轨道的存储资源约束, 每颗卫星的每个轨道存储资源消耗总量不能超过给定的存储资源储备量; 式 (16) 表示决策变量的取值范围.

2 分布式元Q学习协同进化算法

2.1 解的表示

在时间依赖性 AEOSSP 中, 对于同一卫星和轨道下相邻的两个待调度前驱任务m和后继任务n, 若m和n的实际调度开始时间满足约束 (13), 则称之为可达, 记为$m\to n$, 否则记为$m\nrightarrow n$. Peng 等[5] 证明了半敏捷卫星相邻任务的切换时间满足先进先出定理和三角不等式规则, 并采用最小切换时间代替相邻任务的实际切换时间. 在给定m的调度开始时间下, 能够尽早开始进行n的调度, 通过计算相邻任务可达与不可达时间上下限, 实现处理此类问题的排列编码和全局松弛解码方法.

对于一颗卫星某一轨道上一个单独的待调度任务n, 其可行的调度开始时间ST_n取值范围为

$$\mathrm{ST}_n\in[\mathrm{st}_n,\mathrm{et}_n-D_n].\tag{17}$$

在该范围内任意时刻开始调度任务n, 均可满足问题约束, 将可行的最早开始时间记为EST_n, 最晚开始时间记为LST_n, 则存在以下对应关系:

$$\mathrm{EST}_n=\mathrm{st}_n,\tag{18}$$

$$\mathrm{LST}_n=\mathrm{et}_n-D_n.\tag{19}$$

对于相邻的前驱任务m和后继任务n, 若ST_m已知, 则n的最早开始时间$\mathrm{EST}_{m\to n}$可通过计算最小

切换时间获取, 其具体含义如图 2 所示. 此时, ST_n在$\mathrm{EST}_{m\to n}$至LST_n的任意时刻, 两个任务均可完成调度.

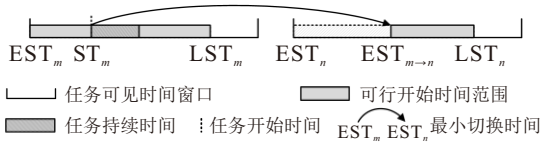

图2　后继任务n的最早开始时间

若ST_m不断向右侧的LST_m移动 (松弛), 任务n的$\mathrm{EST}_{m\to n}$将同样向LST_n松弛. 若存在某一时刻ST_m松弛导致$\mathrm{EST}_{m\to n}$超出LST_n, 则记该时间为最早不可达时间$\mathrm{UEST}_{m\nrightarrow n}$, 其具体含义如图 3 所示. 此时, 若$\mathrm{ST}_m$大于等于$\mathrm{UEST}_{m\nrightarrow n}$, 将导致$n$无法被调度.

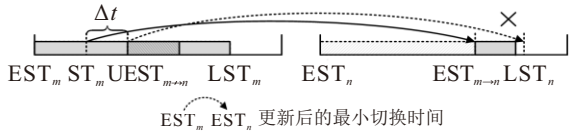

图3　后继任务n的最早不可达时间

同样地, 若ST_n已知, 则m的最迟开始时间$\mathrm{LST}_{m\to n}$可通过计算最小切换时间获取, 其具体含义如图 4 所示. 此时, ST_m在EST_m到$\mathrm{LST}_{m\to n}$的任意时刻, 两个任务均可完成调度.

图4　前驱任务m的最晚开始时间

若ST_n不断向左侧的EST_n松弛, 任务m的$\mathrm{LST}_{m\to n}$将同样向EST_m松弛. 若存在某一时刻ST_n松弛导致$\mathrm{LST}_{m\to n}$超出EST_m, 则记该时间为最晚不可达时间$\mathrm{ULST}_{m\nrightarrow n}$, 其具体含义如图 5 所示. 此时, 若$\mathrm{ST}_n$小于等于$\mathrm{ULST}_{m\nrightarrow n}$, 则将导致$m$无法被调度.

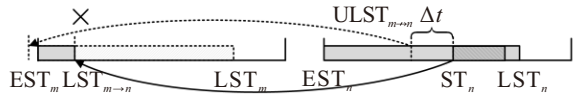

图5　前驱任务m的最晚不可达时间

上述涉及到的时间计算可以采用预处理方法获取, 具体方法可参见文献 [5]. 根据获取到的各项任务开始时间, 可以实现基于全局松弛策略的可行解表示方法.

对于某一卫星和轨道上任意数量的待调度任务VTW, 采用排列编码可以确定每个任务的插入顺序. 现有任务v插入到已被调度的前驱任务m和后继任务n之间, 根据全局松弛规则, 存在以下条件:

1) 基于m的前驱任务$m-1$确定m的$\mathrm{EST}_{m-1\to m}$, $\mathrm{EST}_{m-1\to m}$要小于$\mathrm{UEST}_{m\nrightarrow v}$.

2) 基于n的后继任务$n+1$确定n的$\mathrm{LST}_{n\to n+1}$, $\mathrm{LST}_{n\to n+1}$要大于$\mathrm{ULST}_{v\nrightarrow n}$.

3) 基于m的$\mathrm{EST}_{m-1\to m}$获取v的$\mathrm{EST}_{m\to v}$, 基于n的$\mathrm{LST}_{n\to n+1}$获取v的$\mathrm{LST}_{v\to n}$, $\mathrm{EST}_{m\to v}$要小于等于$\mathrm{LST}_{v\to n}$.

当上述条件均满足时, 则确定任务v可以插入到m与n之间. 在插入任务v后, 需要更新v之前任务的LST和之后任务的EST, 插入任务的具体过程如图 6 所示.

