Song et al. EURASIP Journal on Image and Video Processing (2018) 2018:29
https://doi.org/10.1186/s13640-018-0268-8

EURASIP Journal on Image
and Video Processing

RESEARCH Open Access

An emergency task autonomous planning method of agile imaging satellite

Yanjie Song[1*], Danjie Huang[2], Ziyu Zhou[3] and Yingwu Chen[1]

Abstract

As the number of satellite emergency imaging tasks grows, the main goal of satellites becomes putting forward solutions and meeting users' demands in a relatively short time. This study aims to investigate the problem of emergency task planning for agile satellites. Through the analysis of the problem and its constraints, a model of emergency task autonomous planning was implemented. According to the characteristics of emergency tasks, a strategy to deal with the tasks of different emergency levels and various quantities was proposed. We put forward three algorithms for quick insertion of emergency tasks, i.e., emergency task insertion algorithm (ETIA), general emergency task insertion algorithm (GETIA), and general emergency task planning &insertion algorithm (GETPIA). The experimental result showed that the strategy and algorithms can not only respond quickly to observation tasks but also produce effective planning programs to ensure the successful completion of observation tasks.

Keywords: Emergency task, Autonomous planning, Agile imaging satellite, Replanning

1 Introduction

As the satellite imaging technology develops in recent years, people have become increasingly dependent on it. For example, China has its own commercial remote sensing satellite constellation, SuperView-1. SuperView-1 consists of four optical satellites with the resolution of 0.5 m. It will provide customers with remote sensing data services for multiple fields. Figure 1 is a picture taken by it. There are a large amount of images that need to be completed every day. With the emergence of increasingly strict and meticulous requirements of users as well as some temporary demands, the traditional satellite management and control modes that are highly dependent on the commands from the ground have experienced difficulties in meeting the needs of future development of space technology [1, 2]. Such situation is reflected in the patterns of the traditional model. Due to the limitation of communication time between ground stations and satellites, it is very difficult for satellites to respond in time to the temporary observation demands of users or to obtain information about some emergency events, resulting in a lag in task completion [3]. Even if the tasks could be completed in time, the imaging quality would be difficult to be guaranteed. Therefore, the most effective way to reduce the response time of satellites as well as complete all kinds of emergency tasks of high quality without delay is to rely on the satellites to make the task plans and propose solutions through autonomous reprogramming.

The satellite image acquisition is an important scheduling problem in satellite mission planning [4]. Replanning is especially critical as the number of imaging tasks increases. The autonomous replanning can effectively solve the problem of limited communication time between ground and satellite, so as to improve the utilization efficiency of satellite resources [5, 6]. The CASPER (Continuous Activity Scheduling Planning Execution and Re-planning) of NASA has successfully made the E0-1 satellite reprogram online by using a local search algorithm for iterative repair [7]. The FireBIRD of DLR (German Aerospace Agency) has carried a VAMOS (Autonomous Task Planning On-board a Spacecraft) platform, which uses real-time constraint checking to reprogram tasks and adjust the follow-up program in time [8]. The AGATA (Autonomy Generic Architecture-Test and Application) platform of CNES has adopted the task triggering mechanism, which can respond quickly to the tasks [9]. Baek et al. [10] have proposed a set of visualized satellite

* Correspondence: songyj_2017@163.com
[1]College of Systems Engineering, National University of Defense Technology, Changsha 410073, Hunan, China
Full list of author information is available at the end of the article

