TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 15/20 pp843–862
DOI: 10.26599/TST.2023.9010064
Volume 29, Number 3, June 2024

A Fast Insertion Tabu Search with Conflict-Avoidance Heuristic for the Multisatellite Multimode Crosslink Scheduling Problem

Weiyi Yang, Lei He, Xiaolu Liu*, Weican Meng, and Yingwu Chen

Abstract: An agile earth-observing satellite equipped with multimode cameras capable of transmitting observation data to other satellites is developed to rapidly respond to requests with multiple observation modes. This gives rise to the Multisatellite Multimode Crosslink Scheduling (MMCS) problem, which involves allocating observation requests to agile satellites, selecting appropriate timing and observation modes for the requests, and transmitting the data to the ground station via the satellite communication system. Herein, a mixed integer programming model is introduced to include all complex time and operation constraints. To solve the MMCS problem, a two-stage heuristic method, called Fast insertion Tabu Search with Conflict-avoidance (FTS-C) heuristic, is developed. In the first stage, a conflict-avoidance insertion algorithm is designed to generate a high-quality initial solution by considering the requests transmission and download. Further, the tabu search-based second stage optimizes the initial solution. Finally, an extensive empirical study based on a real-world situation demonstrates that FTS-C can generate a solution with higher quality in less time than other state-of-the-art algorithms and the CPLEX solver.

Key words: earth observation satellites scheduling; tabu search heuristic; data transmission and download; mixed integer programming model

1 Introduction

Agile Earth Observation Satellites (AEOSs) are widely used for resource exploration, disaster surveillance, urban planning, and crop monitoring[1]. Current satellites, such as the high-resolution multimode observation satellite launched by China in 2020, are equipped with multimode cameras, enabling them to acquire images with different resolutions. Generally, a higher resolution leads to better image quality; however, it also requires longer observation time and larger onboard storage. Users tend to select images with higher resolutions, even when lower resolutions can meet their requirements. Therefore, with an increasing number of user requests, it is challenging for satellite operation management to satisfy more users with limited satellites[2].

Generally, the fulfillment of requests involves two steps: observation and download. Currently, most AEOSs are capable of transmitting observation data to ground stations through intersatellite links due to the growth of intersatellite link technology[3]. Therefore, the observation data can be downloaded to the ground more quickly, thereby shortening the response time. Consequently, another step is added to the process of fulfilling requests: observation, transmission, and download, as shown in Fig. 1. This development has increased the complexity of the effective management of data transmission across the AEOSs system[4].

• Weiyi Yang, Lei He, Xiaolu Liu, and Yingwu Chen are with College of Systems Engineering, National University of Defense Technology, Changsha 410000, China. E-mail: {yangweiyi15, helei, lxl_sunny, ywchen}@nudt.edu.cn.
• Weican Meng is with Beijing Institude of Tranking and Telecommunication Technology, Beijing 100000, China. E-mail: wwss_000@163.com.
* To whom correspondence should be addressed.
Manuscript received: 2023-03-06; revised: 2023-06-11; accepted: 2023-06-13

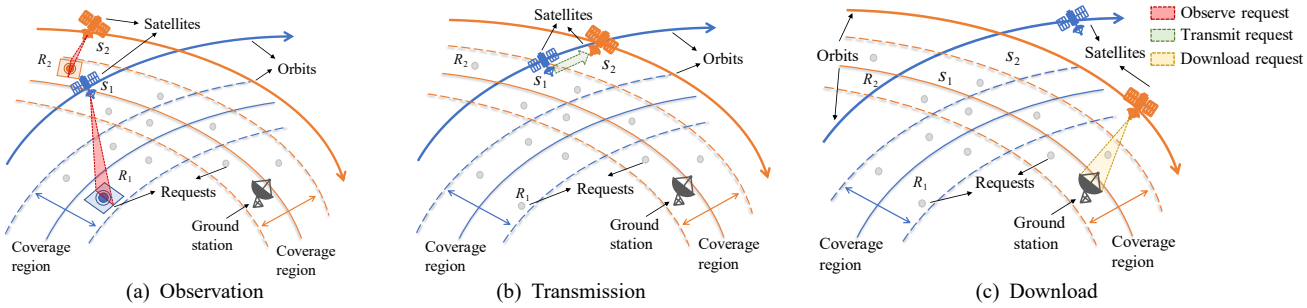


Fig. 1 Diagram of the process of fulfilling requests.

The aforementioned features of the satellite system give rise to a new scheduling problem owing to the selection of different observation modes and the transmission of observed data between the satellites[5]. There is a set of observation requests with predefined priority. The problem presented in this study involves allocating observation requests to satellites, selecting the timing and observation mode for the requests, and delivering the data generated to the ground through the satellite communication system. We name the problem the Multisatellite Multimode Crosslink Scheduling (MMCS) problem. The key word "crosslink" in the name of this problem emphasizes the intersatellite data transmission process.

Existing studies have widely examined the Multiple Earth Observation Satellite Scheduling (MEOSS) problem, which is similar to MMCS in terms of the scheduling process. Particularly, a set of weighted tasks is scheduled for the satellites in their visible time windows, which is subject to energy consumption restrictions. Therefore, we first review the MEOSS problem, which has been proven to be NP-hard[6]. A basic introduction can be found in Ref. [7]. MEOSS can often be modeled as a common problem structure, such as a 0-1 knapsack problem[8], a mixed integer model[9, 10], an original constraint satisfaction problem[11], or a model with multiobjective frameworks[12]. The metaheuristic method is a popular approach for solving the MEOSS problem, which is inspired by general phenomena or specific domain knowledge[13], including genetic algorithms[14], adaptive large neighborhood search[15], tabu search[16], and particle swarm optimization[17]. In addition, Lagrangian relaxation[18] and exact algorithms (such as branch and bound[19, 20], and dynamic programming methods[21]) have also been employed to simplify the problem-solving procedure and obtain an optimal solution. Notably, Wang et al.[22] formulated the MEOSS problem as an unrelated parallel machine scheduling problem, then developed a novel preprocessing algorithm to effectively reduce the problem size. Additionally, the introduction of deep learning techniques and the advent of the transformer network provide a fresh perspective for solving the online MEOSS problem[23]. However, in contrast to MMCS, storage capacity constraints have been ignored in some of the abovementioned studies. Additionally, few studies have formulated task observation, transmission, and download as a unified model due to the limitations of satellite technology.

Despite similarities with MEOSS, MMCS has some extra features in its problem objectives, decisions, and constraints. First, MMCS schedules requests in different observation modes. Particularly, in addition to scheduling the time of observation, MMCS decides the mode of observation, increasing the variable space of the model. Chen et al.[9] addressed the problem of selecting imaging types for different observation requests and developed an improved Mixed Integer Linear Programming (MILP) model. Second, new constraints, including task dependencies, precedence, and time-dependent transition time, are modeled in MMCS, which are often overlooked in many recent multiorbit satellite scheduling formulations[1, 7, 24, 25]. Third, the observation, transmission, and download operations in MMCS are integrated into a unified model, considering the time dependencies and the coupled relationships between them. Therefore, the number of decision variables and the structural complexity of the unified model are increasing.

Only a handful of researchers have investigated task observation and download scheduling. Marinelli et al.[18] investigated satellite range scheduling with resource constraints and proposed a time-indexed 0-1 programming formulation. Although a Lagrangian relaxation heuristic was introduced to reduce variable

spaces in their model and solve the problem, satellite energy and memory storage consumption were not considered. Hu et al.[26] developed a B&P algorithm to compute optimal integer solutions for the MILP formulation which they proposed for the satellite observation and download scheduling problem. However, these methods can only be used to solve small instances due to time complexity limitations. Wang et al.[25] proposed a nonlinear programming model to address the integrated Earth Observation Satellites (EOSs) scheduling problem, taking energy and memory storage into consideration. Based on the research of Wang et al.[25], Sarkheyli et al.[16] constructed a mixed integer programming model with cloud coverage constraints and task revisit properties. Similarly, Xiao et al.[27] studied the uncertainties of weather during the observation process and proposed a two-stage flow scheme to address it. Berger et al.[13] proposed a quadratically constrained program solver relying on the approximate objective function proposed by Wang et al.[25] and added some new operational constraints. Zhang and Xing[28] and Chen et al.[29] adopted improved genetic algorithms for solving the satellite observation and downlink scheduling problem. Among them, Zhang and Xing[28] further incorporated the real-transmit mode, in which the satellite downloads observation data immediately when the observation time window overlaps with the downlink time window. However, as opposed to MMCS, these studies ignored data transmission across satellites.

Zhu et al.[30] and Kennedy[31] considered data transmission between satellites in their integrated EOSs scheduling problem. Zhu et al.[30] utilized a relay satellite to transmit data, which serves as a receiver (from a satellite) and a transmitter (to a ground station). A two-phase genetic annealing algorithm was designed to optimize the proposed mixed integer programming model. Rather than using a relay satellite, Kennedy[31] demonstrated and verified the feasibility of data transmission through satellite crosslinks, their data transmission settings are very similar to those for MMCS, where satellites are capable of imaging, receiving, and transmitting data to other satellites, and downloading data to the ground. However, to the best of our knowledge, none of these studies examined agile satellites in the integrated satellite scheduling problem.

Several studies employed exact optimization algorithms for the MEOSS problem, such as branch and bound[19, 20], Russian Doll search[32], and dynamic programming methods[21]. These exact optimization algorithms are more suitable for the small-scale MEOSS problem than for the MMCS problem, which has complex time coupling constraints. Therefore, heuristics and metaheuristics are ideal alternatives. Local search approaches comprise commonly employed heuristics. The critical issues with these algorithms are the use of iterations and the effect of neighborhood structures in the search space. Liu et al.[15] developed an Adaptive Large Neighborhood Search (ALNS) to search for a conflict-free solution and a time slack strategy to handle the transition time for tasks in a single satellite scheduling problem. This method was later adapted to solve a multisatellite scheduling problem by He et al.[33] Based on these studies, He et al.[10] designed an algorithm that is a novel hybridization of ALNS and other local search algorithms. However, this study did not consider data transmission across satellites and the selection of the observation mode, which are common restrictions in reality. In addition, there exist limited up-to-date studies related to Tabu Search (TS), which has strong convergence and has shown great success in dealing with complex time constraints[34]. In particular, TS outperforms other metaheuristics like ALNS, because it prevents ineffective search moves, which is crucial when each search has a substantial computational cost in problems with complex time constraints like MMCS. Therefore, this study considers a TS-based algorithm and adopts relevant state-of-the-art techniques to solve the MMCS problem.

Therefore, this study investigates a novel problem involving multiple mode selection and data transmission across a multisatellite system, which differs from state-of-the-art integrated EOSs scheduling[13, 16, 35]. We propose an effective heuristic that includes a conflict-avoidance insertion algorithm that considers request transmission and download to generate a high-quality initial solution. Further, an improved algorithm based on TS is designed to optimize the solution with an acceptable runtime. Numerical experiments demonstrate the effectiveness of the algorithm in terms of solution quality and efficiency against state-of-the-art algorithms.

The remainder of this study is organized as follows: Section 2 describes MMCS and its mathematical model. Section 3 outlines a Fast insertion TS with

Conflict-avoidance (FTS-C) heuristic. Section 4 presents the experimental results, and Section 5 presents our conclusions.

2 Problem Description

2.1 Problem definition

In the MMCS problem, there are a set of requests with a predefined priority and a set of satellites with intersatellite links to fulfill the requests and download them to the ground station.

