doi: 10.3969/j.issn.0490-6756.2020.05.011

基于冲突成像概率的多星任务预调度算法

徐明明[1], 王俊峰[2]
(1. 四川大学计算机学院, 成都 610065; 2. 四川大学空天科学与工程学院, 成都 610065)

摘 要: 研究高效的卫星调度算法对于解决任务分配不合理问题,充分利用卫星资源收集地面信息,提高对地观测系统的效率具有重要意义. 本文针对多星分布式协同调度模型的第一阶段预调度问题,考虑卫星性能指标和成像约束条件,将多星调度问题分解为单星自主调度问题. 为求解该问题,本文通过计算任务的可用时间窗口之间的潜在冲突系数和实际冲突系数及能量系数,提出一种基于冲突成像概率的调度(Collision Imaging Probability-Based Schedule, CIPBS)算法,根据可用时间窗口分布特点预测每个任务被每颗卫星成功调度的概率,从而设计任务分配方案,尽可能提高能被成像任务的总权重. 本文设计了 3 种不同分布的任务场景来评估 CIPBS 算法的适应性和效率,实验结果表明均有 10%~20%的性能提升.

关键词: 对地观测; 预调度; 图; 组合优化
中图分类号: TP301.6 **文献标识码**: A **文章编号**: 0490-6756(2020)05-0894-09

A collision probability-based algorithm for multi-satellites task pre-scheduling problem

XU Ming-Ming[1], *WANG Jun-Feng*[2]
(1. College of Computer Science, Sichuan University, Chengdu 610065, China;
2. School of Aeronautics and Astronautics, Sichuan University, Chengdu 610065, China)

Abstract: It is of great significance to study the efficient satellite scheduling algorithm to solve the problem of unreasonable task assignment and make full use of satellite resources to collect ground information, and improve the efficiency of the Earth observation system. Aiming at the first stage pre-scheduling problem of multi-star distributed cooperative scheduling, the multi-star scheduling problem is decomposed in the paper into a single-star autonomous scheduling problem under the satellite performance and imaging constraints. In order to solve the problem, a collision imaging probability-based schedule (CIPBS) algorithm is proposed by computing the real conflict coefficient, potential conflict coefficient, and energy allocation coefficient between the available time windows of the task, imaging probabilities of each task scheduling successfully by each satellite is predicted based on the distribution features of available time window, so the task assignment plan is designed to maximize the total weight of tasks that can be imaged. In this paper, three different types of task scenarios are designed to evaluate the adaptability and efficiency of the CIPBS algorithm. The experimental results show that the performance is improved by 10% to 20%.

Keywords: Earth observation; Pre-scheduling; Graph; Constrain optimization

收稿日期: 2019-08-28
基金项目: 四川省重点研发计划项目(2018GZ0529); 四川省重大科技专项(19ZDZX0024); 国家自然科学基金(91338107)
作者简介: 徐明明(1997-), 四川成都人, 硕士研究生, 研究方向为计算机网络. E-mail: 245561368@qq.com
通讯作者: 王俊峰. E-mail: wangjf@scu.edu.cn

1 引　言

对地观测是卫星一个非常重要的应用，据联合国外太空事务局（UNOOSA）记录，截至 2018 年底，有 601 颗对地观测卫星仍在轨运行，约占全球在轨卫星总数的 33%[1]. 卫星成像主要采用光学成像方式，其中小部分通过雷达等方法成像. 当卫星经过目标上方时，称之为“过顶”，此时卫星与目标之间可见，卫星才能对目标进行成像. 卫星的数量和传感器的分辨率都在逐年上升，但卫星的瞬时视野依然受限，对地观测需求不断增大，且发射卫星的费用非常高昂，卫星的工作寿命也很短暂，卫星资源仍然紧缺，为了缓解这一问题，大量研究者针对对地观测任务调度模型和算法进行不断优化和改进，力争在满足任务和资源约束的条件下为每个任务更加合理地分配卫星资源和决定任务的执行时间段.

对地观测任务调度可分为单星调度和多星调度. 早期的研究大多针对单颗卫星资源进行调度，随着观测任务日益繁重，卫星数量也不断上升，如果采用单星调度，调度系统内缺乏通用性，卫星的观测能力得不到充分利用. 如何综合利用卫星网络，高效的整合卫星资源，全面地发挥多卫星系统的观测能力成为各国研究的重点. 卫星调度问题是一个 NP 难问题[2]，无法在多项式时间内找到该类问题的最优解，只能通过离散化和次优化，在可接受的时间内找到较优解，主要的研究方法可分为确定性算法，搜索算法或启发式算法：① 确定性算法中采用较多的是分支定界法[3-5]，该算法需要消耗大量的空间来存储叶节点的边界. 当界定方法不适合时，分支定界法的性能退化至接近穷举法；② 搜索算法在多星调度问题上广泛使用，包括蚁群优化算法[6]、遗传算法[7]、爬山算法[8]、禁忌搜索算法[9-10]和模拟退火算法（Simulated Annealing，SA）[11]等，这种方法得到的解质量较高，但搜索空间较大时，计算时间较长，并且有陷入局部最优的风险；③ 启发式算法基于先验信息制定任务调度规则，Chen 等人[12]提出了几种基于优先级和冲突避免的启发式策略，并开发了基于时间的贪婪方法，基于权重的贪婪方法，以及改进的差分进化算法. Cho 等人[13]重新定义多星调度问题为旅行商问题的变体，并结合一种典型的基于邻域搜索的路径改进算法——Lin-Kernighan-Helsgaun 启发式算法来解决该问题. 王慧林等人[14]针对异构地球观测资源网提出了两种算法用以协调和分配观测任务，包括最高权重优先分配算法和禁忌列表模拟退火算法. 该类算法规则较为复杂，设计思路的偏离将造成性能与最优解相去甚大.

