Journal of Systems Engineering and Electronics
Vol. 32, No. 6, December 2021, pp.1407 – 1420

Real-time online rescheduling for multiple agile satellites with emergent tasks

WEN Jun[†], LIU Xiaolu[†], and HE Lei[†,*]

College of Systems Engineering, National University of Defense Technology, Changsha 410073, China

Abstract: The emergent task is a kind of uncertain event that satellite systems often encounter in the application process. In this paper, the multi-satellite distributed coordinating and scheduling problem considering emergent tasks is studied. Due to the limitation of onboard computational resources and time, common online onboard rescheduling methods for such problems usually adopt simple greedy methods, sacrificing the solution quality to deliver timely solutions. To better solve the problem, a new multi-satellite onboard scheduling and coordinating framework based on multi-solution integration is proposed. This method uses high computational power on the ground and generates multiple solutions, changing the complex onboard rescheduling problem to a solution selection problem. With this method, it is possible that little time is used to generate a solution that is as good as the solutions on the ground. We further propose several multi-satellite coordination methods based on the multi-agent Markov decision process (MMDP) and mixed-integer programming (MIP). These methods enable the satellite to make independent decisions and produce high-quality solutions. Compared with the traditional centralized scheduling method, the proposed distributed method reduces the cost of satellite communication and increases the response speed for emergent tasks. Extensive experiments show that the proposed multi-solution integration framework and the distributed coordinating strategies are efficient and effective for onboard scheduling considering emergent tasks.

Keywords: agile satellite scheduling, emergent task, onboard rescheduling, distributed coordinating, multi-solution integration.

DOI: 10.23919/JSEE.2021.000120

1. Introduction

The operation of the Earth observation satellite is always subject to various uncertainties, such as emergent tasks, changes of cloud cover, and uncertain breakdowns. To deal with the uncertainties, methods usually include proactive (offline) methods [1] and reactive (online) methods [2]. Proactive methods usually use the knowledge of uncertainties, such as the known probability distribution of future uncertainties, to maximize the expected revenue. Reactive methods are event-driven and they schedule tasks online. In this paper, we focus on the latter methods. We consider a satellite scheduling problem with random emergent tasks arriving in real-time and we assume that the information about the emergent tasks is unknown in advance.

The satellite scheduling problem is an over-subscribed problem, which means that the capacity cannot meet the demand. As a result, only a subset of tasks can be scheduled. For such over-subscribed problems with random task arrivals, some researchers use two-stage methods: first accepting the tasks and then scheduling the accepted ones. Wu et al. [3,4] studied task-accepting policies, which considered the cost to remove scheduled tasks on the timeline, the revenue of the new task, and the resources needed for the new task. If the task is accepted, it is scheduled in the closest time window. Kim et al. [5] proposed a heuristic algorithm to obtain a near-optimal solution of the formulated mixed-integer programming (MIP) based on the time windows pruning procedure. Arredondo et al. [6] and Snoek [7] used reinforcement learning to study the policies of acceptance. Other methods include dynamic programming [8], simulation-based rules [9], and meta-heuristics [10].

Although separating the accepting and scheduling problems could lead to lower complexity, the two problems are coupled, and separating them could affect the solution quality. In methods that treated the two problems as a whole, Rahman et al. [11] proposed a memetic algorithm, whose online running time is around tens of seconds; Su et al. [12] transformed the dynamic problem to several static ones by using the rolling horizon method. Similar methods were also adopted by Qiu et al. [13] and Liao et al. [14]. Xu et al. [15] proposed a dynamic programming method with online running time around tens of seconds. Wang et al. [16] proposed a simple but fast

Manuscript received October 22, 2020.
*Corresponding author.
†Co-first authors.
This work was supported by the National Natural Science Foundation of China (72001212, 71701204, 71801218) and the China Hunan Postgraduate Research Innovating Project (CX2018B020).

heuristic method, which could merge the emergent task into the scheduled ones. Similarly, Chien et al. [17] and Beaumet et al. [18] also used a heuristic-based greedy method to reschedule tasks with the aim of providing timely solutions, but the solution quality cannot be guaranteed. Wu et al. [19], Li et al. [20] and He et al. [21] used the ant colony optimization method. Chu et al. [22] proposed an exact branch and bound method, but the method adopts the anytime policy, which provides the best current solution if the time limit of calculation is reached. Cui et al. [23] proposed a dynamic scheduling algorithm based on mission priority with a hybrid genetic tabu search algorithm to obtain the initial satellite scheduling plan.

We can find that many online scheduling methods do not consider the online running time [3,24−26]. This might be acceptable for some problems. However, for the satellite scheduling problem, available time windows for tasks are usually quite short, and if the tasks are not scheduled timely, their time windows might be wasted. In addition, the onboard computer has very weak computational power that is around 10% of that of a typical personal computer on the ground [22]. Complex methods such as memetic algorithms [11] and dynamic programming [15] could take hundreds of seconds or even several hours, which is certainly unacceptable. As a result, we can see that, for the online scheduling problem of satellites, the methods are usually simple and greedy [16−18,27] or anytime [22], which sacrifices the solution quality to deliver timely solutions and increase the responsiveness of the satellite.

Another difficulty of this problem is to coordinate multiple satellites with limited communication opportunities. Current multi-satellite coordinating methods adopt a centralized manner [28−33], which uses one satellite or the ground station to generate solutions for all the satellites. Such methods are certainly not suitable for the problems with random emergent tasks, because they need a long time to calculate and transfer data with limited communication windows. Some researchers use the contract net protocol [34] or similar ideas [35], but such methods require frequent negotiation among satellites, which results in a high communication cost.

As can be seen from the above analysis, there are two questions this paper aims to answer. First, is it possible to generate near-optimal solutions with limited onboard computational resources within limited computation time? Second, is it possible to coordinate multiple distributed satellites to generate complementary solutions without much communication?

To answer the above questions, this paper proposes an efficient real-time online rescheduling algorithm for multiple agile earth observation satellites with random emergent tasks. The contribution of the paper are as follows:

(i) A multiple feasible solution integration framework for emergent task scheduling is proposed for the first time. The framework makes it possible for a satellite to generate solutions on the ground in a short time.

(ii) Multi-satellite distributed cooperative rescheduling methods based on the multi-agent Markov decision process and the mixed-integer programming are proposed. These methods enable the satellites in orbit to generate complementary and non-conflicting solutions without much communication.

(iii) In the experiment, the adaptability of different multi-satellite distributed cooperative strategies to scenarios with different characteristics is analyzed. These conclusions are helpful to guide the selection of appropriate methods for different types of problems in practical engineering applications.

2. Problem

In this paper, we consider a real-time online rescheduling problem for multiple agile Earth observation satellites with random emergent tasks. The problem consists of simultaneously selecting a subset of tasks to be observed from multiple orbits of multiple satellites, generating the associated schedule, and rescheduling the solution when random emergent tasks arrive online. The problem for scheduling the agile Earth observation satellite is difficult because each task has several long time windows and the transition time between any two adjacent tasks depends on the start time of the two tasks. The static version of the problem is already an non-deterministic polynominal hard (NP-hard) problem. In the considered dynamic version, emergent tasks arrive during the execution in real-time. The rescheduling must be done quickly, which adds difficulties to the problem. The goal of this research is to generate a rescheduling solution online quickly with limited onboard computational resources, which is almost as good as the solution by an offline omniscient solver.

2.1 Problem model

The problem can be described by a three-element tuple $\langle T,S,H\rangle$, where T is the task set, S is the satellite set, and $H=\{1,\cdots,\mathcal{H}\}$ is a limited scheduling interval (usually one day). $\mathcal{H}$ is the discrete scheduling step.