The properties of each request that must be considered are as follows:

• For each request, there is a user-specific priority that describes the importance and urgency of that request. In addition, some requests must follow a sequence constraint predefined by the users. For example, some requests must start only after the other requests are fulfilled.

• The request is fulfilled after it is downloaded to the ground station. Note that the request can be downloaded by the satellite observing it and other satellites. When the request is downloaded by other satellites the transmission of the generated data between the observing and downloading satellites occurs via the intersatellite link. Particularly, as shown in Fig. 1a, satellite S_1 observes request R_1, while satellite S_2 observes request R_2. Further, satellite S_1 transmits the data generated for request R_1 to satellite S_2, as shown in Fig. 1c. Subsequently, satellite S_2 downloads the data generated for R_2 and the data generated for R_1 to the ground station in Fig. 1c. This process of fulfilling a request includes three types of tasks: observation, download, and transmission.

2.2 Assumptions

As the MMCS problem is complicated owing to many constraints, the following assumptions and simplifications are made:

• Each satellite has a maximal accumulated operation times and a maximal onboard storage per satellite orbit. The maximal accumulated operation times is a simplification of the satellite's onboard energy constraints. In addition, each satellite has onboard data storage, which is consumed with every observation and is relieved by every download and transmission. The data storage consumed must be within the capacity.

• MMCS considers a satellite with several observation modes. The selection of different modes for a request will influence the revenue obtained and the memory storage consumption.

• MMCS assumes that the intersatellite link is fixed, which means that the task has only one transmission route once the observing satellite and the downloading satellite are determined.

2.3 Notations

2.3.1 Sets and parameters

The sets and parameters used for the problem formulation are listed as follows:

(1) $R = 1, 2, \ldots, N_R$ is the set of requests with $|R| = N_R$.

(2) $T = 1, 2, \ldots, N_T$ is the set of tasks with $|T| = N_T = 4N_R$. From the task set T, $T^o = 1, 2, \ldots, N_R$ is the set of observation tasks, $T^d = N_R + 1, N_R + 2, \ldots, 2N_R$ is the set of download tasks and $T^t = 2N_R + 1, 2N_R + 2, \ldots, 4N_R$ is the set of transmission tasks. The task set is the union of the observation set, the download set and the transmission task set, which can be expressed as $T = T^o \cup T^d \cup T^t$.

(a) d_i^{unit} is the unit execution duration of task i.

(b) ω_i^{unit} is the unit execution revenue of task i, where $\omega_i^{\text{unit}} \in [1,10]$. One of the objectives is to maximize the total revenue from all the observation tasks.

(c) v_i^{unit} is the unit consumed storage for task i. Note that the value of unit consumed storage v_i^{unit} for the observation task i is positive, while the value of unit consumed storage for the corresponding download task $i+N_R$ is negative. This can be expressed by $v_i^{\text{unit}} = -v_{i+N_R}^{\text{unit}}$

(d) δ is a coefficient that reflects the linear relationship between the observation mode and the task unit duration d_i^{unit}.

(e) ε is a coefficient that reflects the linear relationship between the observation mode and the task unit consumed storage v_i^{unit}.

(3) $S = 1, 2, \ldots, N_S$ is the set of satellites, and k is the index for the satellites in the set of satellites S. For satellite k, the following attributes are defined:

(a) E_k represents the maximal accumulated operation time per orbit for satellite k.

(b) C_k is the maximal storage capacity of satellite k.

(c) α_k is the dummy source task for satellite k.

(d) β_k is the dummy sink task for satellite k.

(e) a_1, a_2, a_3, and a_4 are four different transition

angular velocities for different transition angles.

(4) $W_{ik} = 1, 2, \ldots, N_{W_{ik}}$ is the set of visible time windows between task i and satellite k with $|W_{ik}| = N_{W_{ik}}$, and j is the index for the time window in the set of visible time windows W_{ik}. For each time window j in W_{ik}, the following attributes are defined:

(a) b_{ijk} is the start time of time window j between task i and satellite k.

(b) e_{ijk} is the end time of time window j between task i and satellite k.

(c) a^{γ}_{ijk}, a^{π}_{ijk}, a^{φ}_{ijk}, b^{γ}_{ijk}, b^{π}_{ijk}, and b^{φ}_{ijk} are the parameters of the functions of angles.

(5) $U = 1,2,\ldots,N_U$ is the set of orbits with $|U| = N_U$, and u is the index for the set of orbits. For each Orbit u, the following attribute is defined:

q^{u}_{ijk} is a binary parameter that equals 1 only if task i has a visible time window j on satellite k during orbit u; otherwise, $q^{u}_{ijk} = 0$.

(6) $M = 1, 2, \ldots, N_M$ is the set of observation modes with $|M| = N_M$, where m is the index for the set of observation modes. The task revenue, task duration, and task consumed storage are linearly related to the observation modes selected by the requests. Particularly, if task i is selected to be observed using observation mode m, then its task revenue, task duration and task consumed storage are given by $m \cdot \omega_i^{\text{unit}}$, $m \cdot \delta \cdot d_i^{\text{unit}}$, and $m \cdot \varepsilon \cdot v_i^{\text{unit}}$, respectively.

2.3.2 Variables

The variables used for problem formulation are listed as follows:

(1) x_{ijkm} is a binary variable that equals 1 only if task i and visible time window j are selected by satellite k using the observation mode m. All tasks belonging to the same request have the same observation mode.

(2) l_i, a decision variable of the model, is the start time of task i. The aim of solving this problem is to determine the values of x_{ijkm} and l_i for each task i.

The following defines several variables that can be determined if x_{ijkm} and l_i are given:

- v_i is the real storage consumed by the selected task i, which is given by the product of v_i and the selected observation mode m. It is calculated by $v_i = \sum_{m\in M} \sum_{k\in S} \sum_{j\in W_{ik}} m \cdot x_{ijkm} \cdot \varepsilon \cdot v_i^{\text{unit}}$.
- d_i is the real task duration of the selected task i, which is given by the product of d_i^{unit} and the selected observation mode m. It is calculated by $d_i = \sum_{m\in M} \sum_{k\in S} \sum_{j\in W_{ik}} m \cdot x_{ijkm} \cdot \delta \cdot d_i^{\text{unit}}$.
- z_{ik} is the onboard data volume remaining on the satellite k after processing task i.
- $y_{ii'k}$ is a binary variable that equals 1 only if task i and task i' are executed on the same satellite k, and task i is the immediate precedent task of task i'.
- γ_i, π_i, and φ_i are the roll angle, pitch angle, and yaw angle for task i, respectively. They are calculated by $\gamma_i = a^{\gamma}_{ijk} \cdot l_i + b^{\gamma}_{ijk}$, $\pi_i = a^{\pi}_{ijk} \cdot l_i + b^{\pi}_{ijk}$, and $\varphi_i = a^{\varphi}_{ijk} \cdot l_i + b^{\varphi}_{ijk}$, where the task start time l_i is in the time window j of satellite k.
- $\theta_{ii'}$ is the total angle to transition between two adjacent observations for task i and task i'. It is calculated by $\theta_{ii'} = |\gamma_i - \gamma_{i'}| + |\pi_i - \pi_{i'}| + |\varphi_i - \varphi_{i'}|$.
- $s_{ii'}$ is the transition time between two adjacent observations for task i and task i'. This is the time taken by the satellite to transition from the observation angle for task i to task i'. It is determined by the following piecewise linear function:

$$s_{ii'} = \begin{cases} 10 + \dfrac{\theta_{ii'}}{a_1}, & \theta_{ii'} \leqslant 15; \\ 15 + \dfrac{\theta_{ii'}}{a_2}, & 15 \leqslant \theta_{ii'} \leqslant 40; \\ 20 + \dfrac{\theta_{ii'}}{a_3}, & 40 \leqslant \theta_{ii'} \leqslant 90; \\ 25 + \dfrac{\theta_{ii'}}{a_4}, & \theta_{ii'} > 90 \end{cases} \tag{1}$$

2.4 Mathematical formulation

2.4.1 Objective function

The optimization objective G is composed of two parts, the total task revenue G_r and the average response time G_s. The total task revenue is the sum of the real task revenue that is the product of ω_i^{unit} and the selected observation mode m. The response time of request i is defined as the difference between the execution start time for the observation task i and download task $i + N_R$. The objective G_{s} is to minimize the average value of all selected requests,

$$G_r = \max \sum_{m\in M} \sum_{i\in T^o} \sum_{k\in N_S} \sum_{j\in W_{ik}} m \cdot x_{ijkm} \cdot \omega_i^{\text{unit}} \tag{2}$$

$$G_s = \min \frac{\sum\limits_{i\in T^o} (l_{N_R+i} - l_i)}{N_R} \tag{3}$$

2.4.2 Constraints

The constraints for this problem are listed below.

(1) Uniqueness observation constraint

$$\sum_{m\in M} \sum_{k\in S} \sum_{j\in W_{ik}} x_{ijkm} \leqslant 1, \quad i \in T^o \tag{4}$$

Constraint in Formula (4) states that each request is observed at most once, regardless of what kind of observation mode is used.

(2) Sequence and precedence constraint

$$\sum_{r\in\beta_k\cup T^o,\, r\neq i} y_{irk}=\sum_{m\in M}\sum_{j\in W_{ik}} x_{ijkm},\ \ i\in\alpha_k\cup T^o,\ k\in S \tag{5}$$

$$\sum_{r\in\alpha_k\cup T^o,\, r\neq i} y_{rik}=\sum_{m\in M}\sum_{j\in W_{ik}} x_{ijkm},\ \ i\in\beta_k\cup T^o,\ k\in S \tag{6}$$

Constraints in Eqs. (5) and (6) state that there can only be one task directly preceding and following each selected task on a satellite, respectively.

(3) Time window constraint

$$\begin{aligned}&b_{ijk}\leqslant l_i+H\cdot\left(1-\sum_{m\in M}x_{ijkm}\right),\\&l_i\leqslant e_{ijk}+H\cdot\left(1-\sum_{m\in M}x_{ijkm}\right),\\&i\in T,\ j\in W_{ik},\ k\in S\end{aligned} \tag{7}$$

where H is a suitable large number.

Constraints in Formula (7) provides the time bound for a selected task.

$$l_i+d_i+s_{i\,(i+N_R)}\leqslant l_{i+N_R},\ \ i\in T^o \tag{8}$$

$$l_i+d_i+s_{i\,(2N_R+2i-1)}\leqslant l_{2N_R+2i-1},\ \ i\in T^o \tag{9}$$

$$l_{2N_R+2i}+d_{2N_R+2i}+s_{(2N_R+2i)(i+N_R)}\leqslant l_{i+N_R},\ \ i\in T^o \tag{10}$$

Constraints in Formulas (8)–(10) are precedence constraints that restrict some tasks to follow a user-defined order. Constraint in Formula (8) requires the end time of observation task i to be earlier than the start time of the corresponding download task $i+N_R$. Constraints in Formulas (9) and (10) mandate that the transmission tasks should be executed between observation task i and the corresponding download task $i+N_R$.

(4) Transition time constraint

$$l_i+d_i+s_{ii'}+H\cdot(y_{ii'k}-1)\leqslant l_{i'},\ \ i,i'\in T,\ k\in S \tag{11}$$

Constraint in Formula (11) requires that the time between any two tasks must be enough for the satellite to finish its transition.

(5) Maximal slew times constraint

$$\sum_{m\in M}\sum_{i\in T}\sum_{j\in W_{ik}} x_{ijkm}\cdot q^u_{ijk}\leqslant E_k,\ \ k\in S,\ u\in U \tag{12}$$

Constraint in Formula (12) restricts the maximal operation times, which bounds the operation times for every satellite in each obit.