为了提高卫星调度系统的灵活性和鲁棒性，Morris 等人[15]提出了一种分布式的调度模型思路：将多星调度分解为两个子问题，预调度和单星自主调度. 首先，为每个任务分配卫星资源，然后每个卫星执行自主单星调度算法. 但他们只描述了框架的模型，并没有针对上述框架提出具体的策略. 已有一些研究者采用基于分解的分布式模型来进行多星观测调度：李菊芳等人[16]采用自适应蚁群优化算法搜索最优的任务分配方案，在单星自主调度阶段采用非常快速模拟退火算法. 实验结果表明，对比于禁忌搜索算法，性能提升并不明显（<8%）. 孙凯等人[17]提出了一种新的学习型遗传算法来解决任务资源匹配问题，并采用后移滑动策略及最优插入位置搜索策略解决单星任务调度子问题. Sinha 等[18]提出了一种基于多代理的卫星系统建模方法，并利用最大增益消息算法处理卫星中的任务分配问题. 当搜索算法应用于多星预调度阶段时存在缺陷：根据后续的单星调度阶段的结果更新解的适应度，完成反馈，导致预调度算法和单星自主调度算法的耦合. 此外，搜索算法的终止条件和初始值的设定也会影响性能，设置不当会导致分配方案不合理.

针对以上问题，本文提出了一种基于规则的启发式算法，适用于分布式的卫星任务规划系统，其目标是在预调度阶段，考虑能量约束，化解任务之间的冲突，完成优化目标，并且通过将计算负载从单个节点分散到多个来减少计算时间，从而提高系统的效率和稳定性. 该预调度算法实现简单，通用性强，只需卫星和任务的先验信息，后续的单星自主调度阶段的算法可以结合实际情况自由选择，无需编写反馈接口.

2 多星成像任务调度模型

2.1 多星成像任务调度模型输入与数据的预处理

多星调度问题可以表示为五元组$<S,T,C,F,O>$，其中，卫星集合 S 可表述为$\{S_k\}_{k=1}^{N_S}$，N_S是卫星的数量；任务集 T 可表述为$\{T_k\}_{k=1}^{N_T}$，N_T是数量任务；C 是约束集；F 是优化目标函数；O 是调度结果. 多星调度问题的输入参数及含义如表 1 所示.

根据图像卫星的特点，作如下假设.

① 用户提交任务后,无法撤回或更改;

② 用户无法在任务规划期间提交任务;

③ 一个任务可能有几颗卫星可以执行它,但最终只能由其中一颗卫星执行;

④ 卫星在一个运行周期内对一个任务目标点最多只能有一次观测机会;

⑤ 卫星观测具有原子性,一旦观测任务开始执行就无法被中断;

⑥ 在一个时间点,卫星只能对一个任务目标点进行观测.

基于上述前提假设,本文对多星问题的输入进行公式化表达,卫星和任务的主要参数如表 1.

表 1 卫星和任务的主要参数

Tab. 1 Main parameters of satellites and missions

输入	参数类型	参数	参数说明
卫星 S	标识符	i	卫星的标识符, $i\in\{1,2,\cdots,N_s\}$
	轨道参数	in_i	卫星 i 的倾角
		aop_i	卫星 i 的近地点角距
		sa_i	卫星 i 的半长轴
		$raan_i$	卫星 i 的升交点赤经
	载荷能力	b_i	卫星 i 的电池容量
		m_i	卫星 i 的存储能量
		oc_i	卫星 i 的最大开关机次数
任务 T	标识符	j	任务的标识符, $j\in\{1,2,\cdots,N_T\}$
	属性	ρ_j	任务 j 的优先级
		$\binom{long_j}{lat_j}$	任务 j 的观测目标点的经度和纬度
	时间要求	(a_i,d_i)	观察时间范围,任务 j 的最早开始时间和最晚结束日期
		md_i	任务执行最小持续时间

得到输入后,系统进行预处理,通过仿真得到目标点对卫星可见的时间范围,再计算该时间范围与任务的指定观察时间范围重叠的部分,称之为可用时间窗口,表示为

$$TW=\{tw_{ij}\mid 1\leqslant i\leqslant N_{SC},1\leqslant j\leqslant N_T\} \tag{1}$$

其中,$(tw_start_{ij},tw_end_{ij})$为卫星 i 可完成任务 j 的时间段;mpc_{ij}为卫星 i 完成任务 j 的最小电量消耗;mmc_{ij}为卫星 i 完成任务 j 的最小内存消耗.