The task set includes regular tasks and emergent tasks. Each task t_i in T contains the following attributes: $\langle g_i,d_i,a_i,W_i,x_{ij},u_i,v_i\rangle$. g_i is the benefit of performing task t_i; $d_i\in H$ is the processing time of the task, which is the shortest continuous imaging time specified by the user; $a_i\in H\vee a_i=0$ is the random arrival time of the task, for

regular tasks $a_i = 0$ and for emergent tasks $a_i \in H$. The scheduling of emergent tasks can only be started after the task arrives. At the same time, the calculation time should be considered, that is, if the scheduling is not completed before the end of the visible time window of the emergent task, the emergent task will be abandoned. Emergent tasks may include three sources: new tasks submitted by users, new tasks discovered independently by satellites (such as volcanic activities and other natural disasters), and tasks that failed due to uncertain factors (such as cloud cover and satellite failures). W_i is the set of visible time windows for task t_i. $W_i = \{\langle s_{i1}, b_{i1}, e_{i1}\rangle, \cdots, \langle s_{i|S|}, b_{i|W_i|}, e_{i|W_i|}\rangle\}$ contains $|W_i|$ visible time windows, and each window is specified by the satellite $s \in S$ to which it belongs, b and e are the start time and the end time of the visible time window and $b, e \in H$, x_{ij} and u_i are decision variables and $x_{ij} \in \{0,1\}$ represents whether task t_i is observed in the jth time window, and u_i represents the observation start time of task t_i. v_i is the observation end time of the task, and $v_i = u_i + d_i$.

The solution to this problem can be expressed as $\{\mathrm{Sol}_1, \mathrm{Sol}_2, \cdots, \mathrm{Sol}_S\}$, and the solution of the satellite $s \in S$ is expressed as $\mathrm{Sol}_s = \{\mathrm{Sol}_s^1, \mathrm{Sol}_s^2, \cdots, \mathrm{Sol}_s^{\mathcal{H}}\}$, which defines the solution at each decision step. The scheduling sub-problem at each step can be regarded as a static scheduling problem. The scheduling goal is to maximize the total revenue of all selected tasks. The problem model is defined as follows:

$$\max \quad \sum_{i=1}^{|T|} \sum_{j=1}^{|W_i|} g_i x_{ij}$$

s.t.

$$\sum_{j=1}^{|W_i|} x_{ij} \leqslant 1, \quad \forall t_i \in T, \tag{1}$$

$$b_{ij} \leqslant u_i \leqslant e_{ij}, \quad x_{ij} = 1, \tag{2}$$

$$v_i + \tau_{ii'} \leqslant u_{i'}, \quad p_{ii'} = 1, \tag{3}$$

$$x_{ij} \in \{0,1\}, \quad t_i \in T, w_{ij} \in W_i, \tag{4}$$

$$p_{ii'} \in \{0,1\}, \quad t_i, t_{i'} \in T, \tag{5}$$

$$\tau_{ii'} \geqslant 0, \quad t_i \in T. \tag{6}$$

Constraint (1) is the uniqueness constraint, which means that each task can be observed at most once.

Constraint (2) is the visible time window constraint, which means that each task can only be observed in one of the visible time windows.

Constraint (3) is the attitude transition constraint, which means that the time interval between any two adjacent observations must be sufficient for the satellite to transit its observation attitude. $\tau_{ii'}$ represents the attitude transition time from task t_i to task $t_{i'}$, $\tau_{ii'} = \dfrac{|\theta_i - \theta_{i'}|}{a} + t_{\mathrm{stable}}$, where θ_i and $\theta_{i'}$ are the satellite attitude angle of the observation task t_i and task $t_{i'}$ respectively, a is the angular velocity of the satellite attitude maneuver, t_{stable} is the minimum stabilizing time required for satellite attitude maneuver, and $p_{ii'}$ is a binary variable with 1 representing task t_i and task $t_{i'}$ are two adjacent tasks.

Constraints (4)−(6) represent the domain of each variable.

2.2 Problem analysis

In current rescheduling algorithms, no information about unscheduled tasks is considered. However, this is a kind of important information that can be utilized. Here is an example to illustrate this point. In Fig. 1, the satellite can only choose from Tasks 5−8 or Tasks 1−4 to observe. With the assumption that Tasks 5−8 have higher revenue, the satellite will choose the optimal solution, which is observing Tasks 5−8. However, if an emergent Task 9 with very high revenue is submitted during the execution, a traditional task repair policy will not be able to insert Task 9 into the current solution. In addition, if emergent Task 9 arrives just before the satellite observes Task 5, a rescheduling process on the ground will not be timely enough. However, in this case, if the satellite retains the solution of observing Tasks 1−4, which is slightly worse than the optimal solution when submitting Task 9, the revenue of the solution will exceed the current optimal solution, and the satellite can quickly generate a new optimal solution without performing a series of scheduling calculations (for example, to find and compare a variety of revenues of sequencing Tasks 1−9).

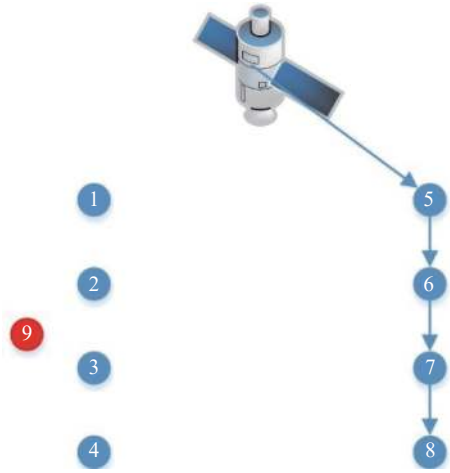


Fig. 1 Example of satellite observation

The idea proposed in this paper is to transfer the com-

plex online rescheduling process to offline scheduling on the ground. The core method is to use the powerful computing power of the ground to generate multiple feasible solutions, including feasible scheduling solutions of tasks outside the scheduling scheme that cannot be contained by a single optimal solution.

3. Method

We propose a new framework for the problem as shown in Fig. 2, which includes an offline scheduling and training stage and an online rescheduling stage. The idea to achieve an efficient online rescheduling method with limited onboard computational resources is to move the difficulties of scheduling multiple agile satellites (i.e., selecting the tasks and sequencing them with the time-dependency) to the ground. Instead of generating a single optimal solution, the offline training stage generates multiple high-quality solutions. When an emergent task arrives, the satellite can select a solution to insert the emergent task into it quickly without carrying out too much calculation.

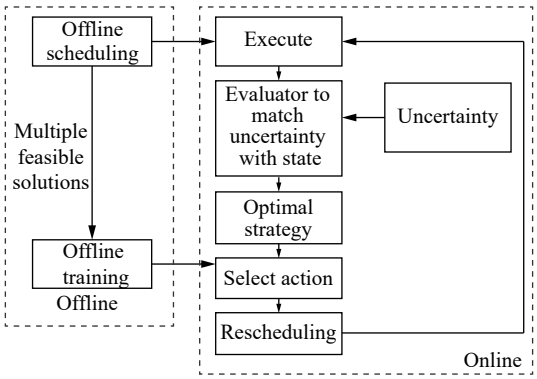


Fig. 2 Rescheduling algorithm framework

In the offline stage, we use a meta-heuristic algorithm called the adaptive large neighborhood search (ALNS) to generate multiple high-quality solutions. When an emergent task arrives, the satellite only needs to compare the multiple solutions and choose the best one. Since the multiple solutions might have repetitive observations which could reduce the revenue, we use a multi-agent Markov decision process (MMDP) model to train an optimal policy to decouple the multiple solutions. With the trained policy, we make sure that the solutions selected by the satellites in different sections have the highest revenue and the least repetitive observations.

In the online stage, we adopt an efficient insertion policy that determines the feasibility of insertion according to pre-computed time slacks considering time-dependent transition times. The solution into which the emergent task can be inserted and which has the highest revenue will be selected as the new solution replacing the current solution in the corresponding section. If the new solution has the same tasks that are also in the solutions in other sections, these solutions will be changed to other ones according to the offline trained policy. Another problem is that the optimal policy will become less accurate when many emergent tasks are inserted because it is trained based on offline generated solutions. To solve this problem, we also propose to solve the same MMDP model onboard to decouple the multiple solutions with accurate solution information.