- **Download task selection constraint**

$$\sum_{k\in S}\sum_{j\in W_{ik}} x_{(i+N_R)jkm}=\sum_{k\in S}\sum_{j\in W_{ik}} x_{ijkm},\quad i\in T^o,\ m\in M \tag{13}$$

Constraint in Eq. (13) mandates that the download task $i+N_R$ will be executed if its corresponding observation task i is selected, and the observation modes of the tasks belonging to the same request are the same.

(6) Data storage constraint

$$\begin{aligned}&z_{ik}+v_{i'}+H\cdot(1-y_{ii'k})\geqslant z_{i'k},\\&z_{ik}+v_{i'}+H\cdot(y_{ii'k}-1)\leqslant z_{i'k},\\&z_{ik}\leqslant C_k,\ \ i\in T,\ k\in S\end{aligned} \tag{14}$$

Constraint in Formula (14) mandates that the satellite storage for the entire period cannot exceed the maximal storage capacity of the satellite.

(7) Transmission task constraint

When an observation task is not downloaded by the satellite observing it, a set of transmission tasks over the intersatellite link are executed. The corresponding transmission tasks of observation task i can be indexed by $2N_R+2i-1$ and $2N_R+2i$ in task set T, where task $2N_R+2i-1$ corresponds to the satellite sending the request data and $2N_R+2i$ corresponds to the satellite receiving the request data, for the transmission task $2N_R+2i$ and $2N_R+2i-1$,

$$\begin{aligned}&x_{(2N_R+2i-1)jkm}=\\&\quad\max\left(0,\sum_{j\in W_{ik}}x_{ijkm}-\sum_{j\in W_{ik}}x_{(N_R+i)jkm}\right),\\&x_{(2N_R+2i)jk'm}=\\&\quad\max\left(0,\sum_{j\in W_{ik'}}x_{(N_R+i)jk'm}-\sum_{j\in W_{ik'}}x_{ijk'm}\right),\\&i\in T^o,\ k,\ k'\in S,\ k\neq k'\end{aligned} \tag{15}$$

The first equation in Eq. (15) mandates that the transmission task $2N_R+2i-1$ will not be executed on satellite k if its corresponding observation task i is not selected or downloaded by satellite k that observed it. The second equation in Eq. (15) expresses the same restrictions.

$$l_{2N_R+2i-1}=l_{2N_R+2i},\quad i\in T^o \tag{16}$$

Constraint in Eq. (16) mandates the transmission tasks $2N_R+2i-1$ and $2N_R+2i$ for the observation task i must be simultaneous.

(8) Variable domains constraint

$$x_{ijkm} \in \{0,1\}, \quad i \in T, j \in W_{ik}, k \in S, m \in M \tag{17}$$

$$y_{ii'k} \in \{0,1\}, \quad i, i' \in T, k \in S \tag{18}$$

Constraints in Formulas (17) and (18) are related to variable domains.

3 Methodology

Herein, a two-stage method is proposed to solve the MMCS problem. In the first stage, a conflict-avoidance insertion algorithm is designed, taking request transmission and download into account to generate a high-quality initial solution. Further, the second stage based on TS aims to further optimize it. We name this method the FTS-C heuristic.

3.1 Conflict-avoidance insertion algorithm

Herein, we first propose the conflict-avoidance insertion algorithm, taking request transmission and download into account to construct a high-quality initial solution. There are several difficulties:

(1) The maximal slew times and the data storage constraints mandate that there is a probability that some requests will not be scheduled, subsuming this problem to an oversubscribed scheduling problem.

(2) The time coupling among the observation, download, and transmission tasks of a request is tight, which complicates the local adjustment of the scheduled task.

(3) The obtainable execution revenue and the storage consumption for a request are uncertain until its observation mode is determined, making the problem complex.

(4) The data storage would run out quickly if continuous observations are performed. Therefore, timely transmission or downloading of the observed request is crucial.

The first difficulty is addressed by the proposed request ranking heuristic in Section 3.1.2. The second difficulty is handled by the designed time slack insertion strategy in Section 3.1.3, particularly for tasks that have close time coupling with each other. In Section 3.1.4, we address the last two difficulties by proposing the position selection heuristic for the observation and download positions. The conflict-avoidance insertion algorithm framework is shown in Section 3.1.5.

3.1.1 Characteristics for time windows

Herein, we define some characteristics of time windows, which are used in the conflict-avoidance insertion algorithm that will be introduced later.

The first characteristic is the size of a time window set. Generally, the size of a time window set $W_l = \cup_j w_{l,j}$ can be measured using two indicators: (1) the number of the time windows in window set $|W_l|$, and (2) the sum of all time window duration $T_{W_l} = \sum_j (e_{l,j} - b_{l,j})$, where $e_{l,j}$ and $b_{l,j}$ are the end time and start time of the time window $w_{l,j} \in W_l$, respectively. For a collection of the time window sets $W = \{W_1, W_2, \ldots, W_{N_W}\}$, the size of time window set W_l in W is given as follows:

$$\text{Size}\left(W_l, \bigcup_l W_l\right) = \frac{|W_l|}{\max\limits_l |W_l|} \cdot \frac{T_{W_l}}{\max\limits_l T_{W_l}} \tag{19}$$

Another important characteristic of time windows is the overlapping time. As shown in the Fig. 2, for two different time window sets $|W_l|$ and $W_{\hat{l}}$, we define the set of overlapping time windows as follows:

$$W_{l,\hat{l}}^{\text{overlap}} = \left\{ w_{l,\hat{l},j'}^{o} \middle| [b_{l,j}, e_{l,j}] \cap [b_{\hat{l},\hat{j}}, e_{\hat{l},\hat{j}}] \right\}, \quad \forall\, w_{l,j} \in W_l, \forall\, w_{\hat{l},\hat{j}} \in W_{\hat{l}} \tag{20}$$

where $e_{\hat{l},\hat{j}}$ and $b_{\hat{l},\hat{j}}$ are the end time and start time of time window $w_{\hat{l},\hat{j}} \in W$, respectively.

3.1.2 Request ranking heuristic

With limited accumulated operation times and onboard satellite resources, the order of requests to be inserted can directly influence the performance of the insertion algorithm. In general, a request with fewer and shorter feasible observation time windows is difficult to schedule. Hence, a higher priority should be given to schedule it. A Revenue-based Request ranking Heuristic that considers Observation and Download Flexibility (RRH-ODF) is designed by taking this into consideration. RRH-ODF evaluates the flexibility of a request with three indices: the revenue r_i, the observation scheduling flexibility OR_i and the download scheduling flexibility DR_i. First, OR_i is designed to reflect the difficulty of observing a request. We use the ratio of W_i to the entire set of all request

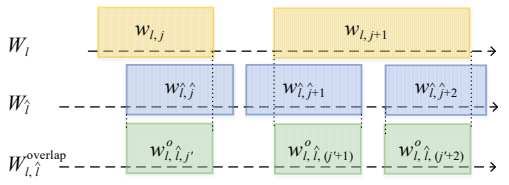


Fig. 2 Diagram of overlapping window set $W_{l,\hat{l}}^{\text{overlap}}$.

time windows $\bigcup_{i\in T^o} W_i$ to denote the flexibility OR_i, which is given by

$$\mathrm{OR}_i = \mathrm{Size}\left(W_i, \bigcup_{i\in T^o} W_i\right), \quad i \in T^o \tag{21}$$

Second, we account for the overlapping of the download time windows and observation time windows. This is because requests with extensive overlap between their observation and download time windows have a higher probability of being observed and downloaded by one satellite without any intersatellite transmission. The unnecessary intersatellite transmission would lead to a waste of time and memory storage resources. Hence, we define the overlap between download windows and request i as follows:

$$\mathrm{DR}_i = \mathrm{Size}\left(W^{\mathrm{overlap}}_{i,(i+N_R)}, \bigcup_{i\in T^o} W^{\mathrm{overlap}}_{i,(i+N_R)}\right), \quad i \in T^o \tag{22}$$

where the overlap window set $W^{\mathrm{overlap}}_{i,(i+N_R)}$ is composed of observation time window set $W_i = \bigcup_{k\in S} W_{ik}$ overlapped with the corresponding download time window set $W_{i+N_R} = \bigcup_{k\in S} W_{(i+N_R)k}$.

Clearly, a request should be scheduled and prioritized if it obtains a higher revenue ω_i^{unit}, smaller observation flexibility OR_i and larger download flexibility DR_i. Taking this into consideration, RRH-ODF ranks the request to be scheduled by the indices RW_i, which are calculated by the expression, $\omega_i^{\mathrm{unit}} - \mathrm{OR}_i + \mathrm{DR}_i$.

3.1.3 Time slack insertion strategy

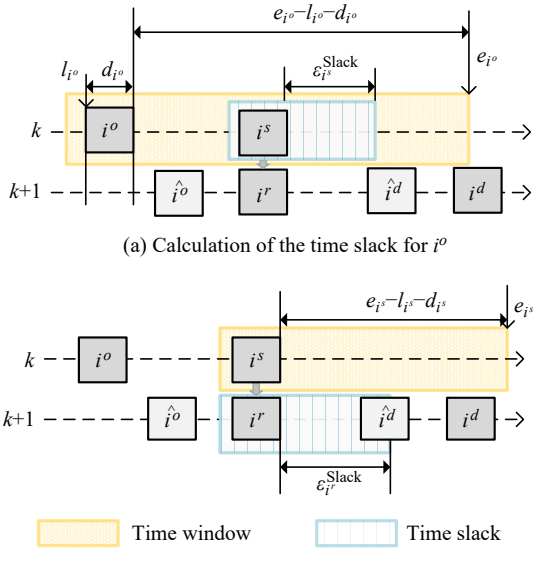
Given a timeline with a set of scheduled tasks, the conflict-avoidance insertion algorithm inserts the tasks of the unscheduled requests in that timeline without any conflicts. A scheduled task in the timeline can be shifted to the left or right in its time window to insert subsequent tasks. Based on the average response time objective G_s, we start the scheduled task as early as possible. For instance, if a task is going to be inserted between task i and task $i+1$, task i will remain unchanged, and task $i+1$ will be pushed to the right as far as the new task could be placed. This guarantees that no task has an empty space on the left side of the window, which is effective in decreasing the running time of the moving operations.

To calculate the extent to which a task can be pushed, we propose a Time slack insertion strategy for complex Time Coupling (T-TC) constraints. The T-TC is different from the time slack idea described in Refs. [10, 36], where the time slack is defined as the maximum duration a task can be postponed before the timeline becomes infeasible. However, such ideas cannot address the problem under consideration herein, where the postponing of an observation task would also affect the coupled download and transmission tasks. Therefore, T-TC is designed to handle the tight-time coupling of different types of tasks.

As shown in Fig. 3, there is a scheduled request $r_i = \{i^o, i^d, i^r, i^s\}$ and another scheduled request $r_{\hat{i}} = \{\hat{i}^o, \hat{i}^d\}$. Request r_i can be decomposed into an observation task i^o, a download task i^d, and transmission tasks i^s and i^r, where i^s represents sending the observed data and i^r means receiving the data. We use the notation $\varepsilon_i^{\mathrm{Slack}}$ to represent the time slack of task i. If the task is not a transmission task, the time slack of each task depends only on the latest start time of its next job on the timeline. In Fig. 3a, the time slack of i^o only depends on the latest start time of i^s. The time slack of i^o can be calculated as follows:

$$\varepsilon_{i^o}^{\mathrm{Slack}} = \min\left(\varepsilon_{i^s}^{\mathrm{Slack}} - \Delta s_{i^o,i^s}, e_{i^o} - l_{i^o} - d_{i^o}\right) \tag{23}$$

where $\varepsilon_{i^o}^{\mathrm{Slack}}$ and $\varepsilon_{i^s}^{\mathrm{Slack}}$ represent the time slack of i^o and i^s, d_{i^o} refers to the task execution duration of i^o, l_{i^o} is the task start time for i^o, $\Delta s_{i^o,i^s}$ is the difference in the transition times between task start at l_{i^o} and $l_{i^o} + \varepsilon_{i^o}^{\mathrm{Slack}} - d_{i^o}$, and b_{i^o} and e_{i^o} represent the start and end times for the time window of i_o, respectively.