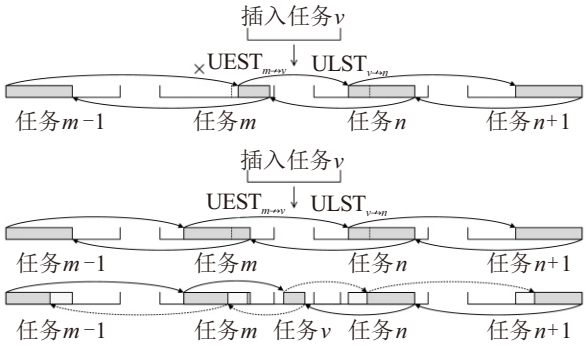

图6　基于全局松弛的插入任务 (解码) 过程

给定任意次序的待调度任务插入序列UT, 根据上述全局松弛任务插入规则, 可对UT进行解码, 得到最终可行解 (个体)π. 具体解码过程如下:

step 1: 输入UT, 初始化空解π, 令UT中的VTW数量为$|\mathrm{UT}|$, $i=1$.

step 2: 获取UT中第i个VTW, 根据全局松弛规则将其插入到对应卫星和轨道下. 若当前VTW可插入到π, 则更新π中其他任务的EST和LST, 形成新的可行解π', 否则跳转到 step 4.

step 3: 计算π'中每颗卫星的能源和存储消耗量, 判断是否满足式 (14) 和 (15). 若满足, 则令$\pi=\pi'$; 否则, 放弃插入当前VTW, π保持不变.

step 4: 根据π的结果更新UT和$|\mathrm{UT}|$. 若π包含当前VTW, 则删除UT后续同一任务下的其他VTW, 否则UT保持不变.

step 5: 令i的值自加 1. 此时, 若$i=|\mathrm{UT}|$, 则完成解码, 输出π, 否则返回到 step 2.

2.2　启发式初始化

在求解多目标问题时, 初始种群的质量是影响种群收敛速度和最终结果的关键[13]. 为提高初始种群整体解的质量, 基于 2.1 节中的排列编码和解码方法, 提出一种带有贪婪的启发式方法构造初始解, 通过Greed参数控制初始化个体时任务序列的贪婪率, 并采用一种任务分配策略来获得卫星负载更均衡的解. 给定待调度的任务插入序列UT, 启发式初始化的具体步骤如下:

step 1: 设定种群规模为|PS|, 初始化个体索引$i=1$, Greed $=1$, 种群PS $=\varnothing$.

step 2: 初始化一个空解π_i, 对UT中的每个任务m, 基于E_m进行降序排序, 获得新的任务序列UT′.

step 3: 基于Greed值将UT′划分为UT_1和UT_2, 其中UT_1为UT′中前Greed $\times\,100\,\%$的任务序列, UT_2为剩余的任务序列.

step 4: 对UT_2的顺序进行随机打乱, 之后将UT_1和UT_2前后拼接形成新的任务序列$\mathrm{UT}_{\mathrm{final}}$.

step 5: 基于$\mathrm{UT}_{\mathrm{final}}$的任务插入顺序, 按照第 2.1 节的方式进行解码, 在插入任务前首先根据式 (6) 计算各卫星和轨道的能耗, 将任务优先插入到具有最低能耗的卫星和轨道上, 得到可行解π_i.

step 6: 将得到的可行解π_i放入到PS内, 此时若$i=|\mathrm{PS}|$则完成种群初始化, 输出PS; 否则, 更新Greed和i的值, 并返回到 step 2, 更新后的Greed值为$1-(i/|\mathrm{PS}|)$, i值自加 1.

2.3 进化算子设计

为扩大 DMCEA 的解空间搜索范围, 针对多星 MO-AEOSSP 下的任务属性和时间窗口属性进行分析, 开发了相应属性的进化算子, 这些算子以“破坏-修复”的成对组合形式出现. 其中: 任务属性为任务的收益、能源消耗量和插入机会, 时间窗口属性为VTW之间的冲突长度. 基于上述信息, 进化算子通过动态任务删除与插入机制对当前解进行改进.

2.3.1 破坏算子

破坏算子根据特定的规则从可行解π中删除部分任务. 设π中被调度的任务数量为TN, 破坏算子对π中任务的移除比例设置为RP, 则破坏算子对π中任务删除数量RN $=$ TN $\cdot$ RP. 3 个破坏算子的规则如下:

1) 随机破坏.

给定π, 随机选择RN个任务进行删除, 形成新的可行解π'.

2) 基于收益/能源消耗量破坏.

给定π, 计算每个任务m的收益/能源消耗量ED_m, 其计算方法如下所示:

$$\mathrm{ED}_m=\begin{cases}\dfrac{E_m}{P_s\cdot W_a+P_r\cdot D_m+P_t\cdot \mathrm{Tran}(\mathrm{ST}_{m-1},\mathrm{ST}_m)},\\ \qquad \text{如果}m\text{存在前驱任务}m-1;\\ \dfrac{E_m}{P_s\cdot W_a+P_r\cdot D_m},\ \text{otherwise.}\end{cases}\tag{20}$$

之后删除具有最大ED_m的前RN个任务, 形成π'.

3) 基于时间窗口冲突破坏.

时间窗口冲突指在同一卫星和轨道下, 某一任务的VTW与其他任务VTW重叠的时间长度. 给定π, 计算π中每个任务m与UT内时间窗口冲突的长度之和CD_m, 其计算方法如下所示:

$$\mathrm{CD}_m=\sum_{n\in\mathrm{UT}}\max(0,\min(\mathrm{et}_m,\mathrm{et}_n)-\max(\mathrm{st}_m,\mathrm{st}_n)).\tag{21}$$

之后删除具有最大CD_m的前RN个任务, 形成π'.

2.3.2 修复算子

修复算子根据特定的规则, 对破坏后的可行解π'重新插入UT内的任务来改进解的质量. 3 个修复算子的规则如下:

1) 基于任务收益修复.

给定π', 对UT内每个任务m基于任务属性E_m进行降序排序, 并按照该顺序把任务依次插入到π'. 在插入每个任务时, 首先基于式 (6) 计算各时间窗口所在卫星和轨道上的能耗大小, 确保任务优先插入到具有最低能耗的卫星和轨道内.

2) 基于任务插入机会修复.

任务插入机会指任务所有VTW长度之和, 其值越大, 表明任务的插入机会越多. 给定π', 根据式 (22) 对UT内每个任务m计算插入机会OP_m,

$$\mathrm{OP}_m=\sum_{a\in\mathrm{AS}}\sum_{k\in S}(\mathrm{et}_m^{a,k}-\mathrm{st}_m^{a,k}).\tag{22}$$

之后基于任务属性OP_m进行升序排序, 并按照该顺序把任务依次插入到π', 该算子同样将任务优先插入到具有最低能耗的卫星和轨道内.

3) 基于时间窗口冲突修复.

