基于人工蜂群算法的中继卫星任务调度研究

开彩红*[①] 肖 瑶[①] 方 青[②]
[①](合肥工业大学计算机与信息学院 合肥 230009)
[②](中国电子科技集团公司第三十八研究所 合肥 230031)

摘 要：研究中继卫星任务调度问题可以为跟踪与数据中继卫星系统(TDRSS)的任务计划编排提供科学合理的决策方法，任务调度模型的建立与调度算法的设计是中继卫星任务调度的两个关键问题。该文针对中继卫星任务调度问题特点，综合考虑中继卫星与用户航天器之间具有可见时间窗、用户提交的任务属性、中继卫星前向资源受限等约束条件，建立了中继卫星任务调度约束规划模型并提出基于人工蜂群(ABC)算法的中继卫星任务调度算法。最后，通过仿真数据分析，表明基于人工蜂群算法的中继卫星任务调度算法是一种有效的、合理的调度方法。

关键词：中继卫星任务调度；可见时间窗；任务属性；约束规划模型；人工蜂群算法
中图分类号： TN927 **文献标识码**： A **文章编号**：1009-5896(2015)10-2466-09
DOI: 10.11999/JEIT150144

Relay Satellite Scheduling Based on Artificial Bee Colony Algorithm

Kai Cai-hong[①] Xiao Yao[①] Fang Qing[②]
[①](*School of Computer and Information*, *Hefei University of Technology*, *Hefei* 230009, *China*)
[②](*NO.*38 *Research Institute*, *China Electronics Technology Group Corporation*, *Hefei* 230031, *China*)

Abstract: Research on the relay-satellite scheduling problem provides scientific decision-making methods for the task planning of the Tracking and Data Relay Satellite Systems (TDRSS). How to develop a reasonable scheduling model and design the scheduling algorithm according to the model are two key issues to address. In this paper, according to the characteristics of the relay satellite scheduling problem, incorporating the constraints brought by the visible time window between the relay satellite and the user spacecraft, mission attributes submitted by users, and the limited resources of the relay satellite, a scheduling programming model is established. Furthermore, a scheduling algorithm based on the Artificial Bee Colony (ABC) algorithm is proposed. Finally, the simulation data analysis shows that the scheduling algorithm based on the ABC algorithm is an effective and reasonable scheduling method.

Key words: Relay satellite scheduling; Visible time windows; Mission attributes; Constraint programming model; Artificial Bee Colony (ABC) algorithm

1 引言

跟踪与数据中继卫星系统(Tracking and Data Relay Satellite System, TDRSS)是指通过位于地球同步轨道的中继卫星，为中、低轨道的航天器与航天器之间、航天器与地面站之间提供数据中继、连续跟踪与轨道测控服务的系统。中继卫星任务调度是指系统管理中心根据用户提出的服务申请，合理地分配中继卫星系统资源，并在资源冲突时进行优化调度，以完成尽可能多的任务[1−4]。随着航空航天领域中对地观测、军事侦察以及深空探测等技术的不断发展，中继卫星数据传输呈现大容量、高速率以及多样化中继任务的特点，TDRSS资源优化调度问题愈凸显得非常重要。

中继卫星任务调度问题是一个多约束的组合优化问题，需要考虑卫星间可见时间窗口、任务属性、卫星资源等方面的约束。中继卫星任务调度的目标是在满足可见时间窗口约束、任务属性约束、卫星资源有限的前提下，通过合理的安排任务执行顺序，以使得尽可能多的任务可以被成功执行，从而提高中继卫星通信资源的使用效率。近年来，出现了不少针对中继卫星任务调度模型和调度算法的研究[3−7]。文献[3]是针对美国的中继卫星系统，采用并行机调度算法对中继卫星调度问题进行研究，其数学模型仅考虑中继卫星与用户航天器之间存在不多于两个可见时间窗口的情况。文献[4]是通过建立满足中继卫星约束问题(CSP)的模型，并对该模型

收稿日期：2015-01-27；改回日期：2015-05-28；网络出版：2015-07-17
*通信作者：开彩红 chkai@hfut.edu.cn
基金项目：国家自然科学基金(61202459, 61571178)
Foundation Items: The National Natural Science Foundation of China (61202459, 61571178)

进行求解得到调度方案。文献[5]提出一种基于任务时间灵活度的调度算法，其主要的思想是动态地调节已调度任务在时间窗口的位置，从而实现更多任务的调度。文献[6]是基于有效基因路径表示的改进遗传算法，通过定义合适的交叉与变异算子，来求解最优的调度方案。文献[7]是针对微波与激光混合链路中继卫星动态调度问题，采用启发式算法来求解最优调度方案。

研究中继卫星任务调度问题的关键在于建立有效的调度约束模型进而设计调度算法，从而能够在短时间内形成一种较优的任务编排方案，以满足工程应用需求。本文首先针对中继卫星与用户航天器之间可见时间窗约束、用户提交的任务属性和中继卫星资源的约束，建立了任务调度约束规划模型。通过合理地设计目标函数，使得规划模型以尽可能成功调度更多任务，并优先调度优先级高的任务为目标。基于所建立的中继卫星任务调度规划模型，本文提出了一种基于人工蜂群(Artificial Bees Colony, ABC)算法的中继卫星任务编排算法。通过在算法的迭代过程中对每个可行调度方案进行评估、邻域操作寻找局部最优调度方案，从而获得全局最优调度方案。最后通过仿真实验分析，表明所提算法在调度效率方面具有优势，能够有效地提高中继卫星的使用效率，且算法本身具有低复杂度的特点，因而适用于工程实践。