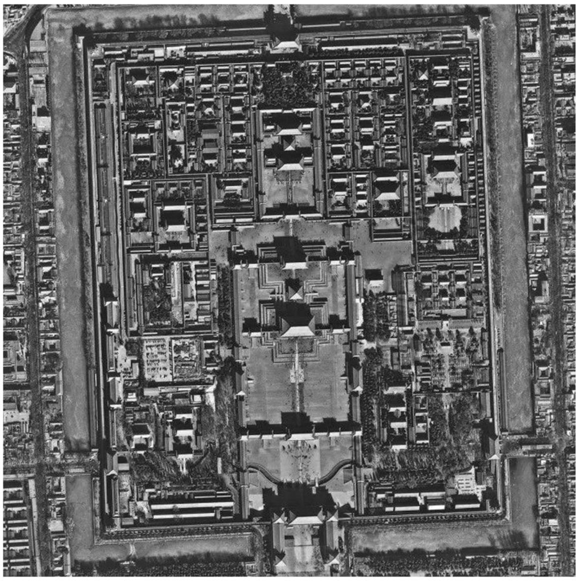

Fig. 1 Picture taken by SuperView-1

autonomous task planning system and demonstrated the feasibility of the system, but the efficiency of the system has not been discussed. Chu et al. [11] have used branch demarcation to effectively solve the real-time scheduling problem of the dual agile satellite system, and the autonomous scheduling method of clustering is likely to be a research direction in the future. Some other studies have analyzed the online scheduling problem from the theoretical level, developed the scheduling framework, and proposed corresponding methods [12, 13]. These studies have taken into account the online scheduling of routine observation tasks, but they paid less attention to the reprogramming of emergency tasks [14, 15]. Additionally, the real-time dynamic scheduling is more often done offline, so it is difficult to respond to emergency tasks [16].

The current study has little research on the replanning and completion process of emergency tasks and lacks a method for generalizing problems. Based on previous studies, we proposed a generalized model for handling emergency task replanning problem.

This paper will discuss and analyze the difficulties in the emergency observation tasks as well as put forward the task planning model of the emergency tasks based on various constraints. In the end, the paper will put forward a combined strategy to cope with several kinds of tasks and design the corresponding algorithm.

The rest of the article is structured as follows. The second part will propose the model of the emergency task planning and specify the corresponding restriction conditions. Then, the strategies and algorithms to solve this problem will be presented in the third part. In the fourth part, the feasibility and effectiveness of the proposed method will be verified by an example. Finally, the conclusion is given in the fifth part.

2 Problem description and model

The observation demands of emergency tasks refer to a series of demands that the imaging satellite needs to complete the observing tasks due to the temporary needs of users or some emergencies. Compared with the general tasks, the emergency tasks have a higher priority to be completed. However, the insertion of emergency tasks inevitably conflicts with the original task plan; thus, the problem arises.

2.1 Problem description

We translated the agile satellite task planning problem into mathematical models so as to design algorithms and solve them. In the following, we described the problem as well as proposed models and constraints based on assumptions.

The problem can be described as follows: A set of emergency tasks requires observation by the agile satellites, but their urgency levels may be different in urgency. The task reprogramming can efficiently accomplish the observation task without intervention of the ground. In this way, the satellite can not only respond quickly to emergency demands, but also perform as many emergency tasks as possible, providing maximum observation profits.

The difficulties in this question are as follows:

1. Generally, there are numerous constraints in satellite task planning, so the reprogramming makes the tasks more difficult.
2. The task time window and imaging time window exist at the same time, so the new problem is a lot more complicated than the general scheduling problem.
3. The operation ability of agile satellite is limited, and the planning algorithms need to be adapted to the resources on the satellite.
4. Emergency task reprogramming requires high efficiency in the process of reprogramming.
5. The emergency tasks are different in the urgency levels and inconsistent with the requirements of the system.

In this article, we analyzed the previous research and proposed a model based on the following assumptions:

Assumption 1: The original planning is the best planning, and emergency replanning could lower the profit.

Assumption 2: The emergency task can be completed through an observation, and there is only one observation opportunity.
Assumption 3: The satellite can communicate with the ground and other satellites (relay satellites) in time and receive the pretreatment results.
Assumption 4: The replanning system can communicate effectively with other systems on the satellite and execute the corresponding commands timely.
Assumption 5: The higher priority the emergency task is, the faster the system accomplishes the task.
Assumption 6: The model only focuses on the satellite imaging planning and observation process.