给定π', 对UT内每个任务m基于式 (21) 计算CD_m, 之后基于CD_m进行升序排序, 并按照该顺序把任务依次插入到π'中.

2.4 分布式元Q学习

元Q学习是一种新的用于Q学习的拓展方法, 通过预训练获得更优的Q学习模型, 使得模型能够快速适应新任务. 训练过程产生的优秀个体用于提高初始化种群的质量, Q表用于指导种群中个体的进化策略选取. DMCEA 在元Q学习的基础上, 采用分布式并行预训练方法同时训练多张Q表, 并将多种进化算子与不同Q表结合, 实现种群中个体的自适应进化算子选择, 进一步拓展解搜索空间, 经多次迭代输出PF最优解集.

2.4.1 分布式元Q学习组成部分

DMCEA 采用分布式Q学习方法选择进化算子, 通过动态种群划分策略, 对收敛性种群和多样性种

群中的个体执行所选的进化算子, 之后对新生成个体进行评估, 实现下一次的状态和奖励反馈. 分布式元Q学习涉及到 3 张Q表, 分别用Q_1、Q_2、Q_3表示. 以下是分布式Q学习涉及到的智能体、状态、动作和奖励设计.

1) 智能体Agent(G). 分布式Q学习采用 3 个智能体G_1、G_2、G_3, 对应 3 张Q表指导个体改进. 在 MO-AEOSSP 下, 3 个智能体 (Q表) 分别设置和处理以下情况:

① F_1和F_2同时改进;

② F_1得到改进;

③ F_2得到改进.

在改进某一个体π时, 3 个智能体基于上述设置, 采用并行执行的方法各自选择一个进化算子, 产生 3 个新的个体π_1、π_2、π_3, 并通过评估新个体的质量分别更新自身Q表.

2) 状态State(S). 每个智能体对个体的改进存在两种状态: 改进和没有改进. 将个体得到改进记为 1, 没有改进记为 0, 则 3 个智能体的状态设置如下:

$$\begin{aligned} S_1 &= \begin{cases} 1, & F_1(\pi_1) < F_1(\pi)\,\&\,F_2(\pi_1) < F_2(\pi); \\ 0, & \text{otherwise}. \end{cases} \\ S_2 &= \begin{cases} 1, & F_1(\pi_2) < F_1(\pi); \\ 0, & \text{otherwise}. \end{cases} \\ S_3 &= \begin{cases} 1, & F_2(\pi_3) < F_2(\pi); \\ 0, & \text{otherwise}. \end{cases} \end{aligned} \tag{23}$$

3) 动作Action(A). 基于第 2.3 节中设计的 3 种破坏和 3 种修复算子, 经过“破坏-修复”组合最终形成 9 个进化算子, 这些进化算子被视为动作. 设进化算子集合为STR, 则存在

$$\text{STR} = \{A_{\text{ind}} | \text{ind} = 1, 2, \ldots, 9\}, \tag{24}$$

其中A_{ind}表示 9 种进化算子的索引.

4) 奖励Reward(R). 由于每个智能体引导种群中个体收敛的状态不同, 3 个智能体将根据个体改进结果产生不同的奖励. 每个智能体的奖励设置如下所示:

$$\begin{aligned} R_1 &= \begin{cases} 1, & F_1(\pi_1) < F_1(\pi)\,\&\,F_2(\pi_1) < F_2(\pi); \\ 0, & \text{otherwise}. \end{cases} \\ R_2 &= \begin{cases} 1, & F_1(\pi_2) < F_1(\pi); \\ 0, & \text{otherwise}. \end{cases} \\ R_3 &= \begin{cases} 1, & F_2(\pi_3) < F_2(\pi); \\ 0, & \text{otherwise}. \end{cases} \end{aligned} \tag{25}$$

2.4.2 预训练

在Q学习中, Q表用来保存“状态-动作”对应的值. 智能体基于Q表和当前状态确定动作. 经典Q学习方法需要不断迭代找到稳定的“状态-动作”组合. 为了充分探索和发挥每个进化算子 (动作) 对问题解空间下不同目标函数的改进性能, 设计了分布式预训练方法, 基于并行架构同时训练多张Q表, 且每张Q表和每个进化算子具有独立的训练过程. 分布式预训练的具体过程如下.

step 1: 初始化Q_1、Q_2、Q_3, 每张Q表均为 2 行 9 列, 初始值全为 0. 其中: 行表示状态S, 列表示动作A. 设置最大训练次数T_{train}, 种群规模|PS|, 训练次数$T = 0$.

step 2: 采用启发式初始化生成初始种群PS.

step 3: 保留PS内所有非支配个体, 删除其他个体.

step 4: 新建临时种群$\text{PS}_{\text{temp}} = \varnothing$. 对PS内个体并行使用每个进化算子进行改进, 将新生成的个体放入PS_{temp}内.

step 5: 对PS_{temp}内每个个体, 基于第 2.4.1 节中状态S、奖励R的设置, 更新状态S_{G_next}, 并通过式 (26) 给出的动作值函数更新Q_1、Q_2、Q_3中的Q值. 每更新 1 次Q表, T的值自加 1.

$$\begin{aligned} &Q_G(S_G, A_{\text{ind}}) \leftarrow \\ &Q_G(S_G, A_{\text{ind}}) + \mu[R_G + \\ &\sigma Q_G(S_{G\_\text{next}}, A_{\text{ind}}) - Q_G(S_G, A_{\text{ind}})], \\ &G \in \{1, 2, 3\},\ \text{ind} \in \{1, 2, \ldots, 9\}. \end{aligned} \tag{26}$$

其中: μ表示学习率, σ表示折扣因子. 为了平衡Q表的探索能力, μ采用 Rakshit 等[14] 的方法进行设置, 将第η次迭代训练记为T_η, 则μ的值为

$$\mu = 1 - \left(0.9 \cdot \frac{T_\eta}{T_{\text{train}}}\right). \tag{27}$$

σ的值在第 3.2 节参数设置部分经过测试后给出.

step 6: 从PS_{temp}中挑选出所有非支配个体, 形成新的PS. 此时, 若T的值大于T_{train}, 则结束预训练, 输出训练后的 3 张Q表和PS; 否则, 返回到 step 4.