(a) Calculation of the time slack for i^o

(b) Calculation of the time slack for i^s

Fig. 3 Three different states while calculating the time slack.

When dealing with a request transmission, the task must be sent and received simultaneously by two satellites. Therefore, the calculation of the time slack of the sending task depends not only on the next job but also on the time slack of the corresponding receiving task. For instance, from Fig. 3b, we can observe that $\varepsilon_{i^s}^{\text{Slack}}$ relies on $\varepsilon_{i^r}^{\text{Slack}}$. Therefore, $\varepsilon_{i^s}^{\text{Slack}}$ can be expressed by

$$\varepsilon_{i^s}^{\text{Slack}} = \min\left(e_{i^s} - l_{i^s} - d_{i^s},\ \varepsilon_{i^r}^{\text{Slack}}\right) \tag{24}$$

In MEOSS, the time slack for a task is determined individually based on its corresponding time window and the latest start time of the next task on the same timeline. However, in the case of MMCS, as the sending and receiving tasks from the same request must occur simultaneously, even if the tasks are on different timelines, their time slacks are interdependent. Therefore, it is crucial to integrate the different timelines when calculating the time slack of the task. It is calculated from the last task to the first one in a backpropagating manner. The implementation procedure is depicted in Algorithm 1.

Two constraints, namely, time window constraints in Formulas (7) and (11), should be checked before determining the position to insert an unscheduled task. The former constraint mandates that the start time and end time of the task should be restricted within its time window. The latter constraint requires sufficient transition time between the unscheduled task, its predecessor task and its successor task. The successor task should be postponed for a duration shorter than its corresponding time slack. If the time is not enough, this process also changes the observation mode from a high resolution to a lower resolution to reduce transmission and download times. As shown in Fig. 4, there are three different cases in the constraint check process for determining the insertion position, which is the start, middle, and end of the time window. In Fig. 4a, the start time of position 1 should not only be greater than the end time of task 1 plus the transition time between task 1 and task i (transition time constraint in Formula (11)), but also greater than the start time of the time window of i (time window constraint in Formula (7)). The positions in the middle of the time window must conform only to the transition time constraint in Formula (11), as shown in Fig. 4b. Figure 4c shows a situation opposite to that shown in Fig. 4a. The delay of task 2 in Fig. 4a, task 3 in Fig. 4b, and task $n+1$ in Fig. 4c should be less than the corresponding time

Algorithm 1 CalculateTimeSlack

Input: Scheduled task set T_k^{Sche} on satellite k ($k \in S$)

Output: Time slack $\varepsilon_i^{\text{Slack}}$ of every scheduled task i

$T^{\text{Sche}} = \bigcup_k T_k^{\text{Sche}}$;
Sort T^{Sche} in ascending order of task start time;
for i from $|T^{\text{Sche}}|$ to 1 **do**
  if i is the last task in T_k^{Sche} ($k \in S$) **then**
    if i is a transmission task **then**
      $\varepsilon_i^{\text{Slack}} = \min\left(e_i - l_i - d_i,\ \varepsilon_{i^s}^{\text{Slack}}\right)$;
    else
      $\varepsilon_i^{\text{Slack}} = e_i - l_i - d_i$;
    end
  else
    if i is a transmission task **then**
      $\varepsilon_i^{\text{Slack}} = \min\left(\varepsilon_{i^{\text{Suc}}}^{\text{Slack}} - \Delta s_{i,\,i^{\text{Suc}}},\ e_i - l_i - d_i,\ \varepsilon_{i^s}^{\text{Slack}}\right)$;
    else
      $\varepsilon_i^{\text{Slack}} = \min\left(\varepsilon_{i^{\text{Suc}}}^{\text{Slack}} - \Delta s_{i,\,i^{\text{Suc}}},\ e_i - l_i - d_i\right)$;
    end
  end
end

Note: i^{Suc} represents the successor task of task i, $\varepsilon_{i^{\text{Suc}}}^{\text{Slack}}$ represents the time slack of task i^{Suc}, and $\Delta s_{i,\,i^{\text{Suc}}}$ denotes the differential in transition time when task i starts at two distinct points, specifically, the earliest possible time l_i and the latest feasible time $l_i + \varepsilon_i^{\text{Slack}} - d_i$, leading to the successor task i^{Suc}.

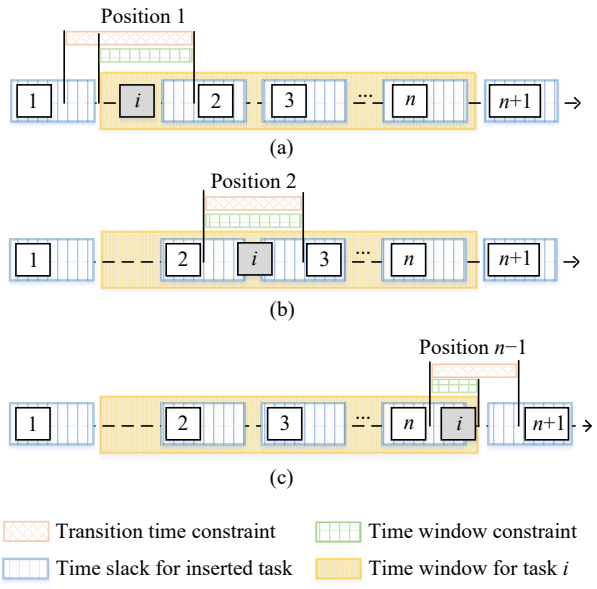


Fig. 4 Three different states encountered in finding the insertion position, (a) at the start of the time window, (b) in the middle of the time window, and (c) at the end of the window.

slack.

The above analysis shows that T-TC can obtain all potential insertion positions. The implementation procedure is depicted in Algorithm 2.

3.1.4 Position selection heuristic

After finding all potential insertion positions for the task, the insertion algorithm selects the best position and observation mode. This problem involves two main challenges. The first challenge is to select the proper insertion position to maximize the system revenue with limited accumulated operation times. The second challenge is to arrange task transmission tasks between the satellites and download them to the ground station to achieve a rapid response to the request despite the limited onboard memory storage.

Algorithm 2 GetPositionSet

Input: Scheduled task set T_k^{Sche} on satellite k ($k \in S$), unscheduled task i, and time window set $\bigcup_{k\in S} W_{ik}$ of the unscheduled task i

Output: output position tuple set P_i for the unscheduled task i to insert

$T^{\text{Sche}} = \bigcup_k T_k{}^{\text{Sche}}$;
for i from $|T^{\text{Sche}}|$ to 1 **do**
  V_{ijk}^{s} ←Find all scheduled tasks related to w_{ijk}
  for each task$v$ in V_{ijk}^{s} **do**
    for each m in M **do**
      if $m \cdot d_i^{\text{unit}} + s_{vi} + s_{i(v+1)} \leqslant l_{(v+1)} - l_v - d_v$ **then**
        if $(v = 1) \wedge (l_{v+1} + \varepsilon_{v+1}^{\text{Slack}} - b_{ijk} > m \cdot d_i^{\text{unit}} + s_{i\,(v+1)})$ **then**
          Set position start time;
          $\text{pl}_{ik} = \max(b_{ijk},\ l_v + d_v + s_{vi})$;
        else if $(v = n) \wedge (e_{ijk} - l_v - d_v > m \cdot d_i^{\text{unit}} + s_{vi})$ **then**
          Set position start time;
          $\text{pl}_{ik} = \max(e_{ijk} - m \cdot d_i^{\text{unit}},\ l_v + d_v + s_{vi})$;
        else
          Set position start time;
          $\text{pl}_{ik} = l_v + d_v + s_{vi}$ for task i;
        end
      end
    end
  end
  Set position observation type $\text{pm}_{ik} = m$ for task i, and add position tuple $p_{ik} =< \text{pl}_{ik}, \text{pm}_{ik} >$ to the position tuple set P_i
end

First challenge: The selection of the insertion position and the observation mode directly influence the system revenue and cost. This is because the higher the resolution associated with the chosen task observation mode, the greater the task revenue, while simultaneously, a higher resolution observation mode would require more task execution time and storage resources. To balance the revenue and cost, a revenue-based, conflict-avoidance observation position selection heuristic is proposed.

Second challenge: Every observation task of the request needs to choose an appropriate time to download to the ground station or transmit to another satellite to satisfy the onboard storage constraints. However, the unnecessary transmission tasks are associated with additional consumption of storage and time. Therefore, a download position selection heuristic is designed to minimize the request response time and extra resource waste on the transmission task.

Overall, the position selection heuristic is divided into two parts based on various task types (i.e., the observation task and the download task). Hence, the heuristic is named the Position Selection heuristic for Observation and Download tasks (PS-OD).

(1) Observation task

For each potential position, we can evaluate the download difficulty by calculating the overlap between the position and the download time window after obtaining the observation mode, the task start time, the observation satellite, and the task time slack. Let the time window for each potential position $p_{ik} \in P_i$ to be $w_{p_{ik}} = [\text{pl}_{ik},\ \text{pl}_{ik} + \text{pm}_{ik} \cdot d_i^{\text{unit}} + \overline{s}]$. Two overlap time windows, $W_{p_{ik}}^{\text{overlap}}$ and $\overline{W}_{p_{ik}}^{\text{overlap}}$, as shown in Fig. 5, are

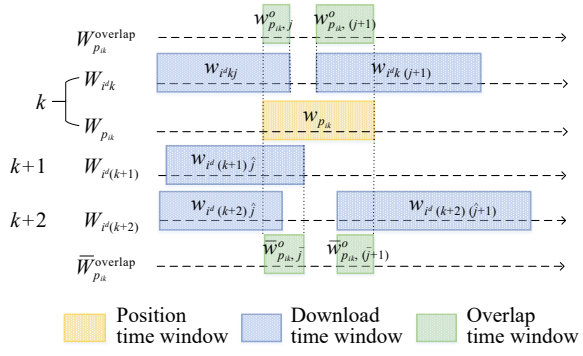


Fig. 5 Diagram of the overlapping window sets $W_{p_{ik}}^{\text{overlap}}$ and $\overline{W}_{p_{ik}}^{\text{overlap}}$, where $W_{i^d k}$, $W_{i^d k+1}$, and $W_{i^d k+2}$ are the download time window sets for observation task i of satellite k, $k+1$, and $k+2$, respectively.

defined to express the position time window $w_{p_{ik}}$ overlapped with download time window of satellite k and its download time window $\bar{k}$ ($\bar{k} \neq k$).