2　中继卫星任务调度模型

2.1 中继卫星任务调度问题描述

中继卫星任务调度是指在满足任务调度约束的前提下，在相对较短时间内得到一种最优的调度方案。中继卫星任务调度的约束包括以下几点：(1)中继卫星与用户航天器之间具有可见时间窗约束。中继卫星与用户航天器并非时时可见，只有当它们位于彼此的可见范围内，才能建立通信链路，即任务的执行时间必须在中继卫星与用户航天器之间的可见时间窗内。如图 1 所示，ts 表示可见时间窗的开始时刻，te 表示可见时间窗的结束时刻，d 表示任务执行的持续时间(不包括服务准备和终止所需要的时间)。(2)任务属性约束。用户提交任务申请时，会提交任务的优先级、任务的持续时间、任务最早开始执行时间及任务最晚结束时间。进行任务调度时须考虑：执行任务的开始时间不小于任务最早开始时间；执行任务的结束时间不大于任务的最晚结束时间；当某个可见时间窗的时间长度小于任务的持续时间，由于本文所讨论的任务持续时间不允许拆分成多个子时间段，所以进行任务调度时必须选

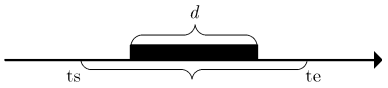

图 1 中继卫星与用户航天器之间的可见时间窗

择其他时间长度比此任务持续时间长的可见时间窗(如图 2 所示，$[\mathrm{ts}_k, \mathrm{te}_k]$表示第 k 个可见时间窗)。(3)对于中继卫星与用户航天器之间的某一条链路而言，同一时间最多只能有一个任务由该链路完成数据传输；与此同时考虑到卫星单址链路冲突的情况，在同一时刻任一颗用户航天器上最多只能有一条链路进行数据传输。

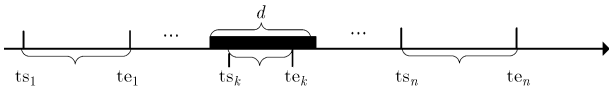

图 2 可见时间窗长度小于任务持续时间

2.2 中继卫星任务调度问题的约束规划模型

2.2.1 符号说明　L 表示所有中继卫星与所有用户航天器之间的链路集合，J 表示待执行任务的集合。

$\rho_i : \rho_i = |P| - p_i$，$p_i$ 为任务 i 的优先级，P 表示任务的优先级集合，$p_i \in P$。进行任务调度时需要考虑任务的优先级对任务调度编排的影响，因此将 ρ_i 记为任务的优先级 p_i 对应的加权值。定义 p_i 越小，任务 i 的优先级越高，即 ρ_i 越大，表示成功调度该任务获得的权值越大。

$\lambda_i : \lambda_i = |J| - \mathrm{pos}_i$，$\mathrm{pos}_i$ 表示任务 i 在某一任务调度方案中执行时间先后的次序，由任务编排中的该任务的开始执行时间决定。进行任务调度时我们希望优先级高的任务被安排在尽可能早的时间执行，因此将 λ_i 记为任务 i 执行时间先后次序对应的加权值。则 pos_i 越小，成功执行任务 i 获得的权值越大。

d_i 为任务 i 的持续时间，是一个连续的时间段。在本文的调度模型中，任务持续时间 d_i 包含两部分：(1)用户提交的实际用于数据通信的时间 $d_i^{'}$；(2) TDRS 服务准备阶段和服务终止阶段所花费的时间 d_0。其中，$d_i^{'}$ 由用户根据任务 i 的实际需要决定；而 TDRS 服务准备阶段和服务终止阶段所必须的时间 d_0 则受限于 TDRS 系统的软、硬件设备 [1)]。考虑到为执行任务所花费的控制开销，并为系统预留足够的软硬件处理时间，在本文的调度模型中，单个任务的持续时间应满足 $d_i \geq d_0$，d_0 为 TDRS 任一单

[1)]如在服务准备阶段，TDRS 地面站需要建立 TCP 通信连接、配置数据服务器系统、完成星间、星地通信链路设备配置等；而在服务终止阶段，则需要完成释放拆除链路、释放 TDRS 资源等工作。

个任务的最小持续时间，由中继卫星控管中心约定。在进行任务编排时，任务 i 的实际持续时间 $d_i = d_i' + d_0$。

B_i 为用户指定的任务 i 的最早开始执行时间、最晚结束时间构成的时间窗。用户提交任务申请时，设定任务在某个时间区间内执行，这个时间区间的开始时间和结束时间即为任务的最早开始时间和最晚结束时间，这个时间区间即为 B_i。A_{ik} 表示能够为任务 i 服务的链路 k ($k \in L$)上的可见时间窗集合。

$W_{ik}: W_{ik} = B_i \cap A_{ik}$，$k \in L$，能够为任务 i 服务的链路 k 上的所有可用时间窗的集合。任务的执行既要在中继卫星与用户航天器之间的可见时间窗内，同时也要在用户提交的时间段 B_i 内，因此任务在链路 k 上的执行时间必须在该链路的可见时间窗与用户提交的时间段的交集内，定义为任务 i 在链路 k 上可用时间窗。

a_{ik}^w 为任务 i 在链路 k 上的第 w 个可用时间窗的开始时间，b_{ik}^w 为任务 i 在链路 k 上的第 w 个可用时间窗的结束时间，$i \in J$，$w \in W_{ik}$，$k \in L$。