2.5 自适应进化算子选择

在迭代搜索的不同阶段, 个体在解空间中的不断变化会使得Q学习模型产生偏差. 为了平衡不同阶段进化算子的探索能力, 提出自适应进化算子选择方法.

不同于经典的ε-Greed进化算子选择方法, 基于概率抽样的选择方法可以使Q值较低的进化算子仍能被选择, 并根据改进结果动态调整Q值, 使得不同进化算子在搜索阶段均能发挥自身效果. 此外, 基于概率抽样的选择方法能够减少Q学习的参数数量. 对每个智能体G, 自适应进化算子选择的过程如下.

step 1: 输入待改进个体π, 基于G的状态S_G获

取每个动作对应的Q值.

step 2: 根据下式计算G中每个进化算子选中的概率$\mathrm{prob}(A_{\mathrm{ind}}^G)$:

$$\mathrm{prob}(A_{\mathrm{ind}}^G)=\frac{Q_G(S_G,A_{\mathrm{ind}}^G)}{\sum\limits_{\mathrm{ind}\in\{1,2,\dots,9\}}Q_G(S_G,A_{\mathrm{ind}}^G)},$$
$$G\in\{1,2,3\},\ \mathrm{ind}\in\{1,2,\dots,9\}. \tag{28}$$

step 3: 根据$\mathrm{prob}(A_{\mathrm{ind}}^G)$的计算结果, 每个$G$通过轮盘赌选择一个进化算子用于改进$\pi$, 将被选中的进化算子记为$A_G$, 每个$G$产生的新解记为$\pi_G$.

step 4: 基于π_G的改进结果更新每个G的状态S_{G_next}, 并根据下式更新每个G的Q值:

$$\begin{aligned}&Q_G(S_G,A_G)\leftarrow\\&Q_G(S_G,A_G)+\mu[R_G+\\&\sigma\max_{\mathrm{ind}\in\{1,2,\dots,9\}}Q_G(S_{G\_\mathrm{next}},A_{\mathrm{ind}})-Q_G(S_G,A_G)],\\&\qquad G\in\{1,2,3\}.\end{aligned}\tag{29}$$

2.6 动态种群划分与选择

为增强 HEGSA 的探索能力, 动态种群划分策略将当前种群PS划分为收敛性种群SP、多样性种群DP和候选种群WP. 其中: SP由PS内所有非支配个体构成; 基于分解的多目标进化算法由于分解操作的存在, 在保持解的分布性方面有着很大优势[15], 因此DP采用动态权重生成的切比雪夫分解方法进行选择; WP由不包含在SP和DP的个体组成. 动态种群划分策略具体过程如下.

step 1: 初始化空种群SP、DP和WP, 设PS中个体数量为$|\mathrm{PS}|$, 给定多样性种群个体数量比例DR.

step 2: 将PS中所有非支配个体划分到SP. 设非支配解数量为$|\mathrm{SP}|$, 则PS内剩余的个体数量为$|\mathrm{PS}|-|\mathrm{SP}|$.

step 3: 计算DP的实际规模大小$|\mathrm{DP}|$, 其计算方法为

$$|\mathrm{DP}|=|\mathrm{PS}|\cdot\mathrm{DR}. \tag{30}$$

step 4: 基于$|\mathrm{DP}|$值大小生成分解权重向量集合WS, 生成方法采用经典的 Das & Dennis 法[16], 每个权重向量λ_i计算方法为

$$\lambda_i(w_1,w_2)=\left(i\cdot\frac{1}{|\mathrm{DP}|-1},1-\left(i\cdot\frac{1}{|\mathrm{DP}|-1}\right)\right),$$
$$i=0,1,\dots,|\mathrm{DP}|-1. \tag{31}$$

step 5: 获取PS内种群个体构成的理想点Z^*, 其计算方法为

$$Z^*(Z_1,Z_2)=(\min F_1(\pi),\min F_2(\pi')),\ \pi,\pi'\in\mathrm{PS}. \tag{32}$$

step 6: 遍历WS中的每个权重向量λ_i, 对每个权重向量, 计算PS内每个个体π的切比雪夫距离

$$\mathrm{CF}_\pi=\mathrm{Max}((F_1(\pi)-Z_1)\cdot w_1,(F_2(\pi)-Z_2)\cdot w_2), \tag{33}$$

将具有最小切比雪夫距离的个体放入到DP.

step 7: 将PS内剩余个体移动到WP, 输出SP、DP、WP.

种群选择策略与划分DP种群的步骤相同, 首先根据种群规模$|\mathrm{PS}|$和式 (31) 确定每个权重向量, 之后通过 step 5 和 step 6 从PS中挑选并形成新的种群.

2.7 算法流程

基于上述设计, DMCEA 流程如图 7 所示.

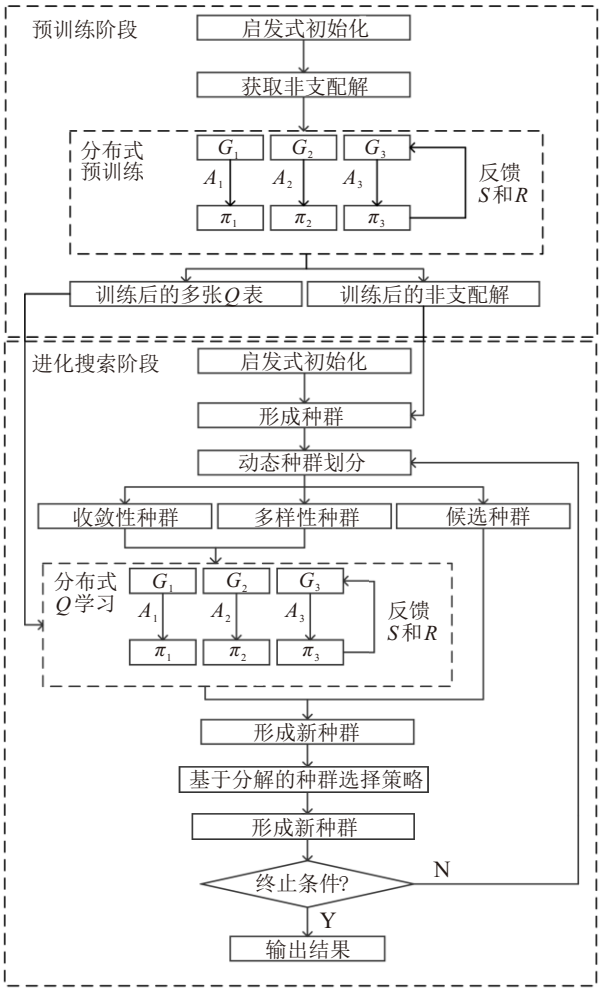

图7 DMCEA 流程

3 实验分析

3.1 实验设置

为了验证 DMCEA 的有效性, 现将该算法与Cplex、经典多目标算法和先进的 MO-AEOSSP 处理算法进行对比. 对于测试算例, 采用 Satellite Tool Kit (STK) 仿真软件进行生成. 实验场景涉及 10 颗卫星和 400 到 1 200 不等的任务数量, 根据卫星和任务数量划分为小、中、大 3 组算例. 具体算例大小设置如表 1 所示, 每个算例以“(卫星数_任务数)”进行表示,