3.1 Multiple feasible solutions generated

The basic method to generate multiple feasible solutions adopts the adaptive large neighbourhood search with tabu search, partial sequence dominance, fast insertion (ALNS/TPF) method proposed in [36,37] and the adaptive task assigning large neighbourhood search (A-ALNS) method proposed in [38]. A-ALNS is mainly oriented to multi-satellite scheduling and generates an initial solution to assign tasks to different satellites. ALNS/TPF is mainly applied to the scheduling of tasks that have been assigned to a single satellite, and multiple optimal and feasible solutions can be saved through running multiple iterations. The generation steps are as follows:

Step 1 Use the A-ALNS algorithm to generate a high-quality initial solution.

Step 2 Assign the successfully-scheduled tasks to the corresponding satellites.

Step 3 Assign the unsuccessfully-scheduled tasks to all satellites.

Step 4 For the new task set on each satellite, the ALNS/TPF algorithm is used to get several new feasible alternative solutions. During the solution process, $n-1$ feasible solutions with the highest revenue are saved.

A process for generating multiple alternative solutions is shown in Fig. 3, where Steps 2 and 3 are omitted.

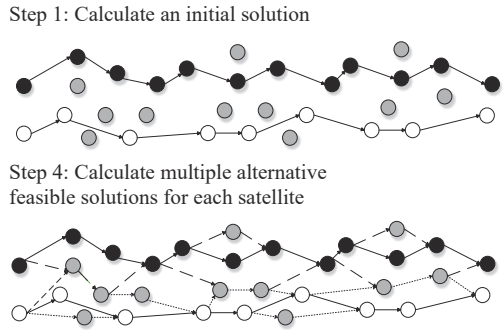


Fig. 3 Multiple alternate feasible solutions

In Step 1, the A-ALNS algorithm is used to generate a high-quality initial solution. Note that this "initial solution" is defined differently from the "initial solution" in

common heuristics, the initial solution here is the best solution found by using the A-ALNS algorithm. Its purpose is to assign tasks to different satellites and then use the single-satellite algorithm to generate multiple feasible solutions on the task set of each satellite to achieve effective coverage of solution space. However, one problem to be considered is that there may be cases of observing the same task among multiple feasible solutions of multiple satellites, and repeated observations will lead to reduced benefits. In order to avoid that problem, this method will only assign unscheduled tasks in the A-ALNS to different satellites. Other tasks successfully scheduled by A-ALNS will only be executed on their selected satellites, and the executing satellites will not be changed in the subsequent scheduling process. In Step 4 of Fig. 3, unscheduled tasks are assigned to different satellites, and the task set on each satellite changes. At this time, the ALNS/TPF method is used to obtain the new task set, and $n-1$ feasible solutions with the highest revenue are retained in the solution process. Finally, on each satellite, a total of n feasible solutions are generated. In these feasible solutions, except for those unscheduled tasks, there will be no repeated observations.

3.2 Multi-satellite distributed cooperation approach

Until now, several alternative feasible solutions have been generated for each satellite. When the emergent task arrives, the online evaluator will check whether the emergent task can be inserted into each feasible solution. This process adopts the quick insertion method introduced in [36] to quickly determine whether each task can be inserted. Each satellite then decides which feasible solution it will choose to perform the emergent task. However, due to the existence of multiple feasible solutions for unscheduled tasks, the feasible solutions selected by each satellite may include certain tasks that have been observed by other satellites. In order to coordinate the feasible solutions of multiple satellite selection, three cooperation mechanisms are proposed in this paper.

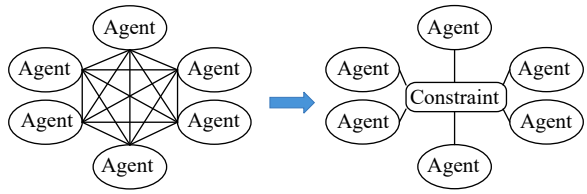
(i) Greedy selection mechanism

This is the simplest selection mechanism. Each satellite only records its multiple feasible solutions. When the emergent task arrives, each satellite invokes an evaluator to evaluate the revenues when inserting the emergent task into each feasible solution and chooses the one with the highest revenue. This method can quickly select a feasible solution, but it is difficult to avoid the situation that the same task is observed by multiple satellites simultaneously.

(ii) Optimal cooperative policy mechanism based on the multi-agent Markov decision process

If each satellite is assumed to be an agent, multiple satellites can constitute a multi-agent system.

One way to solve the problem of repeated observations is to adopt multi-agent communication methods such as a contract network [37] to coordinate the decision-making of each satellite. However, this method which requires real-time information communication has a high application cost. Another method is that each agent records and tracks the current state of other satellites. Through the optimal feasible solution selection policy corresponding to each state combination of offline calculation, the feasible solution can be quickly selected with less communication and avoid repetition. However, since each satellite contains multiple feasible solutions, the joint matrix size of its combined state will increase rapidly with the increase of the number of agents and feasible solutions, which makes it impossible to calculate an effective selection policy. The specific application method is presented in Fig. 4. As shown in Fig. 4(a), each agent needs to consider the joint state of all other agents. In order to solve this problem, the constraint of multi-agent is used to decouple the joint state among agents [39], as shown in Fig. 4(b).



(a) Joint state (b) Decoupling joint state

Fig. 4 Multi-agent decoupling

In order to describe the constraints among solutions, an intermediate variable $P_i(t,a)$ is introduced, which is defined as follows:

$$P_i(t,a)=\begin{cases}1, & \text{task } t \text{ is in solution } a \text{ of agent } i\\ 0, & \text{else}\end{cases}. \quad (7)$$

Other variables of the model are defined as follows:

$\boldsymbol{A}_i$: The action space of agent i. It can be represented by a $2\times n$ matrix.

$\boldsymbol{S}_i$: The state space of agent i. The state space of agent i can be represented by an $n(n+1)$ matrix. The row number of the matrix represents the currently executed solution. In the matrix, the n column represents the best solution that the current emergent task can be inserted into, and the $n+1$ column indicates that the emergent task cannot be inserted into any solutions.

$\boldsymbol{T}_i(s,a,s')$: The state transition matrix of agent i, which represents the probability that agent i transit from state s to s' when performing action a.

$\boldsymbol{R}_i(s,a)$: The revenue matrix of agent i, which represents the revenue of executing action a when agent i is in

state s. The revenue is equal to the revenue of the feasible solution. However, in order to encourage the agent to choose the optimal solution given by the evaluator, when the agent chooses the solution given by the evaluator, the agent's revenue increases by 10 units.

$x^i_{h,s,a} \in [0,1]$: Decision variable, which represents the probability that agent i will perform action a when the state is s at the hth decision step.

H: Scheduling horizon. $h \in H$. Every time an emergent task arrives, a decision is made.

M: The maximum number of observations for a single task. The method uses this value to limit the number of repeated observations.