2.2 多星成像任务调度模型的输出与优化目标

完成预调度后,得到分配矩阵 O.

$$O=\{o_{ij}\mid 1\leqslant i\leqslant N_s+1,1\leqslant j\leqslant N_T\} \tag{2}$$

$$o_{ij}=\begin{cases}1, & \text{任务 } j \text{ 被分配到卫星 } i\\ 0, & \text{else}\end{cases}$$

多星观测调度本质上是一个满足约束的调度机问题,约束条件对于调度过程的制约非常大,从而影响调度结果. 本文采用的是超额订购方式,即在给定的松弛程度下,分配到卫星的任务量可以超出卫星的执行能力,σ_i代表系统的超额分配率.

$$\forall k\in[1,\cdots,N_T],\sum_{i=1}^{N_S+1}o_k\leqslant 1 \tag{3}$$

$$\forall k\in[1,\cdots,N_T],\sum_{i=1}^{N_T}o_{kj}\leqslant\sigma_k\cdot oc_i \tag{4}$$

$$\forall k\in[1,\cdots,N_T],\sum_{i=1}^{N_T}o_{kj}\cdot mpc_{kj}\leqslant\sigma_k\cdot b_i \tag{5}$$

$$\forall k\in[1,\cdots,N_T],\sum_{i=1}^{N_T}o_{kj}\cdot mmc_{kj}\leqslant\sigma_k\cdot m_i \tag{6}$$

其中,S_{N_s+1}是一颗虚拟卫星,所有不能被卫星资源调度的任务将分配到该卫星上;式(3)是任务的单一执行约束,即每个任务只能被调度一次;式(4)~(6)是卫星的载荷能力约束,即每个卫星完成分配到的任务数量、所需的总耗电量和存储容量不能超过给定松弛程度下卫星的载荷能力.

预调度是多星调度的第一阶段,因此其优化目标和整个多星调度的相同. 当单星调度阶段完成时,分配矩阵 O 将被更新,主要是因为某些任务在单星调度阶段未能成功调度,只能分配到虚拟卫星上. 最终的优化目标是最大化执行完单星自主调度算法后可以成功调度的任务的总优先级,可以表示为

$$\max\Big\{\sum_{i=1}^{N_T}\rho_i\cdot(1-o_{i\langle N_S+1\rangle})\Big\}$$
$$\text{S. t. } (3)\sim(6) \tag{7}$$

3 基于冲突概率的调度算法

3.1 算法描述

本节提出 CIPBS 算法来处理多星预调度问题. 该算法的基本思路是在任务分配的过程中,根据当前分配情况计算任务在每个卫星上成功执行的概率,将任务分配给成像可能性最大的卫星. Cho 等人[13]提出影响成像概率的因素为当前任务可用时间窗口与同一卫星上其它任务的可用时间窗口的重合程度,但只考虑这一因素是较为片面的,因为在多卫星场景下,时间窗口冲突的多个任务被分配到不同的卫星资源上,相互的影响不一定存在. 如图 1 所示,为了更充分利用多星调度问题的启发式信息,本文将任务分为冲突任务和非冲突

任务，冲突任务的冲突成像概率由潜在冲突系数、实际冲突系数和能量系数构成，而非冲突任务只需要考虑能量的约束即能量系数，最终得到预分配结果.

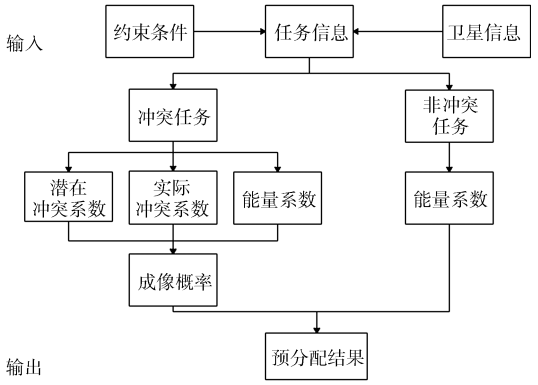

图 1　求解策略
Fig. 1　Solving stategy

预调度阶段的第一步是根据任务间可用时间窗口的重合度，将任务划分为冲突任务和非冲突任务，以简化后期冲突成像概率的求解. 如图 2 所示，$mcit_{ij}$ 是任务 j 在卫星 i 上的可用时间窗口的最长连续无碰撞时长. 如果

$$\exists i \in \{1,2,\cdots,N_s\}, mcit_{ij} > md_j \tag{8}$$

即在卫星 i 上，任务 j 的最长连续可用无碰撞时长超过该任务的最小执行时长，则称任务 j 在卫星 i 上无冲突. 若某一任务在任一卫星上无冲突，就是非冲突任务，否则，该任务是冲突任务.