目标点位置基于全球范围内随机生成, 日程安排的时间范围为2024年6月10日00: 00: 00 ~ 2024年6月11日00: 00: 00. 卫星相关参数如表2所示, 各元素分别表示卫星索引 (No.)、半长轴 (a)、离心率 (e)、轨道倾角 (∂)、近地点幅角 (ω)、升交点赤经 (Ω)、真近点角 (ρ).

表1　算例规模设置

算例规模	卫星数	任务数	算例名称
小	3	400	(3_400)
		450	(3_450)
	4	400	(4_400)
		450	(4_450)
中	6	600	(6_600)
		700	(6_700)
	7	600	(7_600)
		700	(7_700)
大	9	1 000	(9_1 000)
		1 200	(9_1 200)
	10	1 000	(10_1 000)
		1 200	(10_1 200)

表2　各卫星参数

No.	a	e	∂	ω	Ω	ρ
1	7 141 701.7	0.000 627	98.596 4	95.506 9	342.307	125.265 8
2	7 141 701.7	0.000 627	98.596 4	95.506 9	120	17
3	7 141 701.7	0.000 627	98.596 4	95.506 9	240	0
4	7 141 701.7	0.000 627	98.596 4	95.506 9	60	125.265 8
5	7 141 701.7	0.000 627	98.596 4	95.506 9	180	17
6	7 141 701.7	0.000 627	98.596 4	95.506 9	300	0
7	7 141 701.7	0.000 627	75	95.506 9	30	125.265 8
8	7 141 701.7	0.000 627	30	95.506 9	60	17
9	7 141 701.7	0.000 627	98.596 4	95.506 9	90	0
10	7 141 701.7	0.000 627	60	95.506 9	120	125.265 8

为公平起见, 所有算法在每个算例下独立运行10次, 实验采用的CPU为英特尔i5-13400F@2.5 GHz, 内存为16 GB, 所有算法均在Python 3.12.1的环境下进行编程. 所提出算法与两种经典多目标处理算法NSGA-Ⅱ[17]和IBEA[18], 以及两种先进的MMOD-Jaya[8]和ALNS+NSGA-Ⅱ[9]算法进行比较. 所比较算法的具体配置如下:

1) NSGA-Ⅱ和IBEA采用与DMCEA相同的排列编码、全局松弛的解码策略和种群规模参数.

2) NSGA-Ⅱ和IBEA的交叉算子采用部分匹配交叉 (PMX), 交叉位置随机生成. 变异算子采用插入变异策略, 随机交换任务的插入次序, 交叉率 = 0.5, 变异率 = 0.1.

3) ALNS+NSGA-Ⅱ和MMOD-Jaya均使用原文文献中的算法流程和参数.

4) 所有算法以DMCEA算法的总运行时间作为停止标准.

3.2　性能指标和参数设置

为全面衡量算法的收敛性和多样性, 本文采用超体积 (HV) 指标[19]对解集质量进行分析. HV通过计算最终解集形成的PF在目标空间中占据的体积大小来评估算法性能, HV越大表明PF的分布越好, 其计算方法如下所示:

$$\mathrm{HV} = \psi\left(\bigcup_{i=1}^{|\mathrm{PF}|} v_i\right). \tag{34}$$

其中: ψ表示勒贝格测度; v_i表示参考点与PF中第i个解构成的超体积, 参考点设置为 (1, 1); $|\mathrm{PF}|$表示PF中解的数量.

DMCEA包含4个关键参数, 分别为种群规模$|\mathrm{PS}|$, 折扣因子σ, 任务移除比例RP和多样性种群划分比例DR. 为避免实验结果的偶然性, 采用新生成的 (7_700) 算例进行测试, 每个参数的因子水平设置为$|\mathrm{PS}| = \{20, 40, 60, 80, 100, 120\}$, $\sigma = \{0.05, 0.1, 0.3, 0.5, 0.7, 0.9\}$, $\mathrm{RP} = \{0.025, 0.05, 0.075, 0.1, 0.125, 0.15\}$, $\mathrm{DR} = \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$. 在确定某一参数时, 其他参数保持固定值, 每个测试值独立运行10次. 基于测试结果, DMCEA的参数设置如表3所示.

表3　参数设置

参数名称	参数值
种群规模$\|\mathrm{PS}\|$	80
破坏率RP	0.075
多样性种群比例DR	0.2
最大迭代次数$T_{\max}$	50
学习率μ	$1-(0.9\times T_\eta/T_{\max})$
折扣因子σ	0.05
最大训练次数T_{train}	500

3.3　DMCEA策略有效性

为了验证DMCEA所提出的策略有效性, 记录DMCEA各策略在不同算例下运行10次后的平均HV结果, 如表4所示. 此外, 为进一步验证DMCEA启发式初始化和自适应进化算子选择策略的有效性, 添加了随机初始化和Q学习普遍采用的ε-Greed策略 (探索率采用常用的0.1) 的结果作为对比.