This MMDP is constructed as a linear programming (LP) model [40,41] which is defined as follows:

$$\max \sum_{i=1}^{n}\sum_{h=1}^{H}\sum_{s\in \boldsymbol{S}_i}\sum_{a\in \boldsymbol{A}_i} x^i_{h,s,a}\cdot \boldsymbol{R}_i(s,a) \tag{8}$$

s.t.

$$\sum_{a\in \boldsymbol{A}_i} x^i_{h+1,s',a} = \sum_{s\in \boldsymbol{S}_i}\sum_{a\in \boldsymbol{A}_i} x^i_{h,s,a}\cdot \boldsymbol{T}_i(s,a,s'),\ \ \forall i,h,s' \in \boldsymbol{S}_i, \tag{9}$$

$$\sum_{a\in \boldsymbol{A}_i} x^i_{1,s,a} = T_{1,i}(s),\ \ \forall i, s \in \boldsymbol{S}_i, \tag{10}$$

$$\sum_{i=1}^{n}\sum_{s\in \boldsymbol{S}_i}\sum_{a\in \boldsymbol{A}_i, a\%2=0} x^i_{h,s,a} \leqslant M,\ \ \forall h \in H, \tag{11}$$

$$\sum_{i=1}^{n}\sum_{s\in \boldsymbol{S}_i}\sum_{a\in \boldsymbol{A}_i} x^i_{h,s,a} P_i(t,a) \leqslant M,\ \ \forall h \in H, t \in T, \tag{12}$$

$$\sum_{i=1}^{n}\sum_{s\in \boldsymbol{S}_i, s\%(n+1)}\sum_{a\in \boldsymbol{A}_i, a\%2=0} x^i_{h,s,a} = 0,\ \ \forall h \in H, \tag{13}$$

$$0 \leqslant x^i_{h,s,a} \leqslant 1,\ \ \forall i,h,s,a. \tag{14}$$

The objective function (8) maximizes the sum of the profits of all actions. Constraints (9) and (10) are the constraints that maintain the state transition probability, where $T_{1,i}(s)$ represents the initial state probability. Constraints (11) and (12) limit the number of repeated observations. Constraint (11) restricts the number of observations for emergent tasks, and constraint (12) limits the number of observations for each ordinary task. Constraint (13) means that when the emergent task cannot be inserted, the agent will not choose to observe the emergent task. Constraint (14) represents the domain of the decision variable.

The MMDP model requires that each agent can make independent decisions without retaining the feasible solution information of other agents and relying on frequent information interaction. However, with the increase of the number of decision-making steps, the accuracy of each agent's decision-making reasoning for other agents will decrease. Therefore, if a periodic communication mechanism is introduced, accurate state information can be obtained among agents, and the accuracy of decision-making can be improved. During communication, the ground station collects information from the satellite and calculates the new optimal policy of the satellite before the next communication phase. Let $\hat{H} < H$ be the communication cycle, h_{com} be the moment of communication, and s_{com} be the accurate state information during the communication, then the above objective function can be replaced by

$$\max \sum_{i=1}^{n}\sum_{h=h_{\text{com}}}^{\min(h_{\text{com}}+\hat{H},H)}\sum_{s\in \boldsymbol{S}_i}\sum_{a\in \boldsymbol{A}_i} x^i_{h,s,a}\cdot \boldsymbol{R}_i(s,a).$$

Constraints (8) are changed to match the current state:

$$\sum_{a\in \boldsymbol{A}_i} x^i_{h_{\text{com}},s_{\text{com}},a} = 1,\ \ \forall i. \tag{15}$$

(iii) Optimal selection mechanism based on MIP

The biggest drawback of the method introduced in the previous section is that each agent is required to make independent decisions without knowing the decisions of other agents, which will lead to an inevitable decline in revenue over time. If each agent can record the solutions of other agents and update the information with a deterministic method, then multiple agents can maintain a higher revenue without communication. Next, we will introduce an optimal selection mechanism based on MIP.

The MIP model contains the following variables:

$\boldsymbol{A}_i$: Agent i's action space. Agent i contains n optional actions, respectively corresponding to n alternative feasible solutions.

$P_i(t,a)$: The intermediate variable, as defined in (7), representing whether task t is executed when action a is being performed (that is, choosing a feasible solution).

$\boldsymbol{R}_i(a)$: The revenue matrix of agent i, which represents the revenue of agent i's execution of action a, namely the revenue of feasible solution a. The state is not included here, because the MIP is called on at each step, corresponding to the current state.

$x_{ia} \in \{0,1\}$: Decision variables, representing whether agent i selects action a.

The model is defined as

$$\max \sum_{a}\sum_{i} x_{ia}R_i(a) - \sum_{t}\max(0, \sum_{a}\sum_{i} x_{ia}P_i(t,a) - 1)g_t \tag{16}$$

s.t.

$$\sum_{a} x_{ia} = 1, \ \forall i, \tag{17}$$

$$x_{ia} \in \{0,1\}, \ \forall i, a. \tag{18}$$

In the objective function (16), g_t represents the revenue of task t. The goal of the first half of the objective function is to maximize the sum of the benefits of all the selected solutions; the latter half means that the benefits of all repeated observation tasks are subtracted. Constraint (17) means that each agent can only choose one feasible solution when making a decision. Constraint (18) shows the domain of the decision variable.

In order to ensure the decision synchronization of each agent, each agent must record other agents' information on the feasible solutions, which will consume part of the storage resources. At the same time, different from the method introduced in Section 2, this method requires the MIP model to be re-invoked for solving each decision, while the MMDP in Section 2 is only solved once for each satellite communication, so its efficiency will be higher. Despite this, the efficiency of this MIP-based optimal selection mechanism is still higher than the traditional online rescheduling method, because the traditional method solves the rescheduling problem online, and the number of constraints and variables to be considered is much higher than the MIP problem proposed here.

4. Results

The algorithm is written in the C# language and runs on a computer with Intel Core i5-3470 3.20 GHz CPU, 8 GB memory, and 64-bit Windows 7 system. The LP and MIP models are solved by IBM ILOG CPLEX 12.8 [42].

4.1 Design and generation of examples

In order to verify the effectiveness of the proposed algorithm, we generate multiple benchmark instances. The method of generating instances is to randomly generate point targets on the whole world, which contains 12 instances in total. The number of tasks varies from 100 to 400, with the increment step being 100, and each contains one to three satellites. In order to make the instance contain emergent tasks, we randomly select 20% of the tasks from each instance as emergent tasks. When the number of tasks is greater than 200, only 40 tasks are selected as emergent tasks. The scheduling horizon of the instance is from 00:00:00 to 24:00:00, April 20, 2017. During this horizon, each satellite contains about 15−16 orbits. The six orbital parameters of the satellite are the semimajor axis ax, eccentricity e, inclination ic, perigee angle ω, right ascension of ascending node (RAAN) and true perigee angle m. The initial orbit parameters of all three satellites used are shown in Table 1 .

Table 1 Satellite orbit parameters

Satellite	ax	e	ic	ω/(°)	RAAN	m
Satellite 1	7 200 000	0.000 627	96.576	0	175.72	0.075
Satellite 2	7 200 000	0.000 627	96.576	0	145.72	30.075
Satellite 3	7 200 000	0.000 627	96.576	0	115.72	60.075

The parameters of ALNS/TPF and A-ALNS algorithms are the same as those introduced in the corresponding papers, and the other parameters introduced in this paper are fixed as follows:

Number of alternative feasible solutions: $n = 5$.

Schedule horizon based on the MMDP method: $H = 40$. The rescheduling is performed once every emergent task arrives. Thus the length of the scheduling horizon should be greater than the number of emergent tasks.

Observation limit of each decision for a task based on the MMDP method: $M = 2$.

In the experiment, five methods were compared. First, full rescheduling (FR) algorithm. Every time a new task arrives, A-ALNS is called on to regenerate a new scheduling solution. Second, online repair based on a single feasible solution algorithm (single-solution repair, SSR). Third, rescheduling methods based on the greedy selection (GS) policy. Fourth, the optimal cooperative policy mechanism based on the MMDP (MMDP-based optimal policy, MMDPOP). Fifth, the optimal selection mechanism based on MIP (MIP-based selection, MIPS). It should be pointed out that the purpose of the experiment in this section is to verify the effectiveness of the proposed multiple feasible solutions mechanisms and at the same time to compare the performance of multiple multi-satellite coordination mechanisms. Therefore, the onboard rescheduling method uses a simple insertion policy: that is, the insertion policy is adopted to only quickly check the feasibility of task insertion. If the insertion cannot be performed, the task is abandoned and the

task that has been successfully scheduled is not considered for cancellation.

4.2 Analysis of experimental results

In the following experiment, each instance is run 10 times, and the results in the figures and tables are the average of revenue and running time of 10 runs.