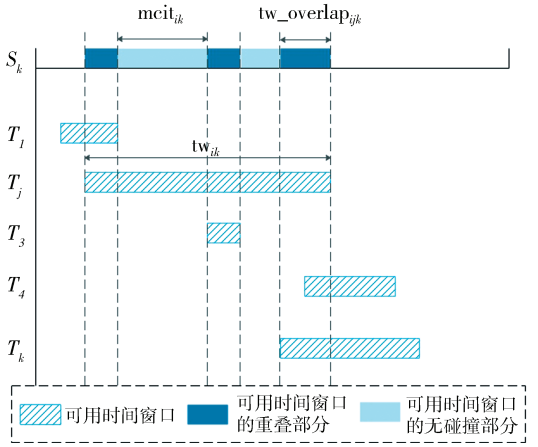

图 2　任务最长连续可用无碰撞时长示意图
Fig. 2　Maximum continuous idle time of the task

如图 2 所示，冲突成像概率由 p_{ij}、r_{ij} 和 e_{ij} 三部分构成，其中，δ 为无穷小量如下式.

$$c_{ij} = \frac{1}{p_{ij}+\delta} + \frac{1}{r_{ij}+\delta} + \frac{1}{e_{ij}} \tag{9}$$

其中，p_{ij} 是潜在冲突系数，反映的是任务间的潜在影响，即尚未分配到卫星 i 的任务在卫星 i 上有可用时间窗口，可能在后续的调度中被分配到卫星 i 上，和任务 j 争抢时间窗. 任务 j 的潜在冲突系数的计算方法为：找出所有未分配但在卫星 i 上有可用时间窗口的任务，计算(tw_start_{ij}，tw_end_{ij}).

$$p_{ij} = \sum_{k \in (T-\Gamma) \cap \Delta^*} \frac{tw\_overlap_{ijk} \cdot \rho_k}{d_k} \tag{10}$$

式(9 中)，r_{ij} 是实际冲突系数，反映的是任务间的实际影响，即在计算任务 j 分配到卫星 i 时的冲突成像概率时，已经有一些任务被分配到卫星 i 上，如果这些任务和任务 j 的执行会产生冲突，这种冲突必须在此时最小化. 在计算过程中，每个卫星都维护各自的有向无环图用以计算实际冲突系数. 记卫星 i 的图为G_i，其最长加权路径为$G_i_longest\text{-}path$，该图的结点由所有分配到该卫星的任务构成，图的有向边表示任务的执行序列. 加入一个任务 j 时，将任务结点 j 插入图内构成图G_i'，找到该图包含结点 j 的最长加权路径$G_i'_longestpath$. 所有因该任务无法执行的已分配任务的总权重就是任务 j 在卫星 i 上的实际冲突系数r_{ij}，可表示为

$$r_{ij} = G_i\_longestpath + p_j - G_i'\_longestpath \tag{11}$$

为了系统负载均衡和满足能量约束，要考虑完成卫星分配到的总任务的能量消耗，故定义能量系数的计算如下.

$$e_{ij} = \frac{B_i}{\sum_{k=1}^{N_T} mpc_{ik}} + \frac{M_i}{\sum_{k=1}^{N_T} mmc_{ik}} \tag{12}$$

综上所述，算法 1 给出了 CIPBS 算法的详细计算过程.

Algorithm 1　Collision Probability-Based Schedule(CIPBS)

1) 生成场景(S,T)
2) 初始化参数
3) 预处理，计算 tw,mmp,mpc,$mcit$
4) for $j \leftarrow 1$ *to* N_t do
5) for　$i \leftarrow 1$ *to* N_{SC} do
6) if　$MCIT_{ij} > MD_j$
7) $\Phi \leftarrow \Phi \cup \{T_j\}$
8) $\Gamma_i \leftarrow \Gamma_i \cup \{SC_i\}$
9) end if
10) end for
11) end for

12) while $\Lambda^* \neq \varnothing$ do

13) 随机选择任务 $T_j \in \Lambda^*$

14) for $i \leftarrow 1$ to N_{SC} do

15) 计算Λ^*中每个任务的 $TW\_overlap_{ijk}$

16) $p_{ij} \leftarrow \sum_{k \in \langle T-\Gamma \rangle \cap \Lambda^*} \frac{TW\_overlap_{ijk} \cdot \rho_k}{md_j}$

17) 构建卫星S_i上分配到的任务的有向无圈图G

18) 将任务 T_j插入G 构成G'

19) $r_{ij} \leftarrow G_i\_longestpath + p_j - G_i'\_longestpath$

20) $e_{ij} \leftarrow \frac{B_i}{\sum_{k=1}^{N_T} o_{ik} \cdot mpc_{ik}} + \frac{M_i}{\sum_{k=1}^{N_T} o_{ik} \cdot mmc_{ik}}$

21) $c_{ij} \leftarrow \frac{1}{p_{ij}+\delta} + \frac{1}{r_{ij}+\delta} + \frac{1}{e_{ij}}$

22) end for

23) if c_{ij}是$\{c_{ik} \mid 1<k<N_S\}$中的最大值并且满足约束(3) ~ (6)