对表4中的平均HV结果进行分析. 首先, 启发式初始化相比于随机初始化, 其HV值提高了10.41% (从0.609 2提升至0.672 6), 表明启发式初始化能够生成质量更高的初始种群. 其次, 通过预训练策略能够进一步提升初始种群质量, HV值的提升幅度达到7.73% (从0.672 6提升至0.724 6). 之后, 在经典ε-Greed探索策略下, DMCEA采用进化算子和动态

表4 DMCEA 各模块运行结果

算例规模	算例名称	随机初始化	启发式初始化	启发式初始化+预训练	DMCEA (ε-Greed)	DMCEA
小	(3_400)	0.668 7	0.696 4	0.722 3	0.971 6	**0.990 2**
	(3_450)	0.622 4	0.683 1	0.706 8	0.923 1	**0.943 3**
	(4_400)	0.637 0	0.704 7	0.761 2	0.975 8	**0.992 2**
	(4_450)	0.646 7	0.696 9	0.773 7	0.963 5	**0.986 8**
中	(6_600)	0.623 1	0.681 9	0.759 8	0.959 8	**0.994 8**
	(6_700)	0.639 6	0.711 7	0.744 3	0.945 2	**0.989 1**
	(7_600)	0.611 3	0.695 9	0.766 2	0.953 8	**0.992 8**
	(7_700)	0.631 4	0.714 1	0.760 4	0.962 1	**0.993 0**
大	(9_1000)	0.572 4	0.643 2	0.708 1	0.914 2	**0.986 7**
	(9_1200)	0.560 9	0.631 8	0.673 3	0.921 6	**0.973 5**
	(10_1000)	0.537 9	0.595 9	0.656 9	0.930 9	**0.986 5**
	(10_1200)	0.558 5	0.616 1	0.662 5	0.912 1	**0.976 0**
平均值		0.609 2	0.672 6	0.724 6	0.944 5	**0.983 7**

种群划分与选择策略迭代改进种群, 其HV值提高了 30.35% (从 0.724 6 提升至 0.944 5), 该结果表明了进化算子和动态种群划分与选择策略的有效性. 最后, 采用自适应进化算子选择策略的 DMCEA 能够进一步提高算法性能, HV值提升了 4.15% (从 0.944 5 提升至 0.983 7). 上述结果表明, DMCEA 各策略均具有有效性.

3.4 与 Cplex 结果对比分析

给出 DMCEA 运行结果与 Cplex (版本 20.1.0) 在小算例下的最优解之间的直接比较. 基于 Peng 等[5] 对单目标 AEOSSP 的研究, MO-AEOSSP 存在非线性目标函数F_2无法直接求解, 因此采用ε约束法对F_2线性化进行求解, Cplex 的最大运行时长设置为 3 600 s, 超过 3 600 s 后得到的近似解用 “*” 标注, 最优结果加粗表示. Cplex 对 MO-AEOSSP 的求解过程如下.

step 1: 采用 Cplex 求解最小化F_1的结果, 该过程记为$F_{1\text{opt}}$, 设求解出的最小化F_1值为ε. 由于线性化F_2是在ε的基础上进行, 为防止误差, 需要保证ε解唯一性.

step 2: 根据ε结果, 设ε下已知每颗卫星a的负载为TEC_a, 对F_2分析可知, 当且仅当满足

$$\sum_{a\in \text{AS}} \left[\text{TEC}_a - \overline{\text{TEC}}\right]^2 = 0 \tag{35}$$

时, F_2取最小值.

step 3: 设理想解的平均TEC为TEC^*, 则TEC^*的计算方法为

$$\text{TEC}^* = \sum_{a\in \text{AS}} \text{TEC}_a \Big/ |\text{AS}|. \tag{36}$$

step 4: 基于TEC^*, 可将F_2线性化为F_2', 其计算方法为

$$F_2'(\pi) = \frac{\sqrt{\sum\limits_{a\in \text{AS}} \dfrac{(\text{TEC}_a - \text{TEC}^*)^2}{|\text{AS}|-1}}}{\text{TEC}^*}. \tag{37}$$

step 5: 添加约束

$$F_1(\pi) \leqslant \varepsilon \tag{38}$$

后, 对最小化F_2'进行求解, 该过程记为$F_{2\text{opt}}$, 此时完成整个求解过程.

表 5 给出了 DMCEA 与 Cplex 的对比结果, 通过最优解与运行时间的比较, 表明 DMCEA 在小算例下获得了高质量的解, 且计算时间远小于 Cplex.

表5 DMCEA 与 Cplex 对比结果

算例名称	Cplex			DMCEA	
	优化类型	(F_1, F_2)	CPU / s	(F_1, F_2)	CPU / s
(3_10)	$F_{1\text{opt}}$	(0, 0.519)	3.05	**(0, 0.441)**	**0.85**
	$F_{2\text{opt}}$	**(0, 0.441)**	5.69		
(3_20)	$F_{1\text{opt}}$	(0, 0.431)	138.59	**(0, 0.315)**	**2.04**
	$F_{2\text{opt}}$	**(0, 0.315)**	1 297.17		
(3_30)	$F_{1\text{opt}}$	(0, 0.229)	581.51	**(0, 0.185)**	**3.46**
	$F_{2\text{opt}}$	**(0, 0.185)**	2 062.14		
(3_40)	$F_{1\text{opt}}$	(0, 0.187)	1 405.59	**(0, 0.036)**	**4.42**
	$F_{2\text{opt}}$	(0, 0.069)*	3 600		

3.5 与已有算法结果对比分析

表 6 给出了所有算法在不同算例下HV的平均值 (标准差), 加粗值表示最优的平均HV结果. 为直观展示 DMCEA 的性能, 采用显著性水平为 0.05 的 Wilcoxon 秩和检验方法对统计结果进行分析, 统计结果采用 “+/−/=” 分别表示不同对比算法的性能显著 “优于/劣于/相似于” DMCEA 的次数. 图 8 为每个算法 10 次独立运行结果汇总的最终PF.

对表 6 中的数据进行分析, DMCEA 在 12 个测试算例下均获得了最好的结果. 其中: 在小规模算例下, DMCEA 与 ALNS+NSGA-Ⅱ、MMOD-Jaya 的表现相似; 在中、大规模算例下, DMCEA 取得了最优的表现, 且运行结果更加稳定; 在相同的运行时长下, DMCEA 能够取得更好的结果.

对图 8 分析可知, DMCEA 在不同规模算例下获得的PF能够完全覆盖其他对比算法. 其中: 在小规模算例下, DMCEA 获得的PF在多样性上超过了其他对比算法; 在中、大规模算例下, DMCEA 在相同时间内找到了覆盖解空间更大的PF, 具有更好的收敛性和分布性.