After obtaining $W_{p_{ik}}^{\text{overlap}}$ and $\overline{W}_{p_{ik}}^{\text{overlap}}$, we denote the index PW_{ik} to select the observation position as follows:

$$\text{PW}_{p_{ik}} = \text{pm}_{ik} \cdot \left(\chi_1 \cdot \text{size}\left(W_{p_{ik}}^{\text{overlap}}, \bigcup_{p_{ik}} W_{p_{ik}}^{\text{overlap}} \right) + \chi_2 \cdot \text{size}\left(\overline{W}_{p_{ik}}^{\text{overlap}}, \bigcup_{p_{ik}} \overline{W}_{p_{ik}}^{\text{overlap}} \right) \right), \ \forall p_{ik} \in P_i \quad (25)$$

where χ_1 and χ_2 are the constant parameters for the index PW_{ik} subject to $\chi_1 >> \chi_2$, and pm_{ik} refers to the observation mode of the position.

(2) Download task

Before selecting the download position of a request, the observation mode pm_i of the request is determined by the observation position selection algorithm. Constraint in Eq. (13) requires that the real task duration for downloading this request is also determined. Therefore, we can only select the position of the download task depending on the task start time $\text{pl}_{i\bar{k}}$ and task executing satellite $\bar{k}$. Intuitively, choosing the download position with an earlier task start time could obtain a better outcome in terms of task response time and system revenue. The task response time refers to the observation task start time minus the download task end time, which is directly influenced by the task start time pl_{ik}. In addition, an earlier download task start time can save onboard storage space for other unscheduled requests. If there is more than one position that starts simultaneously, we tend to choose the position of satellite k that observes the task, considering that transmitting the task to other satellites would waste time and resources. Therefore, the index $\text{PW}'_{i\bar{k}}$ are defined to select the download position as follows:

$$\text{PW}'_{i\bar{k}} = \left(1 - \frac{\text{pl}_{i\bar{k}}}{\max\limits_{i,\bar{k}} (\text{pl}_{i\bar{k}})} \right) \cdot \chi_1 + I(k = \bar{k}) \cdot \chi_2 \quad (26)$$

where an indicator function $I(\cdot)$ is used to signify certain conditions. Particularly, when the satellite for downloading request is the same as the satellite observes it (i.e., $k = \bar{k}$), $I(k = \bar{k})$ equals 1. Otherwise, $I(k = \bar{k})$ equals 0.

3.1.5 Conflict-avoidance insertion algorithm framework

Figure 6 shows the complete process of the conflict-avoidance insertion algorithm. After initializing the request list, the algorithm first ranks the requests with

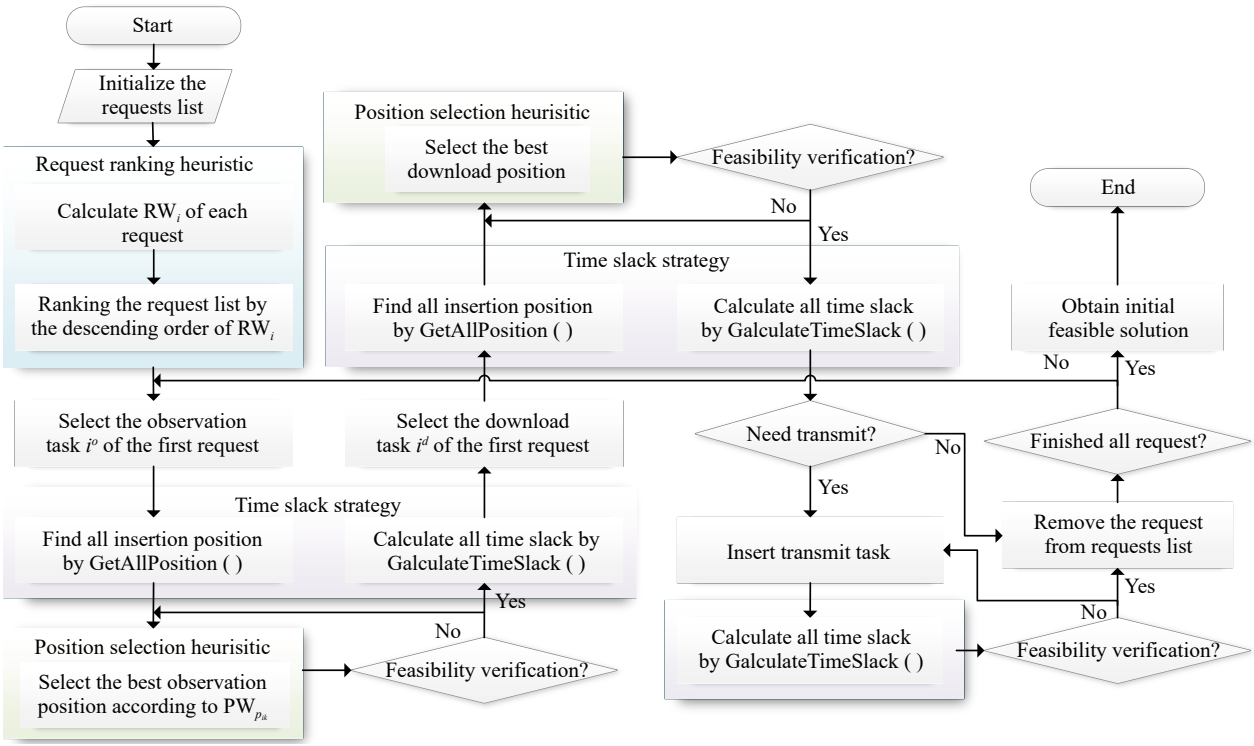


Fig. 6 Diagram of the conflict-avoidance insertion algorithm.

RW_i in RRH-ODF and schedules them one by one. When a request i is selected, observation task i^o, download task i^d and transmission task are inserted into the solution in turn. Four steps are involved in inserting an individual task, regardless of its type: (1) finding all positions using the function GetAllPosition ( ); (2) evaluating the best position; (3) verifying its feasibility and inserting the task in the best position if feasible; and (4) calculating the time slack of all scheduled tasks by the function CalculateTimeSlack ( ). Algorithm 2 is terminated when all requests in the ranked request list are inserted.

3.2 Improvement algorithm based on TS

TS is a well-known local search heuristic first proposed by Glover[37], and has been applied in various areas[38]. In TS, the recently visited solutions or the recently used local search moves are stored in a tabu list to prevent trapping in the local optima. The forbidden solutions or movements lose their tabu states after a certain duration. The main adaptation of TS to the problem proposed in this study is that it is more accurate for domain-specified solution construction and repair procedures for solution improvement. This can avoid the extra effort of resolving conflicts among the tasks from a single request.

3.2.1 Process of TS algorithm

The TS algorithm (Algorithm 3) contains several major steps. First, all requests that fail to be inserted with the maximal observation mode in the current solution are added to a queue. Further, the first request in the queue is split into the observation, download, and transmission tasks. For every task i in the selected request, Algorithm 3 cuts its time window and finds the set of all possible insertion positions P_i. If the position set P_i does not meet the jump condition (see Section 3.2.3), insertion will be attempted in every position in the position set P_i. Otherwise, Algorithm 3 will calculate all potential insertion positions for the removal of each scheduled task in the sequence. The potential insertion position sets for the removal of each scheduled task are denoted as $\tilde{P}_i$. When task i is inserted, it will be added to the removal tabu task list. If task i fails to be inserted after attempting all potential positions in the set P_i and $\tilde{P}_i$, the algorithm will remove all tasks of the request i and add it to the insertion tabu task list. Further, the time slack of every scheduled task will be recalculated because of the change in the scheduled tasks in the current solution

Algorithm 3 Improved algorithm based on TS

Input: Failed scheduled request set FTL; initial solution S^I
Output: Final solution S^*

$S = S^I$ and sort FTL by RRH-ODF;
while not meet terminal criteria **do**
  Select the first r_i from FTL;
  for task i in request r_i **do**
    Calculate P_i by GetPositionSet (S, i, W_i);
    if P_i not meet the jump conditions **then**
      for each $p_{ik} \in P_i$ **do**
        $S' \leftarrow$ Insert p_{ik} into solution S;
        if S' meets Formulas (12) and (14) **then**
          Add i into removing tabu list and break;
        end
      end
    else
      Calculate $\tilde{P}_i$ by GetPositionSet (S, i, W_i) in the case of removing another scheduled task;
      for each $\tilde{p}_{ik} \in \tilde{P}_i$ **do**
        $S' \leftarrow$ Insert $\tilde{p}_{ik}$ into solution S;
        if S' meets Formulas (12) and (14) **then**
          Add i into removing tabu list and break;
        end
      end
    end
    if task i fail to insert **then**
      Add task i to the inserting tabu list;
    end
    CalculateTimeSlack (S');
  end
  if $f(S') > f(S^*)$ **then**
    $S^* \leftarrow S'$;
  end
end

(i.e., insertion or removal of a task). Algorithm 3 repeats these steps until the predefined stopping criteria are satisfied (Section 3.2.3).

3.2.2 Tabu list setting

As mentioned above, we use two tabu lists, namely, the removal tabu task list and the insertion tabu task list. A task is added to the removal tabu task list when it is inserted into the solution. When searching a position for an unscheduled task while removing scheduled tasks, the scheduled tasks in the removal tabu task list are forbidden to be removed. A task will be added to the insertion tabu task list when the task (or the other types of task in the same request) is not inserted into the solution. The tasks in the insertion tabu task list are

not allowed to be inserted into the solution. Every task in the removal (or insertion) tabu task list will be specified as a tabu tenure (i.e., a number of iterations). This means that the task cannot be removed from (or inserted into) the solution during the specific iterations.

3.2.3 Jump conditions and terminal conditions

Jump conditions are those situations in which the algorithm skips looking for all potential insert positions without deleting the task for the selected unscheduled task, and instead looks for all potential insert positions without deleting the already scheduled tasks. There are three terminal criteria:

(1) All tasks are scheduled and receive full revenue.

(2) The number of iterations reaches the maximum allowed value.

(3) The objective value of the current solution does not improve after the maximum number of consecutive iterations.

4 Experimental Simulation

4.1 Experimental design

The experiments are run on a laptop with an Intel Core i7-10710U CPU @ 1.10 GHz and 16.0 GB RAM.

Herein, the satellite system deployment involves one ground station and five isomorphic mid-orbit satellites (see Fig. 7). These satellites are evenly distributed on the equatorial plane and are connected by real-time intersatellite links. These links enable them to mutually transmit observation data. In Table 1, the basic parameters of the satellite system are presented. Other parameters of the angle in Eq. (1) are as follows: $a_1 = 1.5$, $a_2 = 2$, $a_3 = 2.5$, and $a_4 = 3$. The request set is generated following the configuration from Liu et al.[15], which is randomly and uniformly distributed over two geographic regions: the area (i.e., China) and the whole world. For the whole request set and area request set, seven instances are designed. The number of requests changes from 50 to 400 with an increment of 50. The related request parameters in the experiment are shown in Table 2.

For each request, four observation modes can be selected, which are represented as values ranging from 1 to 4. The revenue, processing time, and data storage of the request are positively and linearly dependent on the selected observation mode. In particular, the real data storage and the processing time during scheduling are determined by multiplying the value of the selected observation mode and the respective unit value in Table 2.

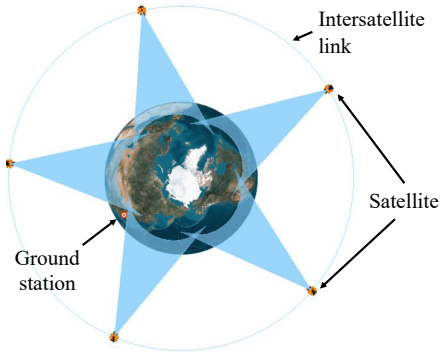


Fig. 7 Satellite system deployment diagram.

Table 1 Basic parameters of satellite system.