24) $o_{ij} \leftarrow 1$

25) $G \leftarrow G'$

26) end if

27) $\Lambda^* \leftarrow \Lambda^* - \{T_j\}$

28) end while

29) 将 Λ 中的任务按照$\frac{\rho}{md}$降序排列

30) while $\Lambda \neq \varnothing$ do

31) 依次选择任务 $T_j \in \Lambda$

32) for $i \leftarrow 1$ to N_{SC} do

33) if $S_i \in \Gamma_j$

34) $e_{ij} \leftarrow \frac{B_i}{\sum_{k=1}^{N_T} o_{ik} \cdot mpc_{ik}} + \frac{M_i}{\sum_{k=1}^{N_T} o_{ik} \cdot mmc_{ik}}$

35) else $e_{ij} \leftarrow \infty$

36) end if

37) end for

38) if e_{ij}是$\{e_{ik} \mid 1<k<N_S\}$中的最大值并且满足约束(3) ~ (6)

39) $o_{ij} \leftarrow 1$

40) end if

41) $\Lambda \leftarrow \Lambda - \{T_j\}$

42) end while

43) return O

其中,Φ 表示所有无冲突任务的集合;Γ_i表示所有在卫星上S_i上无冲突的任务集合;Λ^* 表示所有尚未分配的冲突任务构成的集合;Λ 表示所有尚未分配的无冲突任务构成的集合.

3.2 算法分析

CIBPS 算法的时间复杂度主要由三个部分构成(N 表示任务规模,K 表示卫星规模):1) 划分冲突任务和非冲突任务的复杂度为 $O(K \cdot N^2)$. 其中,计算一个任务在一颗卫星上的最长连续可用无碰撞时长的时间复杂度为 $O(N)$,则计算 N 个任务在 K 颗卫星上的最长连续可用无碰撞时长的时间复杂度为$O(K \cdot N^2)$; 2) 冲突任务调度的复杂度为 $O(K \cdot N^3)$. 其中,对于单个任务在某一卫星资源上计算成像概率而言,计算潜在冲突系数的复杂度为 $O(N)$,采用 Dijkstra 算法构造有向无环图并计算其最短单源路径得到实际冲突系数的复杂度是 $O(N^2)$,计算能量系数的复杂度为 $O(N)$,故复杂度为 $O(N)+O(N^2)+O(N)=O(N^2)$,对于任务规模为 N,卫星规模为 K 的冲突任务调度问题计算时间复杂度为$O(K \cdot N^3)$; 3) 对于无冲突任务,只需计算其能量系数,复杂度为 $O(K \cdot N^2)$. 因此,CIBPS 算法的总时间复杂度为 $O(K \cdot N^2)+O(K \cdot N^3)+O(K \cdot N^2)=O(K \cdot N^3)$.

在开始调度时,p_{ij} 较大,r_{ij} 较小,此时c_{ij} 的计算主要取决于任务间的潜在冲突. 随着任务分配的进行,每一个任务被分配到卫星资源后,该任务对其他任务的潜在冲突就会在随后的调度计算中被剔除,同时,该任务会被添加到该卫星资源的有向无环资源图中,随着卫星资源的已分配任务信息增多,p_{ij} 减小,r_{ij} 增大,此时c_{ij} 的计算主要取决于任务间的实际冲突. 通过这种计算冲突系数分配卫星资源的方式,可以减少多个任务在同一段时间争用同一个卫星资源的情况,以达到均衡合理分配资源的目的.

4 实验与仿真

4.1 实验设置

本文实验运行在 4 GB 内存的 Inter Core i5 3.20 GHz 单核 CPU 上,卫星和任务目标点之间的可见时间窗口通过 Satellite Tool Kit 11.0 软件获取,算法实现语言为 Matlab. 由于在多星观测调度领域,尚无公开的标准数据集[19]. 为了全面的分析 CIBPS 的性能,本文设计了三组任务场景,分别对应不同的任务分布情况,任务的具体生成规则如表 2 所示.

4.2 实验结果

本文将 CIBPS 与 SA 算法进行对比,其中,SA

算法被 Globus 等人在实验中证明在各种典型的对地观测卫星调度算法中具有最好的性能[20]. 为了评估算法的性能，需要完成单星自主调度得到最终的调度结果. 本实验中单星调度算法采用了基于权重的贪婪算法[12]. SA 算法的终止条件设定为最优解在 100 次迭代中未曾更新. 对于每个实验用例，CIBPS 与 SA 都运行了 20 次. 表 3 给出了 3 种分布下 15 个实验用例的计算结果和 CPU 运行时间.

表 2　任务参数生成规则

Tab. 2　Generation rule of tasks

重要参数	分组	生成规则	值
任务数量	Group *a*	所有任务目标点均匀分布于地球表面	对应于 10，30 和 50 个纬度圈，任务总数量分别为 176，566 和 958.
	Group *b*	所有任务目标点分布于全球 32 个主要城市	平均对每个观测点生成 1，3，5 个任务，任务总数量分别为 32，96，160.
	Group *c*	所有任务目标点由分布 *a* 和分布 *b* 混合	Group a 和 Group b 的排列组合，任务总数量分别为 208，272，336，598，662，726，990，1054，1118.
任务优先级		均匀分布	[1,10]
任务最短执行时间		正态分布	均值为 120 s.