ε_{ijk} 为在链路 k 上，执行完任务 i 后立即执行任务 j 的切换时间，$i \neq j \in J$，$k \in L$。

$\eta_{ij}: \left|\eta_{ij}\right|$ 是一个很大的正实数，我们定义为 $\eta_{ij} = (\max_{k\in L, w\in W_{ik}}(b_{ik}^w) - \min_{k\in L, w\in W_{jk}}(a_{jk}^w))$，$i \neq j \in J$，表示执行任务 i 最早开始时间与执行 j 最晚结束时间之间的时间段。假设任务 i，j 的执行时间为 t_i，t_j，由于任务的执行时间必须不小于任务的可用时间窗中最早开始时间即有 $t_j \geq \min_{k\in L, w\in W_{jk}}(a_{jk}^w)$，且任务的执行时间必须小于任务的可用时间窗中最晚结束时间即有 $t_i < \max_{k\in L, w\in W_{ik}}(b_{ik}^w)$，则 $t_i - \max_{k\in L, w\in W_{ik}}(b_{ik}^w) \leq t_j - \min_{k\in L, w\in W_{jk}}(a_{jk}^w)$ 恒成立，即 $t_i - \eta_{ij} \leq t_j$ 恒成立。

任务 i 的定义 某一用户航天器与中继卫星之间的通信服务(在中继卫星系统中，主要是指前向通信)，该任务的优先级为 p_i，任务持续时间 d_i，任务 i 的最早开始执行时间、最晚结束时间构成的时间窗 B_i。

2.2.2 变量说明 $\chi_i^k: \chi_i^k = 1$ 表示任务 i 在链路 k 上被调度成功，否则 $\chi_i^k = 0$，$i \in J$，$k \in L$。

$\chi_{ij}^k: \chi_{ij}^k = 1$ 表示在链路 k 上执行完任务 i 后立即执行任务 j，否则 $\chi_{ij}^k = 0$，$i \neq j \in J, k \in L$。其中 $\chi_{0i}^k = 1$ 和 $\chi_{i0}^k = 1$ 分别表示任务 i 在链路 k 上所有调度成功任务中执行时间次序为最早和最晚的一个任务。

$\gamma_{ik}^w: \gamma_{ik}^w = 1$ 表示任务 i 在链路 k 上的可用时间窗 w ($w \in W_{ik}$)内执行，否则 $\gamma_{ik}^w = 0$。

$o_{ij}: o_{ij} = 1$ 表示任务 i 在任务 j ($j \neq i$)之前执行，否则 $o_{ij} = 0$。

$\delta_{hk}: \delta_{hk} = 1$ 表示同一颗卫星(用户航天器或中继卫星)上的两条链路 h,k ($h \neq k$)发生冲突，否则 $\delta_{hk} = 0$。两条链路冲突是指同一卫星的两条链路在同一时刻都要进行数据传输时所发生的卫星资源冲突。

t_i：任务 i 调度成功的开始执行时间。

2.2.3 任务调度模型 中继卫星任务调度约束规划模型如下：

$$\max\left\{\sum_{i\in J}\left((\rho_i\lambda_i)\sum_{k\in L}\chi_i^k\right)\right\} \tag{1}$$

$$\text{s.t.}\ \sum_{k\in L}\chi_i^k \leq 1,\quad i \in J \tag{2}$$

$$\sum_{j\in J\setminus\{i\}}\sum_{k\in L}\chi_{ij}^k \leq 1,\quad i \in J \tag{3}$$

$$\sum_{k\in L}\left(\sum_{i\in J\setminus\{j\}}\chi_{ij}^k - \sum_{z\in J\setminus\{j\}}\chi_{jz}^k\right) = 0,\quad j \in J \tag{4}$$

$$\sum_{k\in L}\left(\chi_i^k - \sum_{w\in W_{ik}}\gamma_{ik}^w\right) = 0,\quad i \in J \tag{5}$$

$$t_i + \sum_{k\in L}\chi_{ij}^k\left(d_i + \varepsilon_{ijk}\right) - \left(1 - \sum_{k\in L}\chi_{ij}^k\right)\eta_{ij} \leq t_j, i \neq j \in J \tag{6}$$

$$-\left(1 - \sum_{k\in L}\chi_i^k\right)\left|\eta_{ij}\right| + \left(t_i + d_i o_{ij}\right)\left(o_{ij} - o_{ji}\right) \leq \left(1 - \sum_{h\in L}\chi_j^h\right)\left|\eta_{ij}\right| + \left(t_j + d_j o_{ji}\right)\left(o_{ij} - o_{ji}\right),\quad i \neq j \in J,\ \delta_{hk} = 1 \tag{7}$$

$$\sum_{k\in L}\chi_i^k\left(\sum_{w\in W_k}\gamma_{ik}^w a_{ik}^w\right) - t_i \leq 0,\quad i \in J \tag{8}$$

$$\left(1 - \sum_{k\in L}\chi_i^k\left(\sum_{w\in W_{ik}}\gamma_{ik}^w\right)\right)Tb_i + \sum_{k\in L}\chi_i^k\left(\sum_{w\in W_{ik}}y_{ik}^w b_{ik}^w\right) \geq t_i + d_i,\quad i \in J \tag{9}$$

$$\left.\begin{aligned} &\chi_i^k = \{0,1\},\quad i \in J,\ k \in L \\ &\chi_{ij}^k = \{0,1\},\quad i \neq j \in J,\ k \in L \\ &\gamma_{ik}^w = \{0,1\},\quad i \in J,\ k \in L,\ w \in W_{ik} \\ &o_{id} = \{0,1\},\quad i \neq d \in J \\ &\min_{k\in L, w\in W_{ik}}\left(a_{ik}^w\right) \leq t_i \leq \max_{k\in L, w\in W_{ik}}\left(b_{ik}^w\right) \end{aligned}\right\} \tag{10}$$

其中，适应度函数式(1)表明了调度的目标是确保更高优先级的、更多的任务能够调度成功，式中 ρ_i，λ_i 分别代表任务的优先级和任务在该任务编排中执行时间的先后次序对适应度函数的贡献值，它们的乘积 $\rho_i\lambda_i$ 表明了优先级高的任务能够优先调度。约束条件式(2)表明若一个任务调度成功则该任务只能在一条链路上执行，否则该任务调度失败。约束条