综合实验结果及上述分析可知, DMCEA 对不同测试算例具有较为明显的统计意义上的性能优势, 在处理 MO-AEOSSP 时存在较强的竞争力, 表明其具有很好的有效性和高效性.

表6　算法对比结果

算例规模	算例名称	ALNS+NSGA-Ⅱ	MMOD-Jaya	IBEA	NSGA-Ⅱ	DMCEA
小	(3_400)	0.983 1 (0.006 2) =	0.982 3 (0.004 3) =	0.936 9 (0.027 3) −	0.940 5 (0.013 4) −	**0.990 2 (0.005 3)**
	(3_450)	0.936 8 (0.036 1) =	0.938 4 (0.039 4) =	0.824 0 (0.017 1) −	0.886 5 (0.026 2) −	**0.943 3 (0.031 2)**
	(4_400)	0.988 3 (0.005 9) =	0.980 9 (0.010 2) =	0.945 4 (0.004 0) −	0.942 0 (0.013 9) −	**0.992 2 (0.006 2)**
	(4_450)	0.980 9 (0.007 4) =	0.980 4 (0.005 9) =	0.941 2 (0.013 1) −	0.946 9 (0.009 2) −	**0.986 8 (0.005 9)**
中	(6_600)	0.972 2 (0.011 1) −	0.967 1 (0.013 1) −	0.907 3 (0.022 2) −	0.914 9 (0.007 9) −	**0.994 8 (0.000 8)**
	(6_700)	0.967 0 (0.007 0) −	0.952 4 (0.008 2) −	0.891 3 (0.016 0) −	0.890 6 (0.005 8) −	**0.989 1 (0.002 9)**
	(7_600)	0.964 9 (0.008 6) −	0.954 7 (0.009 3) −	0.890 6 (0.013 9) −	0.871 4 (0.016 0) −	**0.992 8 (0.001 9)**
	(7_700)	0.956 9 (0.006 0) −	0.949 9 (0.012 7) −	0.887 9 (0.013 1) −	0.888 5 (0.014 9) −	**0.993 0 (0.001 3)**
大	(9_1000)	0.921 2 (0.010 3) −	0.903 6 (0.016 5) −	0.811 5 (0.018 8) −	0.817 3 (0.016 0) −	**0.986 7 (0.002 3)**
	(9_1200)	0.897 2 (0.006 1) −	0.869 2 (0.012 0) −	0.787 5 (0.010 6) −	0.802 1 (0.012 8) −	**0.973 5 (0.006 1)**
	(10_1000)	0.941 1 (0.007 7) −	0.920 8 (0.007 5) −	0.816 5 (0.007 3) −	0.830 9 (0.011 7) −	**0.986 5 (0.000 5)**
	(10_1200)	0.911 3 (0.008 6) −	0.908 0 (0.014 3) −	0.816 1 (0.004 3) −	0.823 9 (0.013 4) −	**0.976 0 (0.006 9)**
+/−/=		0/8/4	0/8/4	0/12/0	0/12/0	