表 3　实验结果

Tab. 3　Experimental result

Group	No	N_1	N_2	N	T	CIPBS				SA				GAP/%
						$\bar{P}$	R/%	t/s	n	AVG	R/%	CPU/s	n	
a	1	176	0	176	1 072	1 003	93.56	0.15	166	1 003	93.56	119.77	166	0
	2	566	0	566	3 206	2 962.25	92.39	4.72	523.2	2 797.55	87.26	11.826	480.9	5.887
	3	958	0	958	5 234	4 729.5	90.36	62.62	855.1	4 186.8	80.00	20.422	722.4	12.962
b	4	0	32	32	188	177	94.14	0.046 5	29	166.7	88.67	370.81	26.1	6.179
	5	0	96	96	575	453.75	78.91	0.480 6	76.15	402.4	69.98	46.24	64.85	12.761
	6	0	160	160	874	653.6	74.78	1.561 5	116.25	541.4	61.95	68.69	92.85	20.724
c	7		32	208	1 260	1 180	93.65	0.26	195	1 168.45	92.73	35.59	191.1	0.99
	8	176	96	272	1 647	1 472.45	89.40	0.73	243.5	1 386.4	84.18	23.49	225.65	6.21
	9		160	336	1 946	1 676.05	86.12	1.66	286.5	1 523.3	78.28	33.32	254.65	10.03
	10		32	598	3 394	3 121.9	91.98	6.91	548.2	2 930.1	86.33	48.33	498.8	6.55
	11	566	96	662	3 781	3 352.75	88.67	10.24	581.25	3 109.35	82.24	52.17	525.75	7.83
	12		160	726	4 080	3 590.15	87.99	13.64	632.5	3 244	79.51	58.02	553.9	10.67
	13		32	990	5 422	4 873.35	89.88	71.47	877.05	4 302.7	79.36	73.82	740.35	13.26
	14	958	96	1 054	5 809	5 102.3	87.83	84.47	911.35	4 485.65	77.22	75.88	764.75	13.75
	15		160	1 118	6 108	5 224.6	85.54	100.49	937.65	4 607.20	75.43	81.75	789.95	13.40

表 3 中，Group a，b，c 表示三种不同的实验场景，对应于 5.1 节中提到的三种任务目标点分布情况；No 表示每个用例的编号；N_1 表示目标点满足分布 a 的任务数量；N_2 表示目标点满足分布 b 的任务数量；N 表示该用例的总任务数；T 表示该用例的总任务优先级；$\bar{P}$ 是调度成功任务的总优先级；R 是任务调度率即成功调度任务优先级百分比；t 是该用例的 20 次实验平均 CPU 运行时间；n 是成功调度任务数量；GAP 是性能提升比即 CIBPS 相较于 SA 的调度任务总权重上升比例.

可以通过对表 3 的分析比较 CIBPS 和 SA 之间的性能差异. 从实验结果中可以看出，随着任务数量的增加，任务调度率逐渐下降，同时，CIBPS 的任务调度率在大多数情况下都高于 SA 的. 用例 1 中，SA 可以和 CIBPS 实现相同的调度结果，这是因为此时的卫星资源相对充足，但 SA 花费的时间是 CIBPS 数百倍. 当任务数量增加时，卫星资源相对稀缺，任务的可用时间窗口间竞争加剧，此时 CIBPS 是一种更好的资源分配算法，能够缓解这种竞争，遏制任务完成率的急速下降. 因此，随着任务规模的增加，CIBPS 的优势更加明显.

在场景 a(用例 1～3)中，当任务数量较少时，SA 的性能接近 CIBPS，尽管搜索需要更多时间，SA 仍无法找到更好的解决方案，随着任务数量的上升，CIBPS 和 SA 完成率均呈下降趋势，但 CIBPS 相较与 SA 仍有 10%以上的提升. 场景 b(用例 4～6)下的结果有些不同：当任务数量较小时，CIBPS 和 SA 之间的性能差异也较明显，随着任务规

模的增长,SA 的完成率快速下降,CIBPS 的完成率缓慢下降,仍比 SA 提高了约 20%. 场景 c(用例 7~15)的任务目标点分布考虑实际的对地观测任务包含定期的巡航拍摄和突发的集中观测,结合均匀分布和集中分布的特征,增加了更多的任务. 该种场景下,CIBPS 和 SA 的任务完成率介于场景 a 和场景 b 之间,性能提升比也是如此. 因此,三种任务目标点分布的场景按资源短缺程度有如下排序:$b > c > a$,而场景 b 下,CIBPS 的性能提升最大,由此可以得出结论,CIBPS 更擅长在卫星资源紧缺和任务可用时间窗口冲突大的情况下处理多星预调度问题.