件式(3)指明在同一条链路上的所有调度成功的任务中，且表现为在这条链路上执行完任务 i 之后执行任务 j，那满足这样的任务 i 最多只能有一个，即表明了在一条链路上在同一时间最多只能为一个任务服务。约束条件式(4)确保了在同一条链路上执行的任务它们必须排成一个有序序列，也说明了一条链路在同一时刻最多只能执行一个任务。约束条件式(5)表明了调度成功的任务执行时仅能在一个可用时间窗口内进行。约束条件式(6)表明了若两个任务 i，j (j 在 i 之后执行) 在同一链路上执行时，任务 j 的开始执行时间 t_j 在任务 i 执行完 t_i+d_i 之后，进一步强调链路在同一时刻只能执行一个任务；若任务 i, j 没有这层关系，由于 $t_i-\eta_{ij}\le t_j$ 恒成立，所以不等式式(6)仍然成立。约束条件式(7)表示两条链路 h，k 发生资源冲突时，分别在同一颗卫星的两条链路 h,k 上任何两个任务 j,i 均调度成功时，任务 j,i 的执行时间是不能有重叠的，要么任务 j 在任务 i 之前执行即 $t_j+d_j\le t_i$，或任务 i 在任务 j 之前执行即 $t_i+d_i\le t_j$，表明了一颗卫星在同一时刻只能有一条链路为任务服务。约束条件式(8)和式(9)表明了调度成功任务的执行必须在一个可用时间窗内进行，其中约束条件式(8)表明任务的执行时间必须不小于该可用时间窗的开始时间即 $t_i\ge a_{ik}^w$；约束条件式(9)表明任务的结束时间必须不大于该可用时间窗的结束时间 $t_i+d_i\le b_{ik}^w$。约束条件式(10)中 $\min_{k\in L,w\in W_{ik}}\left(a_{ik}^w\right)\le t_i\le\max_{k\in L,w\in W_{ik}}\left(b_{ik}^w\right)$ 表明任务的可执行时间必须在中继卫星与用户航天器之间的可见时间窗内，也只有在可见时间窗内才能执行任务，即进一步约束任务可执行时间 t_i。

3　基于人工蜂群(ABC)算法的中继卫星任务调度算法

3.1　人工蜂群算法概述

ABC 算法是模拟蜜蜂行为提出的一种优化方法，它通过各人工蜂个体的局部寻优行为，最终在群体中使全局最优值凸显出来[8−12]。该算法是一种迭代算法，算法中种群的各个个体寻找的蜜源代表一个可行解，采用适应度值来衡量该蜜源的蜂蜜含量及远近程度。通过在迭代过程中对每个解的质量进行适应度评估和邻域操作，从而得到局部的最优解，最终得到优化问题的全局最优解。ABC 算法在许多优化问题中取得了成功的应用[10−14]。接下来探索如何将 ABC 算法运用于解决中继卫星任务调度问题，并评估其调度决策的效率。

3.2　基于人工蜂群算法的中继卫星任务调度

3.2.1 解的描述　运用 ABC 算法进行中继卫星任务调度时，一个解 (x) 代表了所有待执行任务的一个有序排列，比如，待执行的任务共有 N 个，则调度问题的一个解就代表了这 N 个任务的一个有序排列 ($j_1,j_2,\cdots,j_{N-1},j_N$, j_n 表示任务编号，$n\in N$)。对解 (x) 所确定的任务排列，我们进行任务编排(具体的任务编排方法见第 3.2.3 节)，从而确定按照解 (x) 所确定的任务排序中，有哪些任务可以被成功调度，每个成功调度任务的开始执行时间等。进而，可以计算得出解 (x) 所对应的适应度函数值(具体的计算方法见第 3.2.2 节)。ABC 算法的最优解就是使得适应度函数最大的解，其对应的任务编排即为最优调度方案。

3.2.2 适应度函数的定义及计算　对于一个解 (x)，其适应度函数(目标函数)定义为

$$f(x)=\sum_{i=1}^{N}\alpha_i\rho_i\lambda_i \tag{11}$$

式中 $\alpha_i=1$ 表示该解中第 i 个任务调度成功，否则 $\alpha_i=0$。式中 ρ_i 表示该解中第 i 个任务的优先级对适应度函数的贡献值，λ_i 表示该解中第 i 个任务的执行顺序对适应度函数的贡献值，ρ_i 用来表示安排任务调度时优先考虑优先级高的任务对适应度函数的影响因素，在应用中为了能够平衡“尽可能多的成功调度任务”和“优先安排优先级高的任务”之间矛盾，可以通过设置 ρ_i 式子来调整任务优先级对适应度函数的影响所占比重。适应度函数表明成功调度的任务数越多，适应度函数值越大；当成功调度的任务的优先级贡献值越大且执行顺序贡献值越大，适应度函数值越大。算法最优解就是使适应度函数值最大的一个解，其对应的任务编排即为最优调度方案。

3.2.3 对解(x)的任务编排及解的初始化　在应用 ABC 算法进行中继卫星任务调度时，对解 (x) 进行任务编排的过程就是对全体任务的有序排列(j_1, $j_2\cdots,j_{N-1},j_N$, j_n 表示任务编号)中每个任务 j_i 顺次进行是否可成功调度判定、计算任务执行时间 t_i 的过程。按照解(x)确定的任务顺序，从第 1 个任务开始，我们执行下述调度判定和任务执行时间计算步骤，直至第 N 个任务。其中任务 j_i 就是任务序列中第 i 个任务，为便于描述，下面用 i 代表任务 j_i。则对任务 i 进行执行时间编排的步骤如下：

(1)获取任务序列中任务 i 所有的可用时间窗集合 W_i，并设置一个集合 W_i' 用于表示已被判定为不可成功编排的可用时间窗集合，初始化为 $W_i'=\varnothing$。