Parameter	Value	Error
Orbital period P	21 423 s	3 s
Satellite maximum rolling angle	30°	0.01°
Satellite maximum pitch angle	15°	0.01°
Satellite maximum yaw angle	15°	0.01°
Satellite data storage capacity C_k	120 unit	0.01 unit
Satellite max operation time E_k	1800 s	3 s

Table 2 Request parameters.

Parameter	Value
Request unit data storage v_i^{unit}	Random in (15, 25) unit
Request unit processing time d_i^{unit}	Random in (10, 25) s
Request observation mode	Random in (1, 4)
Request revenue ω_i^{unit}	Random in (1, 10)
Coefficient of task duration δ	0.2
Coefficient of task consumed storage ε	0.7

Particularly, if task i is observed under mode m, its task revenue ω_i, duration d_i, and consumed storage v_i are determined as follows: $\omega_i = m \cdot \omega_i^{\text{unit}}$, $d_i = m \cdot \delta \cdot d_i^{\text{unit}}$, and $v_i = m \cdot \varepsilon \cdot v_i^{\text{unit}}$, where δ and ε are coefficients of task duration and consumed storage, respectively. They directly affect how the observation mode selection impacts those task features. Herein, the coefficient values are based on our survey at the China Centre for Resources Satellite Data and Application. The task duration coefficient δ is set to 0.2, and the task consumed storage coefficient ε is set to 0.7 (see Table 2). Note that the correlation coefficients are merely the input parameters for the simulation experiment scenario, and their values have little impact on algorithm performance in practical applications. Therefore, the value of the coefficients can be adjusted according to the real-world application scenario to

facilitate the broad applicability of the algorithm.

Each request contains an observation task, a download task, and a set of transmission tasks. The request is fulfilled only when it is observed by a satellite and downloaded to a ground station; therefore, it contains at least one observation task and one download task. When an observation task is not downloaded by the satellite observing it, a set of transmission tasks occurs over the intersatellite links. For the same request, the observation task, the download task, and the transmission task share the same parameters as those in Table 2 (e.g., task unit data storage and task unit action duration).

4.2 Experiment on conflict-avoidance insertion algorithm

The conflict-avoidance insertion algorithm contains several heuristics, including the time slack strategy related to transmission and download, the ranking heuristic for requests, and the selection heuristic for different positions. To investigate the effectiveness of each heuristic, we replace it with other baselines while keeping the other heuristics unchanged. Two assessment criteria are employed to compare these heuristics, namely, the average response time and the CPU running time.

4.2.1 Performance of request ranking heuristic

We compare the performance of the ranking heuristic for requests with other heuristics, such as the Revenue-based First (RF) heuristic and the modified Weighted Shortest Imaging Time First (m-WSITF) heuristic[39]. The RF heuristic simply prioritizes the requests with the highest revenue for scheduling, and the m-WSITF heuristic preferentially schedules the request with the highest $\omega_i/(d_i+\overline{\sigma})$ ratio, where ω_i represents the request revenue, d_i represents the request duration, and $\overline{\sigma}$ is the average transition time for every available insertion position in the current solution.

Figure 8 shows the performance of different request ranking heuristics, where the proposed RRH-ODF performs better than other heuristics both in response time and CPU time. RRH-ODF has a significant advantage in terms of response time, especially for the whole request list. The response time of the three heuristics drops as the number of requests increases up to 250, but increases with a further increase in the number of requests for the whole request list. Because the total response time is relatively small in the initial stage, the average response time can be large due to the small number of requests. In addition, the three heuristics have a stable response time for the area request list, since the requests in that list are more crowded and less influenced by changes in the number of requests. Note that the difference in the values CPU times between RRH-ODF and the other heuristics grows with an increase in the number of requests.

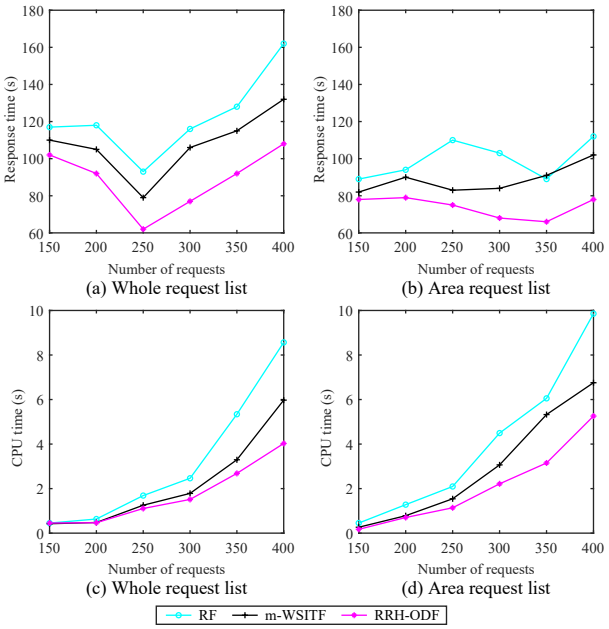


Fig. 8 Performance of RF heuristic and m-WSIPT heuristic compared with the proposed RRH-ODF.

4.2.2 Performance of time slack strategy

Herein, we compare the proposed Time slack insert strategy for T-TC with the No-Time Slack (NTS) strategy and the Backward/forward Time Slack (BTS) strategy[15]. The NTS strategy inserts the task without moving any inserted task, and the BTS strategy only considers moving the two tasks adjacent to the insertion position when inserting a new task.

In Fig. 9, T-TC has a significant advantage in terms of the request response time compared with the baselines, particularly for the area request list. When downloading a request, T-TC can obtain an earlier task insertion position by moving the scheduled task to reduce the request response time. In addition, T-TC has a slight advantage in terms of CPU time, which increases with an increase in the number of requests.

4.2.3 Performance of position selection heuristics

The best PS-OD heuristic contains two heuristics: the observation position selection and the download position selection. We study them separately. The former is compared with the Priority-Based and

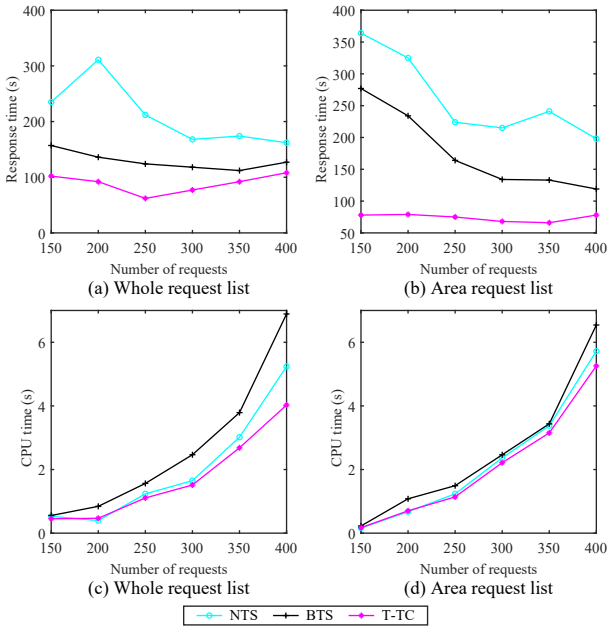


Fig. 9 Performance of NTS strategy and BTS strategy compared to the proposed T-TC strategy for TTC.

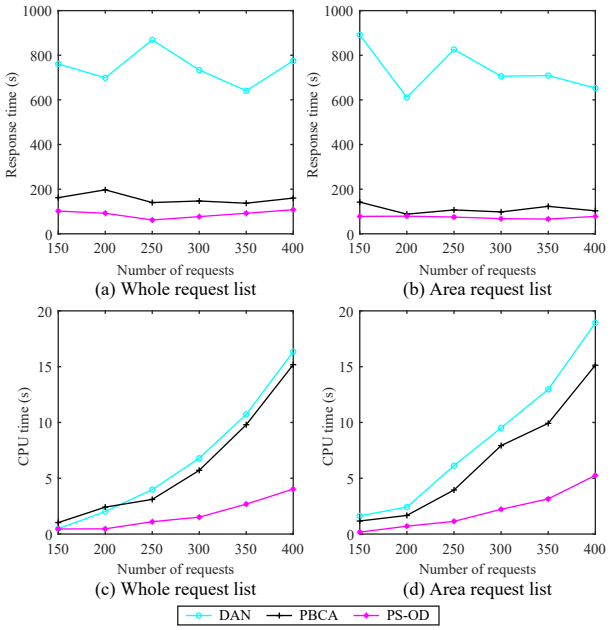


Fig. 10 Performance of DAN heuristic and PBCA heuristic compared to PS-OD.

Conflict-Avoidance (PBCA) heuristic[40], and the latter is compared with the Download-As-Needed heuristic (DAN)[41]. The PBCA heuristic prioritizes the positions that have less overlap with other observation tasks, whereas the DAN heuristic schedules a download task. The PBCA heuristic prioritizes the positions that have less overlap with other observation tasks. The DAN heuristic schedules a download task when the storage space is insufficient, and selects the earliest download position on the satellite that observed the corresponding request. As PBCA and DAN only consider observation and download tasks, respectively, they are combined with our heuristics (i.e., PBCA is combined with the download position selection heuristic of PS-OD, and DAN is combined with the observation position selection heuristic of PS-OD) to solve the problem under consideration.

As shown in Fig. 10, the disparity in performance between DAN, PBCA, and our PS-OD is notable in terms of the response time. The results reveal that download position selection has a more significant impact on the response time than observation position selection. This is because PS-OD allows the request to be transmitted through the satellites, enabling the observed data to be transmitted to the satellite in an as early as possible download time window to minimize the average request response time. In addition, PS-OD has a lower CPU time and growth rate of the CPU time as compared to the baselines, which indicates the good performance of PS-OD for solving large-scale request instances.

4.2.4 Analysis of performance gaps

Herein, we investigate the average performance gaps between the proposed conflict-avoidance insertion algorithm and other baselines, including the time slack strategy, request ranking heuristic, and position selection heuristic. To this end, the performance gap between the proposed heuristic and any baseline heuristic A with n instances is evaluated in the following:

$$\text{gap}^{A,n} = \frac{\text{Result}^{\text{proposed},n} - \text{Result}^{A,n}}{\text{Result}^{A,n}} \tag{27}$$

Therefore, we can obtain the average gap for a different type of request distributed set (i.e., area and whole world-distributed) with N instances in total,

$$\text{Gap} = \frac{1}{N}\sum_{i}\text{gap}^{A,i} \tag{28}$$

The performance gaps are shown in Table 3. The proposed heuristics are significantly superior to all other heuristics considered in this study. Generally, the proposed heuristics work better when the request set is area-distributed, proving their help to handle the complex time conflict among requests. As compared to the NTS and BTS, the proposed T-TC outperforms significantly in the request response time (the

performance gap is at least 50%) while slightly improving the efficiency (CPU time) by 8% on average. In contrast, the proposed request ranking heuristic (i.e., RRH-ODF) improves the CPU time more than the response time. In addition, the download position selection heuristic works dramatically better both in the CPU time and response time than the DAN heuristic (the performance gaps are 87.92% and 89.68%, respectively) for the area-distributed request set. As compared to the PBCA heuristic, the observation position selection heuristic has a higher improvement in the CPU time (more than 58%) than in the response time (below 44%).

4.3 Experiment on full algorithm

Herein, we compare the performance of the proposed FTS-C heuristic with other state-of-the-art algorithms, namely, ALNS[15], Adaptive Large Neighborhood Search with Tabu Search (ALNS-TS)[42], TS[43], and Effective Tabu Search (ETS)[16]. IBM ILOG CPLEX version 12.8 is used for Mixed Integer Programming (MIP) model solving. A time limit of 3600 s is set for CPLEX solving, which needs over 100× as much running time as FTS-C does. The results for each algorithm are presented as an average of 20 runs. The value of the tabu tenure is set as 20 and 25 for insertion and deletion, respectively.