Fig. 5 shows the rescheduling of different algorithms when the number of satellites is one. When the number of satellites is one, the MIPS and GS algorithms are the same, therefore the revenue of the algorithm is almost the same. Because the MMDPOP sometimes chooses not to observe emergent tasks conservatively, the revenues are slightly lower than those of the other two algorithms. As the SSR algorithm only contains a single feasible solution, when the number of tasks increases, the probability that the emergent task can be inserted is lower. As there is no other feasible solution in the SSR algorithm, its revenue gap with other algorithms gradually increases, which also proves the effectiveness of the alternative feasible solution in the algorithm framework proposed in this paper. The FR algorithm has a greater degree of freedom because it fully reschedules the instance when the emergent task arrives, and should theoretically have the highest revenues if given sufficient computing time. However, it can be observed that the FR algorithm only shows advantages for 100 and 200 tasks. Since FR is based on A-ALNS, which only generates a single feasible solution and could not provide global optimal solutions when the solution space is too large, for large-scale instances with 300 and 400 tasks, the gap to global optimum of FR becomes larger and the synthetic value of generating multiple feasible solutions and combining them exceeds a single solution. This explains why FR sometimes performs worse than other methods as shown in Fig. 5(c) and Fig. 5(d). It should be pointed out that the calculation time of using FR is less than the offline calculation time of other algorithms because other algorithms need to calculate multiple feasible solutions and search for more solution space. At the same time, the FR algorithm belongs to a centralized rescheduling method and is an ideal method requiring that all tasks on all satellites are rescheduled every time an emergent task arrives. This exerts great challenges on the satellites' management, control, and communication system and is not applicable under the current technical conditions.

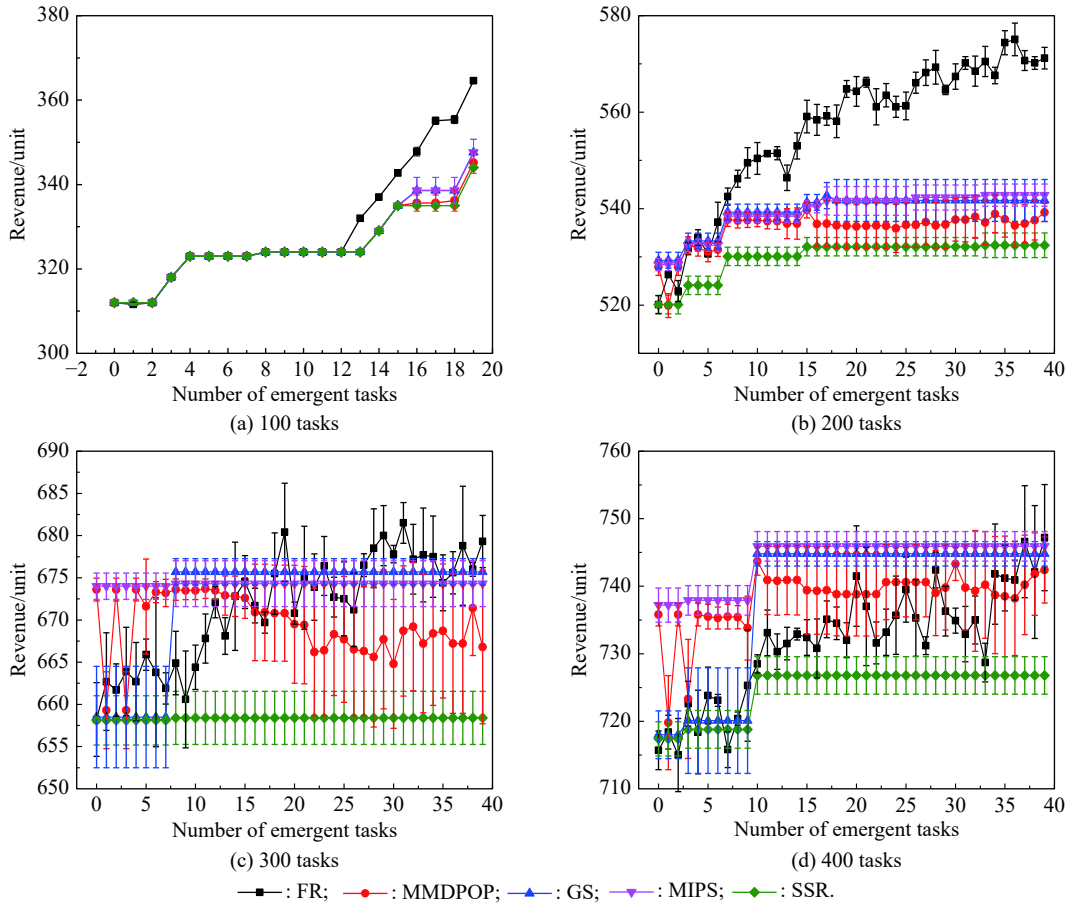


Fig. 5 Comparison of rescheduling results where there is one satellite

When the number of satellites is 2 (as shown in Fig. 6), due to the increase in the number of tasks that can be observed, a single satellite can have a larger space for inserting emergent tasks, so the performance of the SSR algorithm is better than that when there is only a single satellite, and it is also more obvious when the number of satellites is three (as shown in Fig. 7), which even exceeds the MMDPOP algorithm on a smaller scale. The advantages of the MIPS algorithm over other algorithms, especially relative to GS, are more obvious when the number of satellites is large. It can be seen that in Fig. 6 (d) and Fig. 7 (d), the performance of the MIPS algorithm far exceeds the GS algorithm and even exceeds the FR algorithm. It shows the effectiveness of the proposed optimal selection mechanism based on MIP. However, the disadvantage of the MIPS algorithm is that when emergent tasks arrive, the task insertion and the MIP solving operation need to be performed on each satellite, and the online calculation time is longer. Although MMDPOP does not need to resolve the optimal policy, its revenue is not significantly better than the GS algorithm, and it only shows a certain advantage when the number of tasks and the number of satellites are large. Because the GS algorithm is a greedy selection of feasible solutions when the number of tasks and the number of satellites increases, the rate of repeated observations among feasible solutions increases. At the same time, the optimal policy trained using the MMDPOP algorithm can prevent repeated observations to a certain extent, so the performance is better.