从 CPU 运行时间的角度来看,当任务规模较小时,CIBPS 的计算时间明显短于 SA. 随着任务数量的增加,CIBPS 的计算时间以三次量级增加,这验证了算法分析中对于算法复杂度的推算. 综上所述,CIBPS 能够完成在多项式时间内高效解决多星预调度问题的目标.

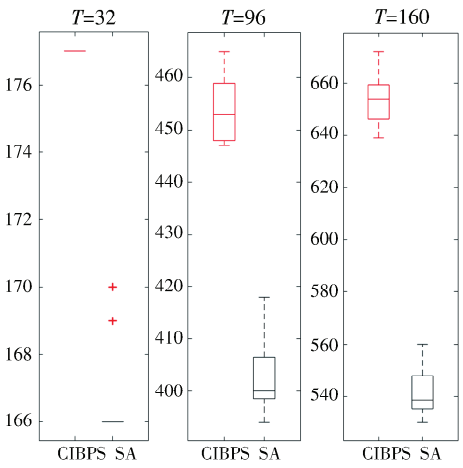

图 3 a 组实验结果盒图

Fig. 3 Box plot for Group(a)

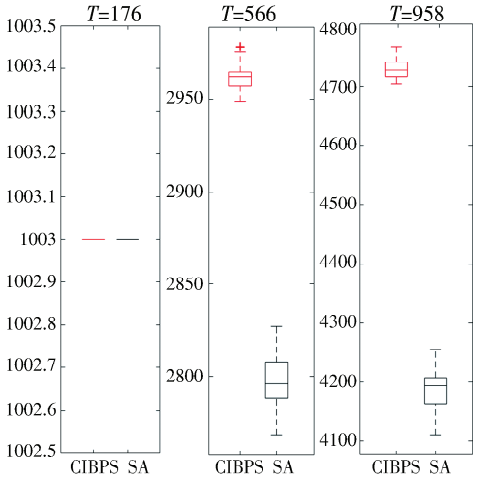

图 4 b 组实验结果盒图

Fig. 4 Box plot for Group(b)

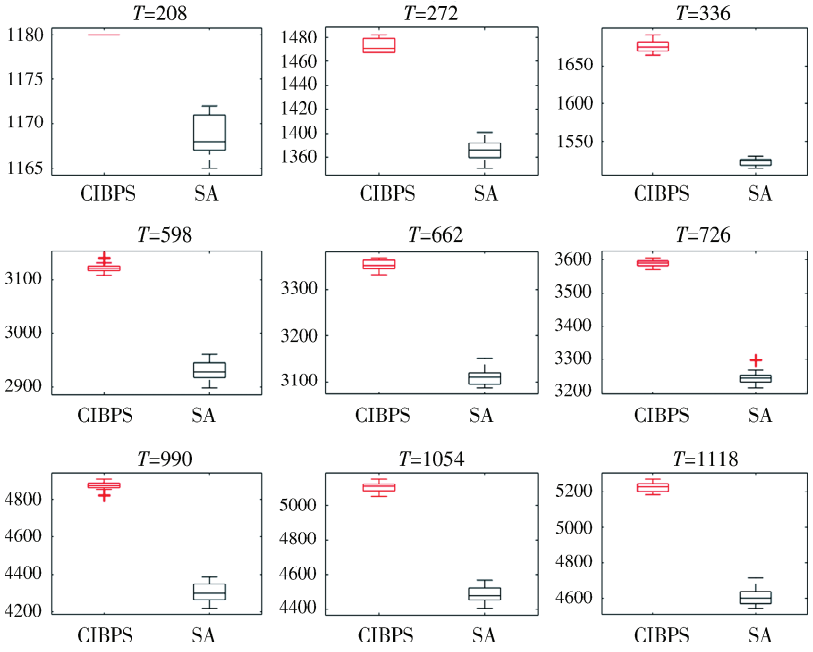

图 5 c 组实验结果盒图

Fig. 5 Box plot for Group(c)

图 3~图 5 是通过箱形图展示所有解的分布,可以看出,在场景 a(用例 1~3)中,在解空间较小时,CIBPS 和 SA 每次都可以找到到最优解,但是随着任务数量的增长,CIBPS 和 SA 在每次运行后

的解可能有所区别，而即使 CIBPS 算法的最差解也比 SA 的最佳解提升 5%以上，SA 算法解的四分位数范围和总范围都是 CIPBS 的 2 倍以上，说明 CIPBS 算法的解更加集中，即 CIBPS 算法可以获得更有效的稳定解决方案. 在场景 b(用例 4～6)中，即使解空间相比于场景 a 更小，即使此时的 SA 的解范围更加集中，CIBPS 的性能提升反而更加明显，原因是此时任务目标点的分布更加集中，任务时间窗口间的冲突更大，CIBPS 通过计算成像概率进行预测，化解任务间的冲突，可以取得更好的效果. 在场景 c(用例 7～15)中，CIBPS 和 SA 的解分布接近场景 a，CIBPS 的解非常稳定，四分位数范围和总范围都明显小于 SA.