(2)从任务 i 的剩余可用时间窗集合 W_i-W_i' 中，选出一个开始时间为最小的可用时间窗 w，记录该可用时间窗的开始时间 a_{ik}^w 和结束时间 b_{ik}^w，以及此链

路 k 所使用的用户航天器和中继卫星。

(3)若该任务 i 是任务序列中的第 1 个任务，获取任务 i 的持续时间 d_i。若 $a_{ik}^w + d_i \le b_{ik}^w$ 成立，则记录任务的开始执行时间为 $t_i = a_{ik}^w$，并记录为任务 i 服务的用户航天器和中继卫星即为链路 k 所在用户航天器和中继卫星，并标记任务 i 调度成功；若 $a_{ik}^w + d_i \le b_{ik}^w$ 不成立，则更新 $W_i^{'} = W_i^{'} \cup \{\omega\}$，并返回(2)继续寻找。

(4)若该任务 i 不是任务序列中的第 1 个任务，则获取任务 i 的持续时间 d_i。根据(2)中记录的链路 k 所在的用户航天器和中继卫星，从任务序列中已成功调度且在解中的排序中位于任务 i 之前的任务中，判断是否存在一个或多个任务，其使用的用户航天器或中继卫星与链路 k 所在的用户航天器或中继卫星相同。若不存在这样的任务，且 $a_{ik}^w + d_i \le b_{ik}^w$，则记录任务 i 的开始执行时间为 $t_i = a_{ik}^w$，记录为任务 i 服务的用户航天器和中继卫星即为链路 k 所在用户航天器和中继卫星，并标记任务 i 调度成功，否则更新 $W_i^{'} = W_i^{'} \cup \{\omega\}$，并返回(2)，原因是此时链路可用时间窗的时间小于任务持续时间。若存在这样的任务，则需要判断是否存在用户航天器或中继卫星资源冲突。具体做法是：根据任务执行时间的先后顺序，从这样的任务中找出执行时间为最后的一个任务 i'，并得到任务 i' 的执行时间 $t_{i'}$ 和持续时间 $d_{i'}$，存在以下两种情况来计算任务可能的开始执行时间：若 $t_{i'} + d_{i'} \le a_{ik}^w$ 且 $a_{ik}^w + d_i \le b_{ik}^w$，则记录任务 i 的开始执行时间为 $t_i = a_{ik}^w$；若 $a_{ik}^w \le t_{i'} + d_{i'}$ 且 $t_{i'} + d_{i'} + \varepsilon_{ii'k} + d_i \le b_{ik}^w$，其中 $\varepsilon_{ii'k}$ 为任务 i 与任务 i' 之间的切换时间，则得到的任务 i 的开始执行时间为 $t_i = t_{i'} + d_{i'} + \varepsilon_{ii'k}$，得到任务 i 执行时间之后，并记录链路 k 所在的用户航天器和中继卫星，标记该任务调度成功；若以上两种情况均不成立，则更新 $W_i^{'} = W_i^{'} \cup \{\omega\}$，并返回(2)继续寻找。

(5)在选择可用时间窗的过程中，若出现多个可用时间窗的开始时间相同则随机选择一个可用时间窗进行进一步判断；若遍历完该任务 i 所有的可用时间窗也没有找到一个可用时间窗为其服务，则标记该任务调度失败。

解的初始化：设置算法种群的规模(m)，随机生成 m 组不同的任务序列，依次对每组任务序列按照上述步骤进行任务编排，求出对应的调度方案，即 m 组解，并计算各解的适应度函数值。

3.2.4 解的邻域操作 在 ABC 算法搜索过程中，为了判断一个解的附近是否存在一个比当前解更优的解，需要通过某种方法获取该解，该方法称为邻域操作方法，得到的解称为邻域解。ABC 算法中解的邻域操作(包括 3.2.5 节介绍的二次邻域操作)是一个局部寻优的过程，所以选择一个较好的邻域操作对于提高算法性能具有积极意义。通过数据仿真测试，本文采用选择一个任务随机插入的邻域操作方法。具体操作如下：从当前解中抽取一个任务，然后随机插入一个位置，这样得到的解即为当前解的邻域解，图 3 表示了该邻域操作方法(以编号为 0～9 的 10 个任务为例)。对于邻域操作产生的新解，我们可以按照第 3.2.3 节中介绍的任务编排方法，对从任务插入位置开始的所有任务重新进行任务编排。图 4 和图 5 表示其它两种常见的操作方法(选择两个任务交换位置法和选择 k 个任务随机插入法)。

3.2.5 解的二次邻域操作 为了能够进一步提高算法找到最优解的速度，基于最优解以比较大的概率落在较优解附近的特点，我们引入了一个基于适应度值的二次邻域操作过程，即从种群中选择一些较优解，然后判断这些解的附近是否存在一个更优的邻域解。本文采用锦标赛的选择方式来选择较优的解来进行二次领域操作。即从该种群中随机抽取两个解(x_p , x_q)，计算它们对应的适应度值($f(x_p)$, $f(x_q)$)。如果 $f(x_p) > f(x_q)$，选择 x_p；否则选择 x_q。

3.2.6 阈值 Lf 以及局部最优值 mf 在 ABC 算法寻优过程中，解的邻域操作以及解的二次邻域操作是一个局部寻优的过程，易陷入局部最优出现算法收敛早熟情况，为此 ABC 算法中定义了一个特有的参数用于跳出陷入局部最优情况，即阈值 Lf 。算法规定若一个解连续经过多次迭代仍然没有找到更好的解，为了防止迭代陷入局部最优，该解将被丢弃，重新随机生成一个新解代替。算法中通过阈值 Lf 来决定该解是否被丢弃(亦即若连续经过了 Lf 次迭代，该解仍未找到更优的邻域解，则放弃该解)。除此以外，为了防止在算法迭代过程中，局部最优解被丢弃，因此保留算法所有迭代过程出现过的最优解，算法中使用局部最优值 mf 保留迭代过程中最好的解和对应的任务编排。