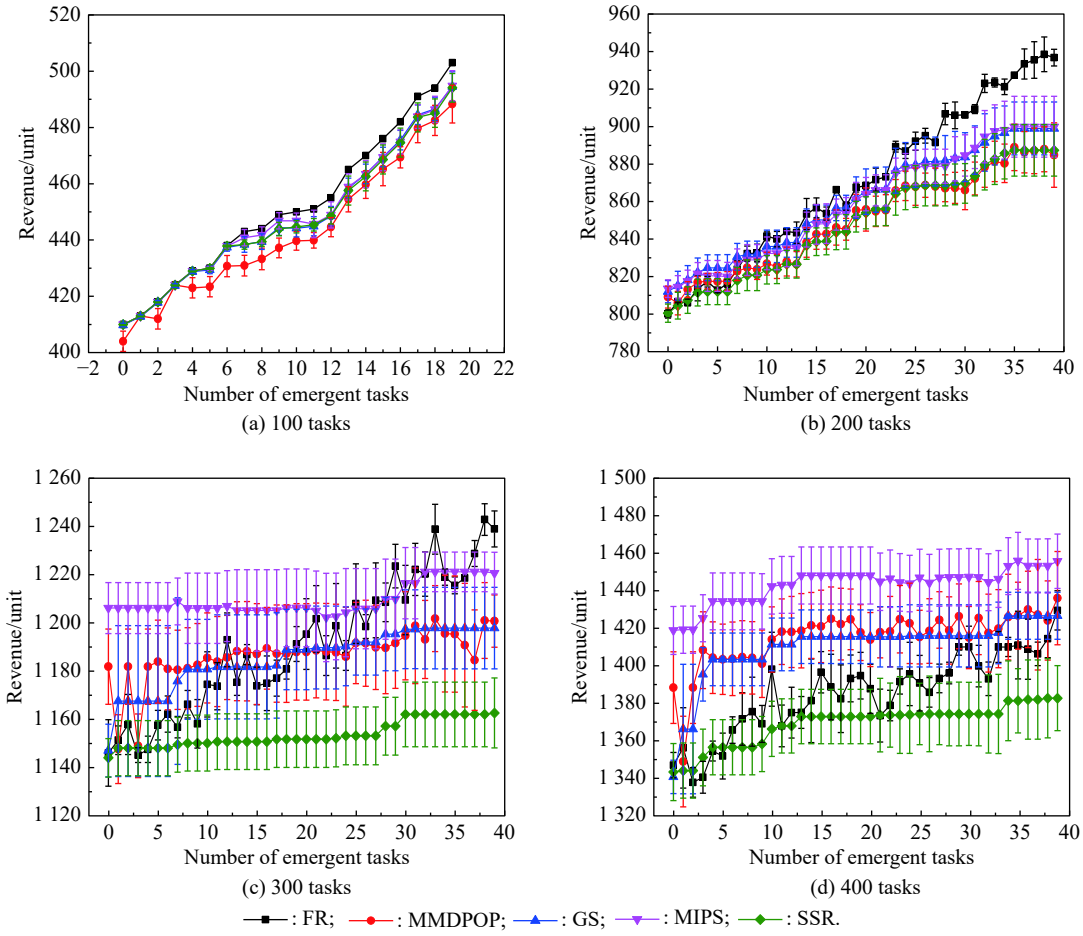


Fig. 6 Comparison of rescheduling results where there are two satellites

The average objective values of all the compared methods are shown in Table 2.

The average time for each algorithm to perform an online rescheduling process on different instances is shown in Table 3. Among them, the FR algorithm consumes the longest time because the solution is completely rescheduled; the MMDPOP, GS, and SSR algorithms only insert tasks and select the solution according to the trained policy in the online phase, so the calculation time is very short; the MIPS consumes relatively longer time because it needs to solve the MIP model online. According to the data in [22,43], onboard computers are usually 10 to 1 000 times slower than typical computers on the ground. Even if the time is multiplied by 1 000, the longest reschedul-

ing time of the MIPS algorithm is 46 s, which is still within the acceptable range. However, the FR algorithm takes up to 10 744.4 s (nearly 3 h). Obviously, although the FR algorithm has the best solution quality, it requires real-time communication between the satellite and the ground, and the calculation time is too long. It is not suitable for solving this problem.

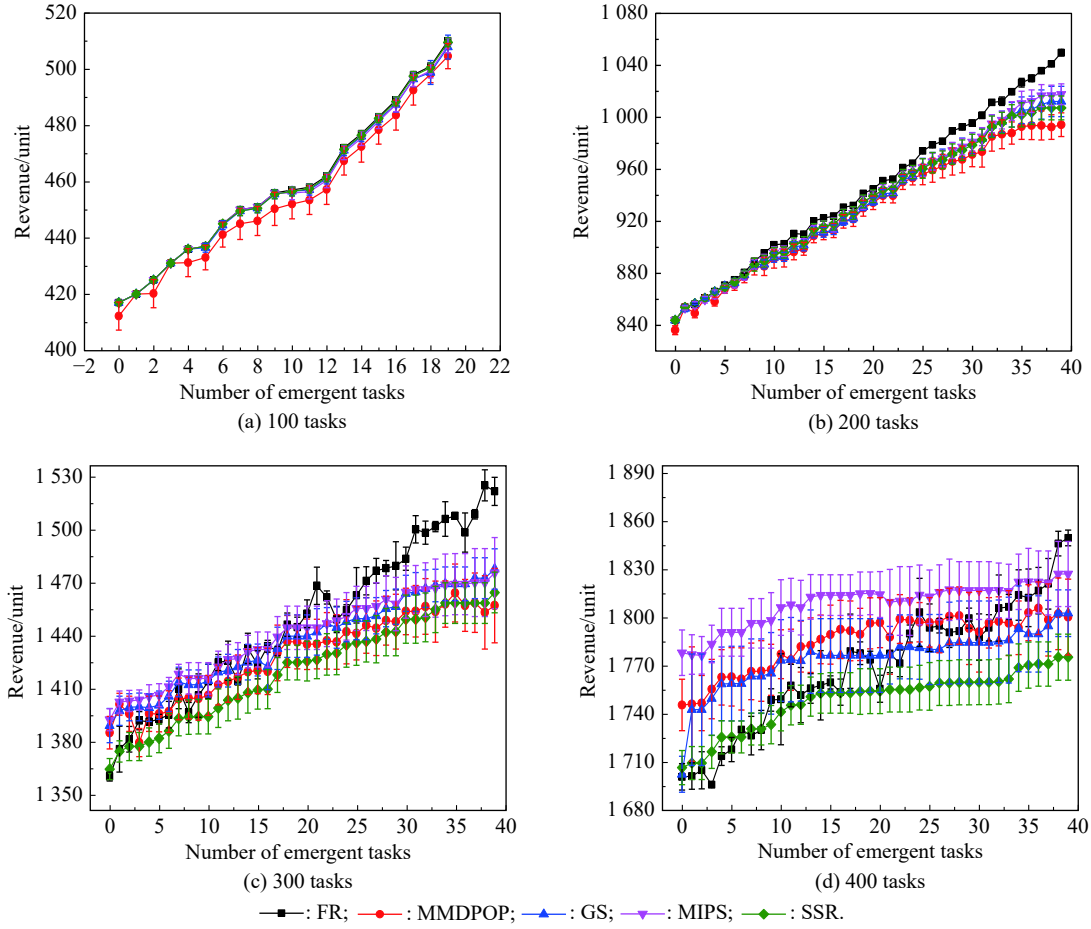


Fig. 7 Comparison of rescheduling results where there are three satellites

Table 2 Comparison of average objective values of different algorithms

Satellite	Number of tasks	FR	MMDPOP	GS	MIPS	SSR
1	100	330.01	325.33	325.87	325.87	325.15
1	200	556.12	535.77	539.46	539.32	530.05
1	300	671.67	669.51	672.26	674.24	658.34
1	400	731.87	738.24	738.47	743.85	724.70
2	100	451.75	442.75	447.61	448.21	447.42
2	200	870.55	849.99	860.48	859.14	848.56
2	300	1 190.67	1 186.92	1 185.50	1 209.66	1 153.84
2	400	1 385.66	1 415.30	1 410.54	1 443.27	1 369.31
3	100	458.75	454.49	458.00	457.98	458.43
3	200	945.23	928.32	933.77	935.87	934.42
3	300	1 446.54	1 427.96	1 435.84	1 439.00	1 419.59
3	400	1 770.61	1 785.36	1 775.09	1 809.00	1 749.47
Average	–	900.78	896.66	898.57	907.12	884.94

Table 3 Comparison of average calculation time of online rescheduling of different algorithms s

Satellite	Number of tasks	FR	MMDPOP	GS	MIPS	SSR
1	100	1.669 8	<0.000 1	<0.000 1	0.008 2	<0.000 1
1	200	4.086 4	<0.000 1	<0.000 1	0.013 4	<0.000 1
1	300	7.008 9	<0.000 1	<0.000 1	0.017 4	<0.000 1
1	400	9.934 2	<0.000 1	<0.000 1	0.023 8	<0.000 1
2	100	1.495 8	<0.000 1	<0.000 1	0.008 7	<0.000 1
2	200	3.962 3	0.000 1	0.000 1	0.017 1	<0.000 1
2	300	7.206 8	0.000 1	0.000 1	0.028 0	0.000 1
2	400	10.744 4	0.000 1	0.000 1	0.037 9	0.000 1
3	100	1.454 7	0.000 1	0.000 1	0.008 6	<0.000 1
3	200	3.588 9	0.000 1	0.000 1	0.018 3	0.000 1
3	300	6.624 7	0.000 1	0.000 1	0.029 5	0.000 1
3	400	10.120 3	0.000 2	0.000 2	0.046 0	0.000 1
Average	–	5.650 7	0.000 1	0.000 1	0.023 9	0.000 1

Fig. 8 shows the effect of different numbers of alternative feasible solutions on the revenue of solutions. Fig. 8 shows the revenue and calculation time of the MIPS algorithm for the instance with two satellites and 200 tasks. As can be seen from Fig. 8(a), with the increase of the number of feasible solutions, the revenue of the solution also increases, but the online calculation time also becomes longer as shown in Fig. 8(b). Since the online repair algorithm used in this paper only considers the insertion of tasks and does not consider the deletion of scheduled tasks, when the number of reserved alternative feasible solutions increases, the feasible solution with lower revenue but larger insertion space is also retained, and more tasks can be inserted in the online rescheduling process.