总的来说，CIBPS 几乎在每种情况下都能找到更好的任务调度解. 由此可以得出结论，CIBPS 算法在性能方面明显优于 SA 算法，CIBPS 算法在有效性和稳定性方面的优势是显而易见的.

5　结　论

本文深入研究了多卫星对地观测任务调度模型和调度算法，对多星观测调度问题进行约束性分析，确定了基本假设和约束条件，针对现有的确定性调度算法的性能退化、搜索算法易陷入局部最优、其它启发式算法规则制定不全面的问题，根据观测卫星的实际情况，提出了 CIPBS 算法，使规划结果尽可能接近最优解. 在三种不同的任务分布下进行实验，通过调度结过分析对比表明本文提出的算法适应于多星预调度模型求解，能稳定的找到更好的解. 但是本文还是存在一些不足，在对地观测任务完成后的卫星数据下传时间窗口分配未进行优化，如何结合 CIBPS 算法的特点设计观测卫星数据下传算法也是今后研究的方向.

参考文献：

[1] 龚燃，刘韬. 2018 年国外对地观测卫星发展综述[J]. 国际太空，2019(2)：50.

[2] Brucker P. Complexity of machine scheduling problems [J]. Ann Discr Mach，1977，1：343.

[3] Lin W C，Liao D Y，Liu C Y，*et al*. Daily imaging scheduling of an earth observation satellite [J]. IEEE T Syst Man Cy A，2005，35：213.

[4] Chu X，Chen Y，Tan Y. An anytime branch and bound algorithm for agile earth observation satellite onboard scheduling [J]. Adv Space Res，2017，60：2077.

[5] Zhang D，Guo L，Cai B，*et al*. A hybrid discrete particle swarm optimization for satellite scheduling problem [C]// Proceedings of the IEEE conference anthology. China：IEEE，2013.

[6] Evans G W，Fairbairn R. Selection andscheduling of advanced tasks for NASA using 0-1 integer linear programming [J]. J Oper Res Soc，1989，40：971.

[7] Mansour M A A，Dessouky M M. A genetic algorithm approach for solving the daily photograph selection problem of the SPOT5 satellite [J]. Comput Ind Eng，2010，58：509.

[8] Beaumet G，Verfaillie G，Charmeau M C. Feasibility of autonomous decision making on board an agile earth-observing satellite [J]. Comput Intell，2015，27：123.

[9] Vasquez M，Hao J K. A "Logic-Constrained" knapsack formulation and a tabu algorithm for the daily photograph scheduling of an earth observation satellite [J]. Comput Optim Appl，2001，20：137.

[10] Sarkheyli A，Bagheri A，Ghorbani-Vaghei B，*et al*. Using an effective tabu search in interactive resources scheduling problem for LEO satellites missions [J]. Aerosp Sci Technol，2013，29：287.

[11] Evans G W，Fairbairn R. Selection and scheduling of advanced tasks for NASA using 0-1 integer linear programming [J]. J Oper Res Soc，1989，40：971.

[12] Chen X，Reinelt G，Dai G，*et al*. Priority-based and conflict-avoidance heuristics for multi-satellite scheduling [J]. Appl Soft Comput，2018，69：177.

[13] Cho D H，Choi H L. Atraveling salesman problem-based approach to observation scheduling for satellite constellation [J]. Int J Aeronaut Space Sci，2019，20：553.

[14] 王慧林，伍国华，马满好. 多类异构对地观测平台协同任务规划方法[J]. 航空学报，2016，37：254.

[15] Morris R A，Dungan J L，Bresina J L. An information infrastructure for coordinating earth science observations [C]// Proceedings of the IEEE International Conference on Space Mission Challenges for Information Technology. Pasadena，CA，USA：IEEE，2006.

[16] 李菊芳，白保存，陈英武. 多星成像调度问题基于分解的优化算法[J]. 系统工程理论与实践，2009，29：134.

[17] 孙凯，邢立宁，陈英武. 基于分解优化策略的多敏捷卫星联合对地观测调度[J]. 计算机集成制造系统，2013，19:127.

[18] Sinha P K，Dutta A. Multi-satellite task allocation

algorithm for earth observation [C]//Proceedings of the Region 10 Conference. Taipei, Taiwan, China: IEEE, 2017.

[19] 蔡德荣. 基于蚁群算法的多星联合成像任务规划问题研究[D]. 成都: 电子科技大学, 2012.

[20] Globus A, Crawford J, Lohn J, *et al*. A comparison of techniques for scheduling earth-observing satellites [C]// Proceedings of the Conference on Nineteenth National Conference on Artificial Intelligence. San Jose, CA: DBLP, 2004.

引用本文格式:

中　文: 徐明明, 王俊峰. 基于冲突成像概率的多星任务预调度算法[J]. 四川大学学报: 自然科学版, 2020, 57: 894.

英　文: Xu M M, Wang J F. A collision probability-based algorithm for multi-satellites task pre-scheduling problem [J]. J Sichuan Univ: Nat Sci Ed, 2020, 57: 894.