First, we compare the proposed algorithm with the CPLEX solver in terms of the solution quality and running time. The solution quality includes the total revenue from completing requests and the average response time from observing the request to download the request. As shown in Table 4, CPLEX can only find the optimal solution for small-scale instances. Its performance is unsatisfactory when the instance size increases. FTS-C can find the same optimal solution in terms of the revenue and the near-optimal solution in terms of the response time when the number of requests is below 50. The running times for FTS-C are on the order of 10^{-1} s, which are several orders of magnitude smaller than those for CPLEX when the number of requests is large.

Then, we compare the proposed algorithm FTS-C with other algorithms in terms of the revenue gap. The revenue gap between any baseline Algorithm A and the proposed FTS-C is calculated by

$$\text{gap}_A = \frac{G_r^{\text{FTS-C}} - G_r^A}{G_r^{\text{FTS-C}}} \tag{29}$$

The results are exhibited in Fig. 11. The revenue gaps between the other algorithms and our proposed FTS-C are greater than 0, which indicates that the revenue obtained by FTS-C is consistently higher than that obtained by the other algorithms. In addition, Fig. 11 shows that the revenue gap for the area request list is higher than that for the whole request list on average, proving that the proposed FTS-C is evidently superior for the area request list.

A comparison of the response time G_s for the five algorithms in this study with 100–400 requests is shown in Fig. 12. The values of response time have been charted on a log scale $\ln(\,)$ for better visual

Table 3 Performance gaps of baselines compared to the proposed insertion heuristic for different instances.

(%)

Criteria	Type of request set	Time slack strategy		Request ranking heuristic		Position selection heuristic	
		NTS	BTS	RF	m-WSIPT	DAN	PBCA
CPU time	Whole	1.57	15.74	33.61	17.96	63.52	58.30
	Area	2.65	16.02	49.68	27.06	77.75	65.17
Response time	Whole	64.70	50.02	27.21	17.79	87.92	43.17
	Area	72.21	68.96	25.05	16.13	89.68	30.80

Table 4 Performance of the proposed algorithm compared to CPLEX for different instances.

Number of requests	CPLEX			FTS-C		
	Total revenue from completing requests	Response time (s)	CPU time (ms)	Total revenue from completing requests	Response time (s)	CPU time (ms)
6	164	32.4	240	164	32.4	42
12	196	42.5	1734	196	42.5	164
25	460	36.7	15 671	460	36.7	257
50	1104	41.8	100 869	1104	41.8	323
100	2046	174.3	3 600 000	2184	78.3	286

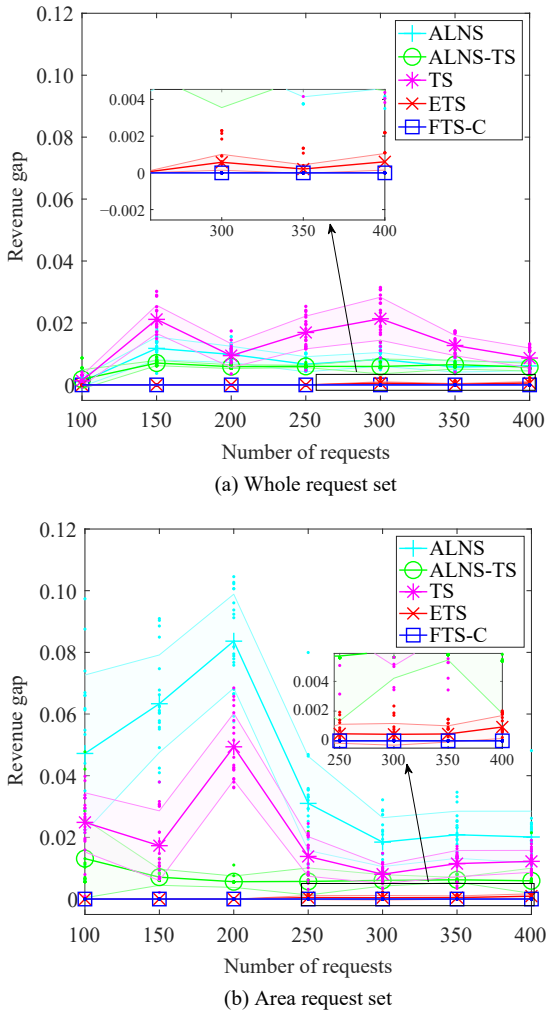


Fig. 11 Performance of revenue gap of different algorithms.

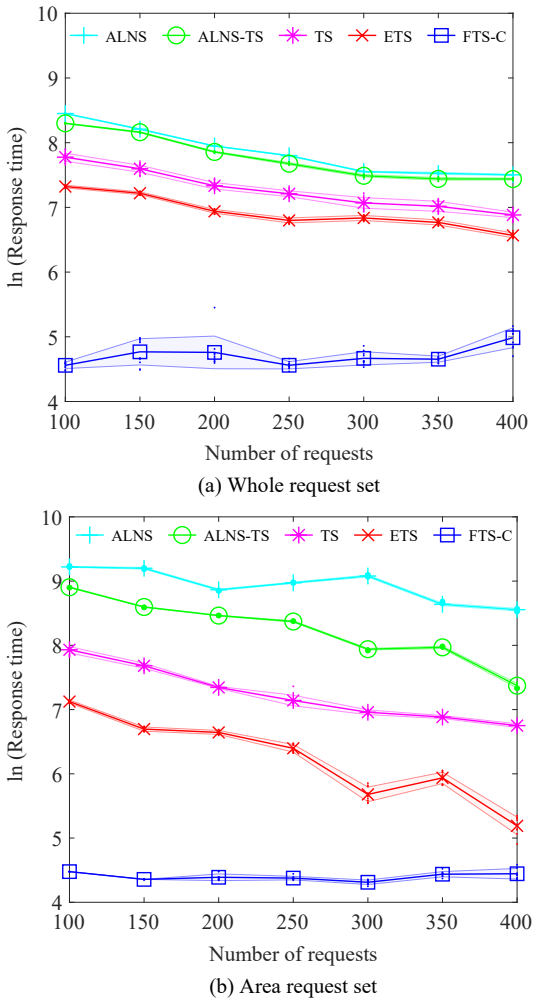


Fig. 12 Performance of response time of different algorithms.

comparison. As shown in Fig. 12, the advantage of FTS-C is significant as compared to the baselines. The response time of FTS-C is stable and remains at a low level with different numbers of requests. For the area request list, the gaps among different algorithms are bigger, especially the gap between FTS-C and ALNS.

Figure 13 shows the variation of CPU time in millisecond for different algorithms with an increasing number of requests. The difference of the values of CPU time for the five algorithms is negligible for small-scale requests. However, as the number of requests increases, the differences among them become more pronounced. While the values of CPU time for ALNS, TS, and ALNS-TS significantly increase with an increasing number of requests, those for FTS-C and ETS increase slowly. Specifically, as shown in Fig. 13b , when the number of requests is 400, the values of CPU time for ALNS, TS, and ALNS-TS have an order of magnitude of 10^5, whereas that of FTS-C has an order of magnitude of only 10^3, demonstrating that FTS-C is the most efficient among the five algorithms.

5 Conclusion

This study investigates a novel problem, namely, MMCS problem, which is different from the traditional MEOSS problem in that MMCS schedules tasks are in different observation modes, and the imaging, transmission, and download operations are integrated into a unified model while accounting for the time-dependent and coupled relationships among them. We formulate MMCS as a mixed integer programming model, which takes into account the observation, data acquisition, transmission, and download tasks. In addition to the traditional objective of the AEOS scheduling problem, maximizing task revenue, a new

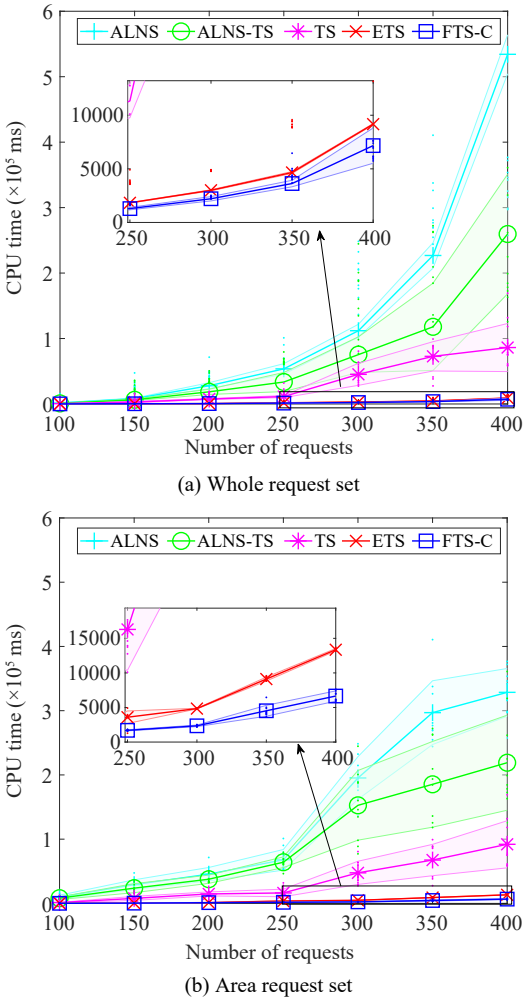


Fig. 13 Performance of CPU time of different algorithms.

and important objective of minimizing the average request response time is added to reflect real-world requirements.

To solve MMCS, FTS-C heuristic is proposed to optimize the request revenue and the average request response time. FTS-C can be split into two parts: a conflict-avoidance insertion algorithm and an improvement algorithm based on TS. In the experimental simulations, FTS-C is tested on different problem instances reflecting real-world situations. Experiments on conflict-avoidance insertion algorithm show that (1) all heuristics discussed in this study perform better on the area-distributed instance, indicating that the algorithm helps to handle high time-conflict requests. (2) The proposed time slack strategy works remarkably well in terms of the solution quality with a slight improvement in efficiency. (3) The proposed request ranking heuristic improves the solution efficiency more than the solution quality. (4) The download position selection heuristic dramatically improves the solution quality and efficiency. Further, an extensive empirical study based on a real-world situation demonstrates that FTS-C can produce a higher quality solution in less time than other state-of-the-art algorithms and the CPLEX solver. The strategies designed to schedule tasks with tight time and storage constraints can be instructive for addressing other similar scheduling problems.

In the future, we plan to enhance the insertion algorithm using machine learning. We believe that the combination of online machine learning and optimization is a promising approach toward this goal. In addition, it would be a challenging and interesting topic to consider the uncertainty of data transmission through satellites and design an algorithm that schedules requests in a stochastic environment with different levels of uncertainty.

Acknowledgment

This research was supported by the National Natural Science Foundation of China (No. 72001212) and the Hunan Provincial Innovation Foundation for Postgraduate (No. CX20200022).

References

[1] J. Wang, E. Demeulemeester, and D. Qiu, A pure proactive scheduling algorithm for multiple earth observation satellites under uncertainties of clouds, *Comput. Oper. Res.*, vol. 74, pp. 1–13, 2016.

[2] F. Yao, J. Li, Y. Chen, X. Chu, and B. Zhao, Task allocation strategies for cooperative task planning of multi-autonomous satellite constellation, *Adv. Space Res.*, vol. 63, no. 2, pp. 1073–1084, 2019.