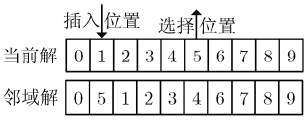

图 3 选择一个任务随机插入

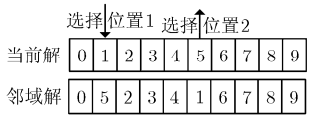

图 4 选择两个任务交换位置

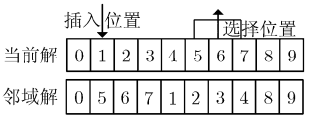

图 5 选择 k(k=3)个任务随机插入

3.2.7 ABC 算法　应用 ABC 算法进行中继卫星任务调度的步骤如下：

(1)初始化。设定算法的参数：任务数 N，种群的规模 m，阈值 Lf，二次邻域搜索循环次数 U，迭代次数 L 。按照第 3.2.3 介绍的随机生成 m 个初始解 x_i ($i=1,2,\cdots,m$)，并记录每个解的适应度函数(f)值。设置跟踪每个解 x_i 的变化的变量参数 l_i ($i=1,2,\cdots,m$)，初始化 $l_i=0$, l_i 记录了解 x_i 经过了连续多次邻域搜索，仍未找到更优的解的次数。设置每个解的邻域解的集合 G_i ($i=1,2,\cdots,m$)，并初始化 $G_i=\varnothing$ 。初始化局部变量最优解 mf，令 f(mf)=0 。令全局循环变量 $v=1$ 。

(2)按照第 3.2.4 节所介绍的邻域操作方式，为种群中每个解生成一个邻域解 $x_i \to \tilde{x}_i$ 。如果 $f(\tilde{x}_i) > f(x_i)$，则设置 $x_i=\tilde{x}_i, l_i=0$；否则 $l_i=l_i+1$ 。

(3)根据解的二次邻域搜索循环次数 U，进行 U 次二次邻域搜索操作。每次二次邻域搜索操作如下：首先根据第 3.2.5 节介绍的锦标赛的选择方式为每次二次邻域搜索得到一个较优解，然后采用第 3.2.4 节介绍的邻域操作方式为每次得到的较优解生成一个对应的邻域解 $\tilde{x}_i$，最后更新该较优解的邻域解集合 $G_i=G_i\cup\{\tilde{x}_i\}$，按照这样的操作方式，依次循环操作 U 次。

(4)根据(3)得到的解邻域集合，对每个解的邻域集合进行判断。若某个解(x_i)的邻域集合是非空的，即 $G_i\neq\varnothing$，则从该解的邻域解集合中选择一个最好的邻域解，即满足 $\hat{x}_i\in G_i, f(\hat{x}_i)=\max_{\hat{x}_i\in G_i}\{f(\hat{x}_i)\}$，然后作如下判断：如果 $f(\hat{x}_i)>f(x_i)$，则设置 $x_i=\hat{x}_i$, $l_i=0$；否则 $l_i=l_i+1$，并清空集合 $G_i=\varnothing$ 。

(5)记录本次迭代中出现的最优解 x_i 。如果 $f(x_i) > f$(mf)，那么更新 mf $=x_i$ 。

(6)判断每个解连续无更新邻域搜索次数是否超过阈值 Lf 。如果 $l_i\geq$ Lf，则放弃 x_i，随机生成一个新解加入种群，且置 $l_i=0$ 。更新变量 $v=v$+1 。若 $v=$ L，算法运行结束；否则转到(2)。

(7) mf 即为本次算法得到的最优解，其对应的调度方案即为最优调度方案。

4　仿真及结果分析

4.1 仿真场景

考虑中继卫星的单址天线的任务调度问题 [2)]，通信链路为前向链路，仿真时间为 00:00:00 ~ 23:59:59。从卫星工具箱(Satellite Tool Kits, STK)[15]中导出一颗中继卫星(TDRS-1)和 4 颗用户航天器(ALOS, JB-3 2, NAVSTAR 58, YAOGAN 4)，它们的轨道参数如表 1 所示。根据中继卫星和各用户航天器的轨道参数，可以计算出中继卫星与各用户航天器之间的可见时间窗[16]。表 2 列出了中继卫星 TDRS-1 与用户航天器 ALOS 之间可见时间窗，其它用户航天器与中继卫星之间的可见时间窗就不再一一列出。假设有 20 个任务申请服务，各个任务的申请服务请求的约束条件如表 3 所示，任务编号记为 Task1,Task2,$\cdots$,Task20[2)]。根据用户提交的任务约束、中继卫星和用户航天器之间的可见时间窗，可以得到任务的可用时间窗。表 4 显示了任务(Task1)的可用时间窗，其它的任务可用时间窗就不再一一列出。

4.2 结果分析

利用基于 ABC 算法得到的最优调度结果如表 5，表 6 所示(种群规模 m =30, 阈值 Lf =200, 迭代次数 L =1000，二次邻域搜索循环次数 U =30，选择一个任务随机插入的邻域操作方式)。其中表 5 是调度成功的任务情况表，表 6 是调度失败的任务情况表。在表 6 指明的任务调度失败的原因中，时间冲突是指中继卫星与用户航天器在这段时间内不可见，即不能进行数据中继；资源冲突是指该任务抢占卫星资源(中继卫星、用户航天器)失败。