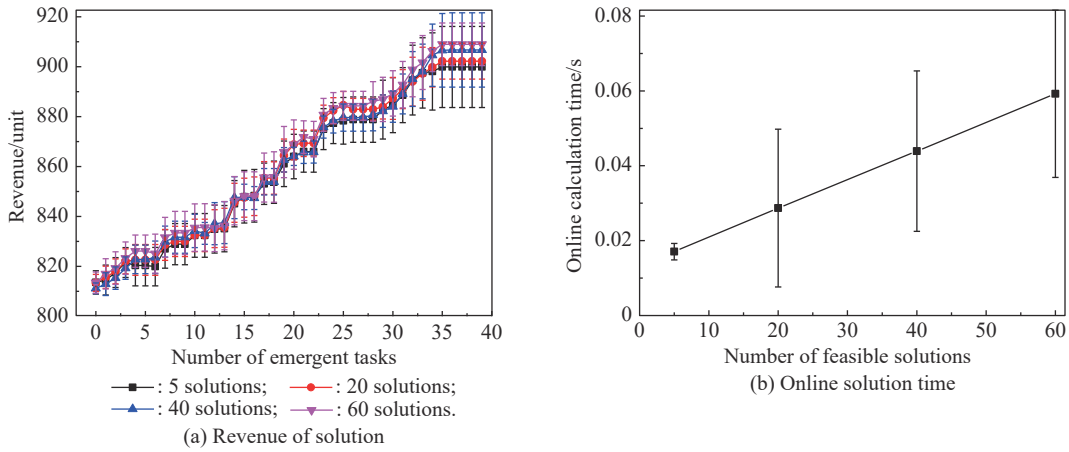


Fig. 8 Influence of different numbers of alternative feasible solutions on solution quality

In Section 3, we introduce the MMDPOP method that can improve the accuracy of the training optimal policy through communications. Fig. 9 shows the influence of different communication times on the solution quality. It can be seen that with the increase in communication times, the solution quality gradually improves.

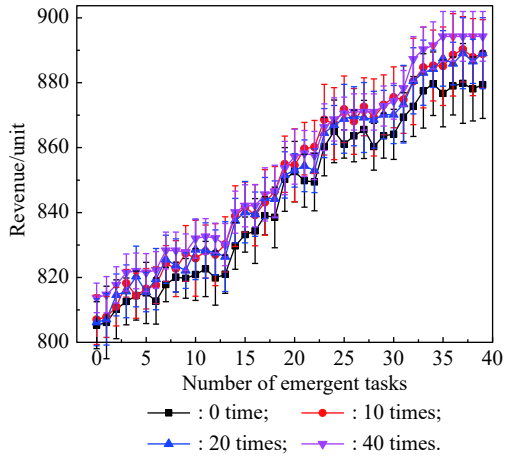


Fig. 9 Influence of different communication times on the quality of solution

5. Conclusions

The problem of multi-satellite distributed autonomous cooperative rescheduling considering emergent tasks is studied. Due to the limitations of onboard computational resources and time, the current common online rescheduling methods usually use simple greedy algorithms to solve this type of problem and sacrifice a certain solution quality for the timely response. In order to better solve this problem, a re-scheduling framework based on multi-solution integration is proposed, which can use the powerful computing power on the ground to generate multiple feasible solutions and convert the complex onboard re-scheduling problem into a feasible solution selection problem. This makes it possible to use a short time on the satellite to obtain a solution that is not worse than that obtained on the ground; a multi-satellite distributed cooperative policy based on the MMDP and MIP is proposed. These methods enable satellites to make independent decisions while in orbit and generate solutions with complementary advantages without conflict. Compared with the traditional centralized coordination method, the distributed coordination method reduces the cost of inter-satellite communication and improves the response speed of autonomous satellites to emergent tasks.

Through multiple sets of simulation instances, the effectiveness of the multi-solution integration framework and distributed cooperative policy proposed in this paper is proved for solving the on-board rescheduling problem. At the same time, experiments show that the optimal cooperative policy mechanism based on MMDP has a shorter calculation time and a poorer solution quality, but its solution quality can be improved through regular communication and training of more accurate optimal cooperative policies; while the optimal selection mechanism based on MIP requires longer online calculation time but with better solution quality. In some large-scale calculation examples, the solution quality even exceeds the complete rescheduling method.

This paper proposes a new integration method of multiple selections of feasible solutions to improve the efficiency of online rescheduling. However, when multiple alternative feasible solutions are generated, simple multiple iterations are used to retain the best several different feasible solutions. The difference between these feasible solutions may be small, and the coverage of the solution space is poor. A more ideal way is to use a certain method to control the difference among the solutions while using the least number of solutions to achieve more even and effective coverage of the solution space. Using swarm intelligence methods and controlling the diversity of populations may be an effective method. In addition, although the current multiple feasible solutions contain the same tasks, they are independent of each other. When inserting emergent tasks online, it needs to be calculated once on each feasible solution. In future research, a certain graph-based structure will be considered to manage multiple feasible solutions, so that a large number of feasible solutions can be rescheduled at the same time, further improving the efficiency of the online solution. Since the onboard computational resources are still very limited, if our efforts could further decrease the requirements on the computational power of the rescheduling process, more power could be used by other systems on the satellites and lots of costs of the aerospace chips could be saved.

References

[1] AYTUG H, LAWLEY M A, MCKAY K, et al. Executing production schedules in the face of uncertainties: a review and some future directions. European Journal of Operational Research, 2005, 161(1): 86–110.

[2] VIEIRA G E, HERRMANN J W, LIN E. Rescheduling manufacturing systems: a framework of strategies, policies, and methods. Journal of Scheduling, 2003, 6(1): 39–62.

[3] WU M C, CHEN S Y. A cost model for justifying the acceptance of rush orders. International Journal of Production Research, 1996, 34(7): 1963–1974.

[4] WU M C, CHEN S Y. A multiple criteria decision-making model for justifying the acceptance of rush orders. Production Planning & Control, 1997, 8(8): 753–761.

[5] KIM J, AHN J. Task and attitude control scheduling of multiple agile satellites considering task-dependent transition time. Proc. of the AIAA Scitech 2020 Forum, 2020. DOI: 10.2514/1.I010775.

[6] ARREDONDO F, MARTINEZ E. Learning and adaptation of a policy for dynamic order acceptance in make-to-order manufacturing. Computers & Industrial Engineering, 2010, 58(1): 70–83.

[7] SNOEK M. Neuro-genetic order acceptance in a job shop setting. Proc. of the 7th International Conference on Neural Information Processing, 2000: 815–819.