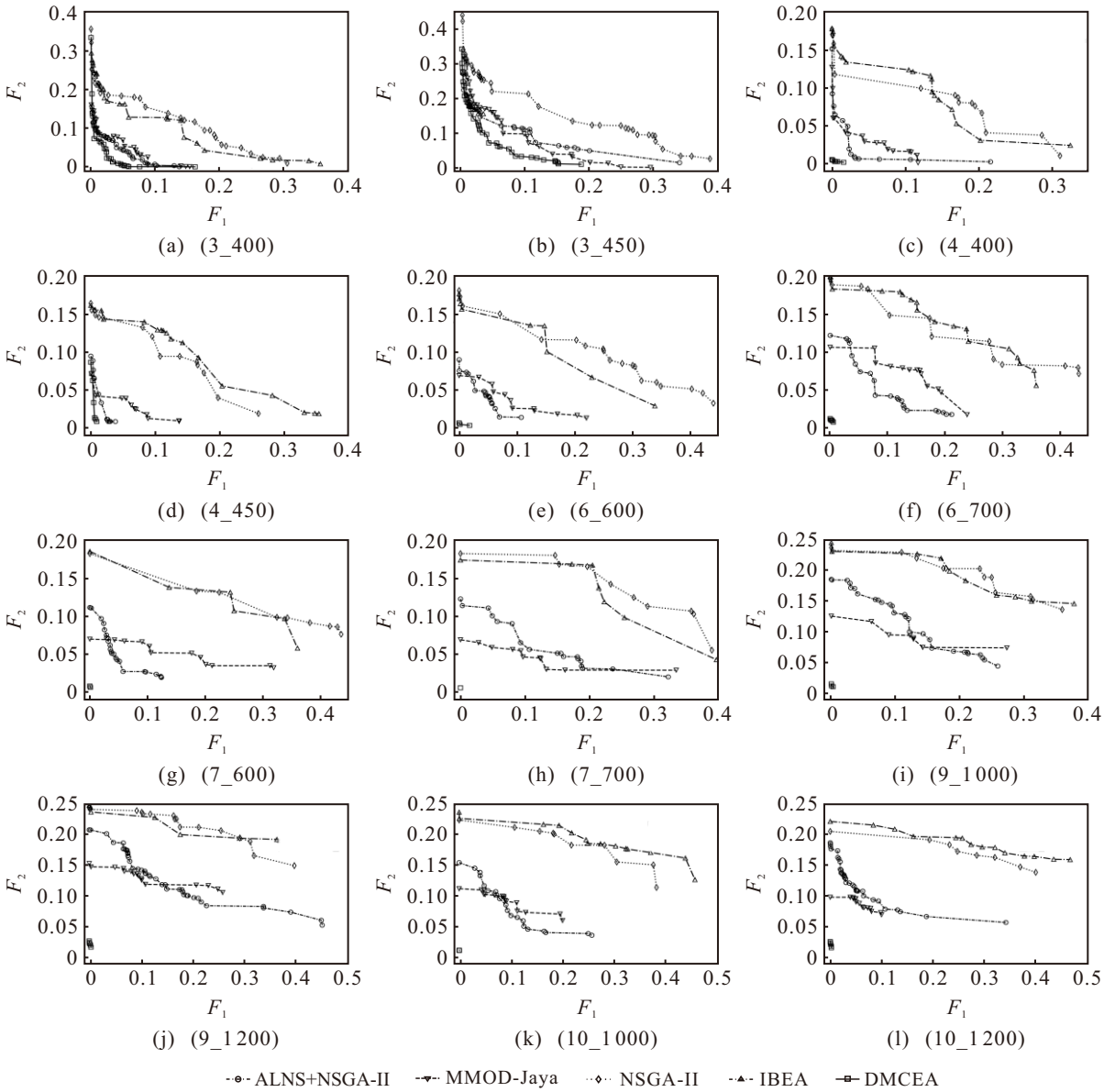

图8　所有算法在不同算例下的PF

4　结　论

本文针对带有时间依赖性的 MO-AEOSSP 提出了一种基于分布式元Q学习的 DMCEA, 设计了多种进化算子, 以自适应进化算子选择的方式探索解空间. 此外, DMCEA 将种群动态划分成 3 个不同的子群进行改进, 进一步平衡种群的收敛性与多样性. 在

不同规模的算例下对所设计的方法进行了仿真实验, 并将结果与经典和先进的多目标处理算法进行比较, 验证了本文所设计的算法在性能上优于其他算法. 下一步的工作重点是将 DMCEA 拓展应用到超大任务规模和动态环境下的实时优化等更复杂的卫星场景, 扩大 DMCEA 的适用范围.

参考文献 (References)

[1] 张佳唯, 邢立宁, 张玮, 等. 基于统一资源编码的成像卫星联合任务规划算法框架[J]. 控制与决策, 2022, 37(6): 1497-1504.
(Zhang J W, Xing L N, Zhang W, et al. A united mission planning algorithm framework based on uniform resource encoding for imaging satellites[J]. Control and Decision, 2022, 37(6): 1497-1504.)

[2] 彭观胜, 宋国鹏, 刘晓路, 等. 考虑时间依赖收益特性的敏捷卫星调度问题[J]. 运筹与管理, 2024, 33(8): 1-7.
(Peng G S, Song G P, Liu X L, et al. Agile earth observation satellite scheduling with time-dependent profits[J]. Operations Research and Management Science, 2024, 33(8): 1-7.)

[3] Lemaître M, Verfaillie G, Jouhaud F, et al. Selecting and scheduling observations of agile satellites[J]. Aerospace Science and Technology, 2002, 6(5): 367-381.

[4] 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.

[5] Peng G S, Song G P, He Y M, et al. Solving the agile earth observation satellite scheduling problem with time-dependent transition times[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2022, 52(3): 1614-1625.

[6] 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.

[7] Wei L N, Xing L N, Wan Q, et al. A multi-objective memetic approach for time-dependent agile earth observation satellite scheduling problem[J]. Computers & Industrial Engineering, 2021, 159: 107530.

[8] Wang B, Feng Y X, Zhang G H, et al. Memetic multi-objective discrete jaya algorithm for cooperative scheduling of multiple agile earth observation satellites[J]. IEEE Transactions on Aerospace Electronic Systems, 2024, 60(6): 8086-8099.

[9] Chang Z X, Zhou Z B, Yao F, et al. Observation scheduling problem for AEOS with a comprehensive task clustering[J]. Journal of Systems Engineering and Electronics, 2021, 32(2): 347-364.

[10] Wang H, Huang W Q, Magnússon S, et al. A strategy fusion-based multi-objective optimization approach for agile earth observation satellite scheduling problem[J]. IEEE Transactions on Geoscience and Remote Sensing, 2024, 62: 3472749.

[11] Huang W Q, Wang H, Wu J Y, et al. A reinforcement learning-enhanced dung beetle optimization approach for agile earth observation satellite scheduling[J]. IEEE Geoscience and Remote Sensing Letters, 2025, 22: 1-5.

[12] Li L M, Chen H, Li J, et al. Preference-based evolutionary many-objective optimization for agile satellite mission planning[J]. IEEE Access, 2018, 6: 40963-40978.

[13] 罗聪, 龚文引. 混合分解多目标进化算法求解绿色置换流水车间调度问题[J]. 控制与决策, 2024, 39(8): 2737-2745.
(Luo C, Gong W Y. A hybrid multi-objective evolutionary algorithm based on decomposition for green permutation flow-shop scheduling problem[J]. Control and Decision, 2024, 39(8): 2737-2745.)

[14] Rakshit P, Konar A, Bhowmik P, et al. Realization of an adaptive memetic algorithm using differential evolution and Q-learning: A case study in multi-robot path planning[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2013, 43(4): 814-831.

[15] 邓武, 蔡幸, 周永权, 等. 一种基于多策略差分进化的分解多目标进化算法[J]. 控制与决策, 2022, 37(2): 387-392.
(Deng W, Cai X, Zhou Y Q, et al. A novel decomposition multi-objective evolutionary algorithm based on differential evolution model with multi-strategy[J]. Control and Decision, 2022, 37(2): 387-392.)

[16] Das I, Dennis J E. Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems[J]. SIAM Journal on Optimization, 1998, 8(3): 631-657.

[17] Deb K, Pratap A, Agarwal S, et al. A fast and elitist multi-objective genetic algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197.

[18] Zitzler E, Künzli S. Indicator-based selection in multi-objective search[C]. Parallel Problem Solving from Nature — PPSN VIII. Berlin, 2004: 832-842.

[19] While L, Hingston P, Barone L, et al. A faster algorithm for calculating hypervolume[J]. IEEE Transactions on Evolutionary Computation, 2006, 10(1): 29-38.

作者简介

张广辉 (1981−), 男, 教授, 博士, 主要研究方向为计算智能、航天系统任务规划、人工智能与天基甲烷观测量化, E-mail: ghzhang@hebau.edu.cn;

魏晨轩 (2000−), 男, 硕士生, 主要研究方向为卫星任务规划、智能优化理论及算法, E-mail: chenxuanwei2023@hotmail.com;

冯彦翔 (1987−), 男, 副教授, 博士, 主要研究方向为空间信息系统优化设计、航天测控与星间链路、导航测控系统安全防护, E-mail: fengyanxiang@xjtu.edu.cn;

李晓玲 (1991−), 女, 副教授, 博士, 主要研究方向为无人系统调度、智能优化理论及应用, E-mail: xiaolingli@chd.edu.cn.