基于 ABC 算法的调度结果分析如下：(1)调度的目标是完成尽可能多任务，且优先级高的任务优先安排调度，考虑中继卫星单址天线任务调度如 Task19 调度失败，Task18, Task4, Task14 调度成功，因为 Task19 抢占资源失败。(2)任务的执行必须在中继卫星和用户航天器的可见时间窗内，且也必须满足用户提交的时间约束，如 Task16 由于在该段时间内中继卫星与用户航天器不能彼此可见，所以 Task16 调度失败。(3)中继卫星与用户航天器之间可能具有多个可见时间窗，但任务的执行只能在一个可见时间窗内完成，因此当时间窗长度小于任务的持续时间长度时应考虑其它的时间窗，且同一时刻一颗卫星最多只能为一个任务服务，Task13 和 Task17 因为竞争资源失败未能调度成功。

接下来将基于 ABC 算法与基于有效基因路径表示的改进遗传算法[6]调度结果作比较，有关有效基因路径表示的改进遗传算法描述和操作请参考文献[6]。表 7(迭代次数为 1000、种群规模为 30)显示了

[2)]本节仿真中只考虑单个中继卫星在同一时刻最多服务一个用户的情形。如若中继卫星有多天线(假设天线数为 k)，在本文的中继卫星任务调度模型中，通过将其视为 k 个中继卫星，可以很容易得到最优调度方案。

表 1 卫星的轨道参数

轨道参数	TDRS-1	ALOS	JB-3 2	NAVSTAR 58	YAOGAN 4
轨道偏心率	0.000000	0.000129	0.000513	0.007706	0.001523
轨道长半轴(km)	42166.300	7063.784	6813.446	26558.528	7023.275
轨道倾角 (°)	0.083	98.156	97.028	55.951	97.873
近地点角 (°)	0.000	103.221	98.901	298.298	119.840
升交点赤经 (°)	88.184	184.813	114.878	25.940	187.747
平均近点角 (°)	282.837	256.915	261.279	60.940	240.434
平均运动(revs/d)	1.0030	14.5957	15.3049	2.0057	14.7548

表 2 中继卫星 TDRS-1 与用户航天器 ALOS 之间的可见时间窗

时间窗口	开始时间	结束时间	持续时间
1	04:01:09	04:59:34	00:58:25
2	05:38:56	06:37:51	00:58:55
3	07:15:29	08:17:18	01:01:49
4	08:48:35	12:34:46	03:46:11
5	13:03:14	14:06:18	01:03:04
6	14:43:10	15:42:26	00:59:16
7	16:21:37	17:20:01	00:58:24
8	17:59:13	18:58:29	00:59:16
9	19:35:22	20:38:28	01:03:06
10	21:06:55	23:59:59	02:53:04

表 3 各个任务的参数

任务	优先级	持续时间(s)	最早开始时间	最晚结束时间	服务的卫星
Task1	2	3000	09:40:00	13:45:00	ALOS
Task2	3	2000	07:00:00	12:44:00	ALOS
Task3	3	2700	10:30:30	14:00:00	ALOS
Task4	1	2400	12:10:00	14:00:00	ALOS
Task5	4	2400	09:00:00	13:20:00	ALOS
Task6	2	1800	16:09:00	18:30:00	JB-3 2
Task7	5	2400	00:40:00	09:00:00	JB-3 2
Task8	2	3000	07:45:00	17:40:00	JB-3 2
Task9	5	1800	17:40:00	23:00:00	JB-3 2
Task10	6	2100	15:40:00	20:30:00	JB-3 2
Task11	7	4200	18:00:00	20:56:00	NAVSTAR 58
Task12	7	3600	07:40:00	21:40:00	NAVSTAR 58
Task13	8	3000	14:30:00	17:10:00	NAVSTAR 58
Task14	3	2700	12:10:00	15:47:00	NAVSTAR 58
Task15	5	5400	14:30:00	17:10:00	NAVSTAR 58
Task16	6	1800	07:00:00	08:30:00	YAOGAN 4
Task17	6	3000	04:34:00	18:00:00	YAOGAN 4
Task18	2	3300	10:23:00	14:41:00	YAOGAN 4
Task19	3	2400	10:23:00	14:41:00	YAOGAN 4
Task20	3	3000	10:00:00	21:00:00	YAOGAN 4

表 4 任务(Task1)的可用时间窗

可用时间窗	开始时间	结束时间	服务的卫星
1	09:40:00	12:34:46	ALOS
2	11:19:55	12:11:42	ALOS

表 5 基于 ABC 算法调度成功的任务情况

任务	优先级	执行任务的开始时刻	持续时间(s)	服务的卫星(用户航天器和中继卫星)	执行任务的先后顺序
Task7	5	04:42:20	2400	JB-3 2, TDRS-1	1
Task8	2	07:50:15	3000	JB-3 2, TDRS-1	2
Task2	3	08:48:35	2000	ALOS, TDRS-1	3
Task5	4	09:21:55	2400	ALOS, TDRS-1	4
Task1	2	10:01:55	3000	ALOS, TDRS-1	5
Task3	3	10:51:55	3000	ALOS, TDRS-1	6
Task18	2	11:36:55	3300	YAOGAN 4, TDRS-1	7
Task4	1	13:03:14	2400	ALOS, TDRS-1	8
Task14	3	13:43:14	2700	NAVSTAR 58, TDRS-1	9
Task20	3	14:48:45	3000	YAOGAN 4, TDRS-1	10
Task15	5	15:38:45	5400	NAVSTAR 58, TDRS-1	11
Task6	2	17:58:42	1800	JB-3 2, TDRS-1	12
Task11	7	18:28:42	4200	NAVSTAR 58, TDRS-1	13
Task10	6	19:38:42	2100	JB-3 2, TDRS-1	14
Task12	7	20:13:42	3600	NAVSTAR 58, TDRS-1	15
Task9	5	21:13:42	1800	JB-3 2, TDRS-1	16