[8] YANG W, FUNG R Y K. Stochastic optimization model for order acceptance with multiple demand classes and uncertain demand/supply. Engineering Optimization, 2014, 46(6): 824–841.
[9] NANDI A, ROGERS P. Using simulation to make order acceptance/rejection decisions. Simulation, 2004, 80(3): 131–142.
[10] GHOMI S M T F, IRANPOOR M. Earliness-tardiness-lost sales dynamic job-shop scheduling. Production Engineering, 2010, 4(2/3): 221–230.
[11] RAHMAN H F, SARKER R, ESSAM D. A real-time order acceptance and scheduling approach for permutation flow shop problems. European Journal of Operational Research, 2015, 247(2): 488–503.
[12] SU L H, CHOU F D. Heuristic for scheduling in a two-machine bicriteria dynamic flowshop with setup and processing times separated. Production Planning & Control, 2000, 11(8): 806–819.
[13] QIU D S, HE C, LIU J, et al. A dynamic scheduling method of earth-observing satellites by employing rolling horizon strategy. The Scientific World Journal, 2013. DOI: 10.1155/2013/304047.
[14] LIAO D Y, YANG Y T. Imaging order scheduling of an earth observation satellite. IEEE Trans. on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2007, 37(5): 794–802.
[15] XU L, WANG Q, HUANG S M. Dynamic order acceptance and scheduling problem with sequence-dependent setup time. International Journal of Production Research, 2015, 53(19): 5797–5808.
[16] WANG J J, ZHU X M, YANG L T, et al. Towards dynamic real-time scheduling for multiple earth observation satellites. Journal of Computer and System Sciences, 2015, 81(1): 110–124.
[17] CHIEN S, TROESCH M. Heuristic onboard pointing rescheduling for an earth observing spacecraft. Proc. of the 25th International Conference on Automated Planning and Scheduling, 2015: 1−13.
[18] BEAUMET G, VERFAILLIE G, CHARMEAU M C. Feasibility of autonomous decision making on board an agile earth-observing satellite. Computational Intelligence, 2011, 27(1): 123–139.
[19] WU G H, MA M H, ZHU J H, et al. Multi-satellite observation integrated scheduling method oriented to emergency tasks and common tasks. Journal of Systems Engineering and Electronics, 2012, 23(5): 723–733.
[20] LI Y Q, WANG R X, XU M Q. Rescheduling of observing spacecraft using fuzzy neural network and ant colony algorithm. Chinese Journal of Aeronautics, 2014, 27(3): 678–687.
[21] HE L, LIU X L, CHEN Y W, et al. Hierarchical scheduling for real-time agile satellite task scheduling in a dynamic environment. Advances in Space Research, 2019, 63(2): 897–912.
[22] CHU X G, CHEN Y N, TAN Y J. An anytime branch and bound algorithm for agile earth observation satellite onboard scheduling. Advances in Space Research, 2017, 60(9): 2077–2090.
[23] CUI J T, ZHANG X. Application of a multi-satellite dynamic mission scheduling model based on mission priority in emergency response. Sensors, 2019, 19: 14–30.
[24] WANG D J, YIN Y Q, CHENG T C E. Parallel-machine rescheduling with job unavailability and rejection. Omega, 2018, 81: 246–260.
[25] BAYKASOGLU A, KARASLAN F S. Solving comprehensive dynamic job shop scheduling problem by using a grasp-based approach. International Journal of Production Research, 2017, 55(11): 3308–3325.
[26] DA SILVA N C O, SCARPIN C T, PECORA JR J E, et al. Online single machine scheduling with setup times depending on the jobs sequence. Computers & Industrial Engineering, 2019, 129: 251–258.
[27] WANG J J, ZHU X G, ZHU J H, et al. A realtime scheduling algorithm for multiple earth observation satellites. Proc. of the 9th IEEE International Conference on Embedded Software and Systems, 2012, 673–680.
[28] GAO K, WU G H, ZHU J H. Multi-satellite observation scheduling based on a hybrid ant colony optimization. Advanced Materials Research, 2013, 765: 532–536.
[29] SHI Y L, JIANG X J, ZHANG Y F, et al. Static routing design of solar synchronous orbit micro-nano satellite constellation. Electronic Design Engineering, 2018, 25(17): 25–29. (in Chinese)
[30] WU G H, WANG H L, PEDRYCZ W, et al. Satellite observation scheduling with a novel adaptive simulated annealing algorithm and a dynamic task clustering strategy. Computers & Industrial Engineering, 2017, 113: 576–588.
[31] YAO F, LI J T, CHEN Y N, et al. Task allocation strategies for cooperative task planning of multi-autonomous satellite constellation. Advances in Space Research, 2019, 63(2): 1073–1084.
[32] KLEINSCHRODT A, NOGUEIRA T, REED N, et al. Mission planning for the TIM nanosatellite remote sensing constellation. Proc. of the 69th International Astronautical Congress, 2018. DOI: 10.1007/978-94-011-5088-0_37.
[33] XU R, CHEN H P, LIANG X L, et al. Priority-based constructive algorithms for scheduling agile earth observation satellites with total priority maximization. Expert Systems with Applications, 2016, 51: 195–206.
[34] FENG P, CHEN H, PENG S, et al. A method of distributed multi-satellite mission scheduling based on improved contract net protocol. Proc. of the 11th International Conference on Natural Computation, 2015, 1062–1068.
[35] SKOBELEV P O, SIMONOVA E V, ZHILYAEV A A, et al. Application of multi-agent technology in the scheduling system of swarm of earth remote sensing satellites. Procedia Computer Science, 2017, 103: 396–402.
[36] HE L, DE WEERDT M, YORKE-SMITH N. Tabu-based large neighbourhood search for time/sequence-dependent scheduling problems with time windows. Proc. of the 29th International Conference on Automated Planning and Scheduling, 2019: 186−194.
[37] HE L, DE WEERDT M, YORKE-SMITH N. Time/sequence-dependent scheduling: the design and evaluation of a general purpose tabu-based adaptive large neighbourhood search algorithm. Journal of Intelligent Manufacturing, 2020, 31: 1051–1078.
[38] HE L, LIU X L, LAPORTE G, et al. An improved adaptive large neighborhood search algorithm for multiple agile satellites scheduling. Computers and Operations Research, 2018, 100(1): 12–25.
[39] HSIEH F S. Analysis of contract net in multi-agent systems. Automatica, 2006, 42(5): 733–740.
[40] DE NIJS F, SPAAN M, DE WEERDT M. Preallocation and

planning under stochastic resource constraints. Proc. of the 32nd AAAI Conference on Artificial Intelligence, 2018, 4662–4669.
[41] PAUKSHTIS E A. Constrained Markov decision processes. Boca Raton: CRC Press, 1999.
[42] IBM. IBM CPLEX Optimizer, 2018. https://www.ibm.com/analytics/cplex-optimizer.
[43] LI G L, XING L N, CHEN Y W. A hybrid online scheduling mechanism with revision and progressive techniques for autonomous earth observation satellite. Acta Astronautica, 2017, 140(1): 308–321.

Biographies

WEN Jun was born in 1997. She received her B.S. degree in information and computing science from the School of Mathematics and Statistics, Changsha University of Science and Technology (CSUST), China, in 2019. She is currently a master student in management science and engineering at the College of Systems Engineering, National University of Defense Technology (NUDT), China. Her research interests are artificial intelligence, deep reinforcement learning, and metaheuristics, mainly focusing on distribution management and satellite scheduling problems such as collaborative mission planning methods of multi-satellite mobile target tracking.
E-mail: jun_wen@aliyun.com

LIU Xiaolu was born in 1985. She received her B.E. degree in system engineering from National University of Defense Technology (NUDT), China, in 2006, and Ph.D. degree in management science and engineering from NUDT in 2011. From October 2015 to October 2016, she was a visiting scholar in École des Hautes Études commerciales de Montréal (HEC), Montreal, Canada. She is currently an associate professor at the College of Systems Engineering, NUDT. Her research insterests are artificial intelligence and metaheuristics, focusing on distribution management and satellite scheduling problems.
E-mail: lxl_sunny@nudt.edu.cn

HE Lei was born in 1991. He received his B.E. degree in management engineering from National University of Defense Technology (NUDT), China, in 2014, and Ph.D. degree in management science and engineering from NUDT in 2019. From October 2017 to October 2019, he was a visiting Ph.D. student in Delft University of Technology, the Netherlands. He is currently a lecturer at the College of Systems Engineering, NUDT. His research interests include artificial intelligence and machine learning. His current research focuses on the optimization of scheduling and planning of Earth observation satellites missions by designing exact and metaheuristic algorithms.
E-mail: helei@nudt.edu.cn