[3] H. Chen, Z. Zhong, J. Wu, and N. Jing, Multi-satellite data downlink resource scheduling algorithm for incremental observation tasks based on evolutionary computation, in *Proc. 2015 Seventh Int. Conf. on Advanced Computational Intelligence*, Wuyi, China, 2015, pp. 251–256.

[4] J. Wu, J. Xiong, H. Dai, Y. Wang, and C. Xu, MIX-RS: A multi-indexing system based on HDFS for remote sensing datastorage, *Tsinghua Science and Technology*, vol. 27, no. 6, pp. 881–893, 2022.

[5] R. Xu, H. Chen, X. Liang, and H. Wang, Priority-based constructive algorithms for scheduling agile earth observation satellites with total priority maximization, *Expert Syst. Appl.*, vol. 51, pp. 195–206, 2016.

[6] L. Barbulescu, J. P. Watson, L. D. Whitley, and A. E. Howe, Scheduling space-ground communications for the air force satellite control network, *J. Schedul.*, vol. 7, no. 1, pp. 7–34, 2004.

[7] J. Wang, E. Demeulemeester, X. Hu, D. Qiu, and J. Liu, Exact and heuristic scheduling algorithms for multiple

earth observation satellites under uncertainties of clouds, *IEEE Syst. J.*, vol. 13, no. 3, pp. 3556–3567, 2019.

[8] M. Vasquez and J. K. Hao, Upper bounds for the spot 5 daily photograph scheduling problem, *J. Comb. Optim.*, vol. 7, no. 1, pp. 87–103, 2003.

[9] X. Chen, G. Reinelt, G. Dai, and A. Spitz, A mixed integer linear programming model for multi-satellite scheduling, *Eur. J. Oper. Res.*, vol. 275, no. 2, pp. 694–707, 2019.

[10] L. He, M. de Weerdt, and N. Yorke-Smith, Time/sequence-dependent scheduling: The design and evaluation of a general purpose tabu-based adaptive large neighbourhood search algorithm, *J. Intell. Manuf.*, vol. 31, no. 4, pp. 1051–1078, 2020.

[11] G. Peng, R. Dewil, C. Verbeeck, A. Gunawan, L. Xing, and P. Vansteenwegen, Agile earth observation satellite scheduling: An orienteering problem with time-dependent profits and travel times, *Comput. Oper. Res.*, vol. 111, pp. 84–98, 2019.

[12] X. Niu, H. Tang, L. Wu, R. Deng, and X. Zhai, Imaging-duration embedded dynamic scheduling of earth observation satellites for emergent events, *Math. Probl. Eng.*, vol. 2015, pp. 731–734, 2015.

[13] J. Berger, N. Lo, and M. Barkaoui, QUEST–a new quadratic decision model for the multi-satellite scheduling problem, *Comput. Oper. Res.*, vol. 115, p. 104822, 2020.

[14] Z. E, R. Shi, L. Gan, H. Baoyin, and J. Li, Multi-satellites imaging scheduling using individual reconfiguration based integer coding genetic algorithm, *Acta Astronaut.*, vol. 178, pp. 645–657, 2021.

[15] X. Liu, G. Laporte, Y. Chen, and R. He, An adaptive large neighborhood search metaheuristic for agile satellite scheduling with time-dependent transition time, *Comput. Oper. Res.*, vol. 86, pp. 41–53, 2017.

[16] A. Sarkheyli, A. Bagheri, B. Ghorbani-Vaghei, and R. Askari-Moghadam, Using an effective tabu search in interactive resources scheduling problem for LEO satellites missions, *Aerosp. Sci. Technol.*, vol. 29, no. 1, pp. 287–295, 2013.

[17] X. Wang, H. Zhang, S. Bai, and Y. Yue, Design of agile satellite constellation based on hybrid-resampling particle swarm optimization method, *Acta Astronaut.*, vol. 178, pp. 595–605, 2021.

[18] F. Marinelli, S. Nocella, F. Rossi, and S. Smriglio, A Lagrangian heuristic for satellite range scheduling with resource constraints, *Comput. Oper. Res.*, vol. 38, no. 11, pp. 1572–1583, 2011.

[19] C. Li and B. Xu, Optimal scheduling of multiple sun-synchronous orbit satellites refueling, *Adv. Space Res.*, vol. 66, no. 2, pp. 345–358, 2020.

[20] D. Madakat, J. Morio, and D. Vanderpooten, A biobjective branch and bound procedure for planning spatial missions, *Aerosp. Sci. Technol.*, vol. 73, pp. 269–277, 2018.

[21] E. Pellegrini and R. P. Russell, A multiple-shooting differential dynamic programming algorithm, Part 2: Applications, *Acta Astronaut.*, vol. 173, pp. 460–472, 2020.

[22] J. Wang, G. Song, Z. Liang, E. Demeulemeester, X. Hu, and J. Liu, Unrelated parallel machine scheduling with multiple time windows: An application to earth observation satellite scheduling, *Comput. Oper. Res.*, vol. 149, p. 106010, 2023.

[23] H. Chen, Z. Lou, S. Peng, J. Wu, and J. Li, HiPGen: An approach for fast generation of multi-satellite observation plans via a hierarchical multi-channel transformer network, *Adv. Space Res.*, vol. 69, no. 8, pp. 3103–3116, 2022.

[24] G. Wu, J. Liu, M. Ma, and D. Qiu, A two-phase scheduling method with the consideration of task clustering for earth observing satellites, *Comput. Oper. Res.*, vol. 40, no. 7, pp. 1884–1894, 2013.

[25] J. Wang, X. Zhu, L. T. Yang, J. Zhu, and M. Ma, Towards dynamic real-time scheduling for multiple earth observation satellites, *J. Comput. Syst. Sci.*, vol. 81, no. 1, pp. 110–124, 2015.

[26] X. Hu, W. Zhu, B. An, P. Jin, and W. Xia, A branch and price algorithm for EOS constellation imaging and downloading integrated scheduling problem, *Comput. Oper. Res.*, vol. 104, pp. 74–89, 2019.

[27] Y. Xiao, S. Zhang, P. Yang, M. You, and J. Huang, A two-stage flow-shop scheme for the multi-satellite observation and data-downlink scheduling problem considering weather uncertainties, *Reliab. Eng. Syst. Saf.*, vol. 188, pp. 263–275, 2019.

[28] J. Zhang and L. Xing, An improved genetic algorithm for the integrated satellite imaging and data transmission scheduling problem, *Comput. Oper. Res.*, vol. 139, p. 105626, 2022.

[29] H. Chen, J. Wu, W. Shi, J. Li, and Z. Zhong, Coordinate scheduling approach for EDS observation tasks and data transmission jobs, *J. Syst. Eng. Electron.*, vol. 27, no. 4, pp. 822–835, 2016.

[30] W. Zhu, X. Hu, W. Xia, and P. Jin, A two-phase genetic annealing method for integrated earth observation satellite scheduling problems, *Soft Comput.*, vol. 23, no. 1, pp. 181–196, 2019.

[31] A. K. Kennedy, Planning and scheduling for earth-observing small satellite constellations, PhD dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, 2018.

[32] T. Benoist and B. Rottembourg, Upper bounds for revenue maximization in a satellite scheduling problem, *Q. J. Belg., French Italian Oper. Res. Soc.*, vol. 2, no. 3, pp. 235–249, 2004.

[33] L. He, X. Liu, G. Laporte, Y. Chen, and Y. Chen, An improved adaptive large neighborhood search algorithm for multiple agile satellites scheduling, *Comput. Oper. Res.*, vol. 100, pp. 12–25, 2018.

[34] H. Zhou, H. Qin, Z. Zhang, and J. Li, Two-echelon vehicle routing problem with time windows and simultaneous pickup and delivery, *Soft Comput.*, vol. 26, no. 7, pp. 3345–3360, 2022.

[35] M. J. Pinto, A. I. Barros, R. Noomen, P. H. A. J. M. van Gelder, and T. L. Tessensohn, A new model proposal for integrated satellite constellation scheduling within a planning horizon given operational constraints, in *Proc. 7th Int. Conf. on Operations Research and Enterprise Systems*, Funchal, Portugal, 2018, pp. 312–319.

[36] C. Verbeeck, P. Vansteenwegen, and E. H. Aghezzaf, The time-dependent orienteering problem with time windows: A fast ant colony system, *Ann. Oper. Res.*, vol. 254, no. 1, pp. 481–505, 2017.
[37] F. Glover, *Tabu Search*. Boulder, CO, USA: University of Colorado, 1988.
[38] C. Yan, J. Ma, H. Luo, and J. Wang, A hybrid algorithm based on binary chemical reaction optimization and tabu search for feature selection of high-dimensional biomedical data, *Tsinghua Science and Technology*, vol. 23, no. 6, pp. 733–743, 2018.
[39] G. Li, L. Xing, and Y. Chen, A hybrid online scheduling mechanism with revision and progressive techniques for autonomous earth observation satellite, *Acta Astronaut.*, vol. 140, pp. 308–321, 2017.
[40] X. Chen, G. Reinelt, G. Dai, and M. Wang, Priority-based and conflict-avoidance heuristics for multi-satellite scheduling, *Appl. Soft Comput.*, vol. 69, pp. 177–191, 2018.
[41] P. Wang, G. Reinelt, P. Gao, and Y. Tan, a heuristic and a decision support system to solve the scheduling problem of an earth observing satellite constellation, *Comput. Ind. Eng.*, vol. 61, no. 2, pp. 322–335, 2011.
[42] L. He, A. Guijt, M. de Weerdt, L. Xing, and N. Yorke-Smith, Order acceptance and scheduling with sequence-dependent setup times: A new memetic algorithm and benchmark of the state of the art, *Comput. Ind. Eng.*, vol. 138, p. 106102, 2019.
[43] A. Sarkheyli, B. G. Vaghei, and A. Bagheri, New tabu search heuristic in scheduling earth observation satellites, in *Proc. 2nd Int. Conf. on Software Technology and Engineering*, San Juan, PR, USA, 2010, pp. V2-199–V2-203.

Weiyi Yang received the BEng degree in simulation engineering from National University of Defense Technology (NUDT), China in 2019. She is currently a PhD candidate at College of Systems Engineering, NUDT, China. Her current research interests include artificial intelligence and metaheuristics, game theory approaches, and satellite scheduling problems.

Lei He received the BEng degree in management engineering from NUDT, China in 2014, and the PhD degree in management science and engineering from NUDT, China in 2019. From 2017 to 2019, he was a visiting PhD student at Delft University of Technology, the Netherlands. He is currently an associate professor at NUDT, China. His current research interests include artificial intelligence and metaheuristics, distribution management, and satellite scheduling.

Xiaolu Liu received the BEng degree from NUDT, China in 2006, and the PhD degree in management science and engineering from NUDT, China in 2011. She is an associate professor at College of Systems Engineering, NUDT, China. Her research mainly focuses on artificial intelligence and metaheuristics, distribution management, and satellite scheduling.

Weican Meng received the PhD degree from Zhengzhou Institute of Surveying and Mapping, China in 2015. He is a research assistant at Beijing Institude of Tranking and Telecommunication Technology, China. His research interests include remote sensing and geospatial intelligence.

Yingwu Chen received the BEng degree in automation, and MEng and PhD degrees in systems engineering from NUDT, China in 1984, 1987, and 1994, respectively. He has been the executive director of Systems Engineering Society of China since 2010. He is a distinguished professor at College of Systems Engineering, NUDT, China, where he focuses on management theory and its applications. He has authored more than 70 research publications. His current research interests include assistant decision-making systems for planning, decision-making systems for project evaluation, management decisions, and artificial intelligence.