表 6 调度失败的任务情况

任务	调度失败原因
Task16	时间冲突
Task13	资源冲突
Task17	资源冲突
Task19	资源冲突

表 7　ABC 算法与遗传算法[6]的比较

算法	Minimum[a]	Maximum[b]	Average[c]	算法运行平均时间(ms)	调度成功的平均任务数
ABC 算法	1208	1234	1223.7	27.859	16
遗传算法	997	1063	1051.9	68.372	12

ABC 算法与该遗传算法相比较的结果，其中 Minimum[a], Maximum[b], Average[c] 分别表示算法运行 20 次得到所有最优解中适应度值的最小值、最大值、平均值。从数据分析中可以看出，就中继卫星任务调度问题而言，ABC 算法能够以较少的运算时间得到更高的适应度函数值，亦即得到中继星资源使用率更高的任务调度方案，其效果相对文献[6]提出的算法较好。

5　结束语

根据中继卫星任务调度问题的特点，综合考虑中继卫星与用户航天器之间具有可见时间窗、提交的任务约束、中继卫星前向资源受限等约束条件，建立了一个调度约束规划模型，基于这个模型提出了一种基于 ABC 算法的中继卫星任务调度方法。仿真结果表明：ABC 算法能够有效解决中继卫星任务调度问题；通过与基于有效基因路径表示的改进遗传算法任务调度结果相比较，从调度成功的任务数、算法的运行时间以及适应度函数值 3 方面考虑，得之其性能比该改进的遗传算法较好。在将来，我们也将与其它启发式算法作比较。

参 考 文 献

[1] Teles J, Samii M V, and Doll C E. Overview of TDRSS[J]. *Advance in Space Research*, 1995, 16(12): 67–76.

[2] 王家胜. 中国数据中继卫星系统及其应用拓展[J]. 航天器工程, 2013, 22(1): 1–6.
Wang Jia-sheng. China's data relay satellite system and its application prospect[J]. *Spacecraft Engineering*, 2013, 22(1): 1–6.

[3] Rojannasoonthon S, Bard J F, and Reddy S D. Algorithms for parallel machine scheduling: a case study of the tracking and data relay satellite system[J]. *Journal of the Operation Research Society*, 2003, 54(8): 806–821.

[4] 方炎申, 陈英武, 顾中舜. 中继卫星调度问题的 CSP 模型[J].

国防科技大学学报, 2005, 27(2): 6–10.
Fang Yan-shen, Chen Ying-wu, and Gu Zhong-shun. CSP model of the relay satellite scheduling[J]. *Journal of National University of Defense Technology*, 2005, 27(2): 6–10.

[5] 陈理江, 武小悦, 李云峰. 基于时间灵活度的中继卫星调度算法[J]. 航空计算技术, 2006, 36(4): 48–51.
Chen Li-jiang, Wu Xiao-yue, and Li Yun-feng. Scheduling algorithm for relaying satellite based on temporal flexibility [J]. *Aeronautical Computing Technique*, 2006, 36(4): 48–51.

[6] Fang Yan-shen and Chen Ying-wu. Constraint programming model of TDRSS single access link scheduling problem[C]. 2006 International Conference on Machine Learning and Cybernetics (ICMLC), Dalian, 2006: 948–951.

[7] 赵卫虎, 赵静, 赵尚弘, 等. 微波与激光混合链路中继卫星动态调度快速启发式算法[J]. 中国激光, 2014, 41(9): 1–7.
Zhao Wei-hu, Zhao Jing, Zhao Shang-hong, *et al.*. Dynamic scheduling fast heuristic algorithm for data relay satellite with microwave and laser hybird links[J]. *Chinese Journal of Lasers*, 2014, 41(9): 1–7.

[8] Karaboga D. An idea based on honey bee swarm for numerical optimization[R]. Technical report TR06. Computer Engineering Department, Engineering Faculty, Erciyes University, 2005.

[9] Karaboga D and Basturk B. On the performance of artificial bee colony (ABC) algorithm[J]. *Applied Soft Computing*, 2008, 8(1): 687–697.

[10] Abu-Mouti F S and El-Hawary M E. Overview of artificial bee colony (ABC) algorithm and its applications[C]. 2012 IEEE International Systems Conference(SysCon), Vancouver, 2012: 1–6.

[11] Karaboga D, Gorkemli B, Ozturk C, *et al.*. A comprehensive survey: artificial bee colony (ABC) algorithm and applications[J]. *Artificial Intelligence Review*, 2014, 42(1): 21–57.

[12] Yi Yu-jiang and He Ren-jie. A novel artificial bee colony algorithm[C]. Intelligent Human-Machine Systems and Cybernetics (IHMSC), Hangzhou, 2014: 271–274.

[13] Kojima M, Nakano H, and Miyauchi A. An artificial bee colony algorithm for solving dynamic optimization problems [C]. Evolutionary Computation (CEC), Cancun, 2013: 2398–2405.

[14] Liang Yun-chia, Chen A H L, and Nien Yung-hsiang. Artificial bee colony for workflow scheduling[C]. 2014 IEEE Congress on Evolutionary Compution(CEC), Beijing, 2014: 558–564.

[15] STK User's Manual Version 6.0.1 for PCS[M]. USA, Analytical Graphics, Inc., 2005: 144–152.

[16] 李于衡, 孙恩昌, 易克初. 中继卫星与用户星双向跟踪关系及策略[J]. 西安电子科技大学学报(自然科学版), 2007, 34(1): 6–10.
Li Yu-heng, Sun En-chang, and Yi Ke-chu. On the tracking strategies and space geometrical relationship between a TDRS and user satellites[J]. *Journal of Xidian University*, 2007, 34(1): 6–10.

开彩红： 女，1982 年生，副教授，研究方向为无线通信与网络通信、网络体系结构和协议、网络系统性能分析与评价.

肖 瑶： 男，1990 年生，硕士生，研究方向为无线通信与网络通信.

方 青： 男，1972 年生，研究员级高级工程师，研究方向为卫星通信系统.