2.2 Model input

I represents the set of original tasks, J represents the set of emergency tasks, K represents the set of tasks that have been excluded from the original task plan; S_i and E_i represent the start time and completion time of the task i in the original plan, WS_i and WE_i represent the task start time window and end time window of the ith item, S'_i and WE'_j represent the start time window and end time window of task j, T'_j represents the duration of task j, w_j represents the task Priority of the task j, $T_{i\to j}$ represents the attitude maneuver time from the ith original task to the jth emergency task, mi represents the storage consumption of the task i, M represents the maximum storage capacity that the satellite can use for replanning, e_i represents the power consumption of task i, E represents the maximum power that the satellite can be used for replanning, and σ represents the longest time required for attitude maneuver of the satellite.

2.3 Model output

x_j and y_k represent the decision variables in the model, where X_j indicates that the emergency task j is selected; otherwise, it equals 0. y_k indicates that the task k is deleted; otherwise, it equals to 0. S'_j and E'_j represent the start time and end time of the emergency task j, SS represents the start time of the replanning, SE represents the end time of the replanning, and IG represents the time spent on the generation of commands.

The objective function is designed to realize the maximization of the total profit of replanning tasks. In the model, $f_i(x)$ refers to the profit in the original task, $f_j(x)$ is the profit of emergency task, $l_k(x)$ refers to the loss value from the cancel task of the original plan, i, j, and k refers to the original task sequence, emergency task sequence, and task sequence canceled correspondingly. The model is as follows:

$$\max\left[\sum_{i=1}^{I} f_i(i) + \sum_{j=1}^{J} f_j(j) x_j - \sum_{k=1}^{K} l_k(k) y_k\right] \tag{1}$$

2.4 Constraints

1. Each of the emergency tasks can be executed once at most.

$$x_j \le 1 \tag{2}$$

2. The total amount of time spent on replanning and command generation should not exceed the start time window of emergency task j.

$$SE + IG \le WS'_j \tag{3}$$

3. The total storage consumption of all tasks should not exceed the fixed capacity limit that satellites can use for replanning.

$$\sum_{i\in I} m_i + \sum_{j\in J} m_j - \sum_{k\in K} m_k \le M \tag{4}$$

4. The total power consumption of all tasks should not exceed the amount of power the satellite can be used in replanning.

$$\sum_{i\in I} e_i + \sum_{j\in J} e_j - \sum_{k\in K} e_k \le E \tag{5}$$

5. There must be sufficient time from original task i to emergency task j.

$$E_i + T_{i\to j} \le S'_j \tag{6}$$

6. The time window of emergency task j is not overlapped with the time window of original task i:

$$[S_i, E_i] \cap \left[S'_j, \mathrm{E}'_j\right] = \varnothing \tag{7}$$

7. Each of the emergency tasks should be executed within the time window allowed respectively:

$$WS'_j \leq S'_j < E'_j \leq WE'_j \tag{8}$$

8. There is no overlapped time among emergency tasks:

$$\left[S'_{j_1}, E'_{j_1}\right] \cap \left[S'_{j_2}, E'_{j_2}\right] \cap \ldots \cap \left[S'_{j_n}, E'_{j_n}\right] = \varnothing, \quad j_1, j_2, \ldots, j_n \in J \tag{9}$$

These constraints make the problem optimization more complicated. To solve this problem effectively, it is very important to put forward a set of efficient solutions. Next, we started from the characteristics of emergency tasks and considered the solutions of emergency task reprogramming in different circumstances.

The importance of the emergency task is regarded as the weight, which is related to the priority of the task. The mathematical expressions are as follows:

$$w_i = \begin{cases} 1.5 & 5 \leq \text{priority} \\ 1.3 & 3 \leq \text{priority} < 5 \\ 1.0 & 0 \leq \text{priority} < 3 \end{cases} \tag{10}$$

It is obvious that the higher the task priority, the greater impact on the final profit of the function. Above all, we have described the model and the constraints completely. In the next chapter, we designed a set of algorithms to solve this problem.

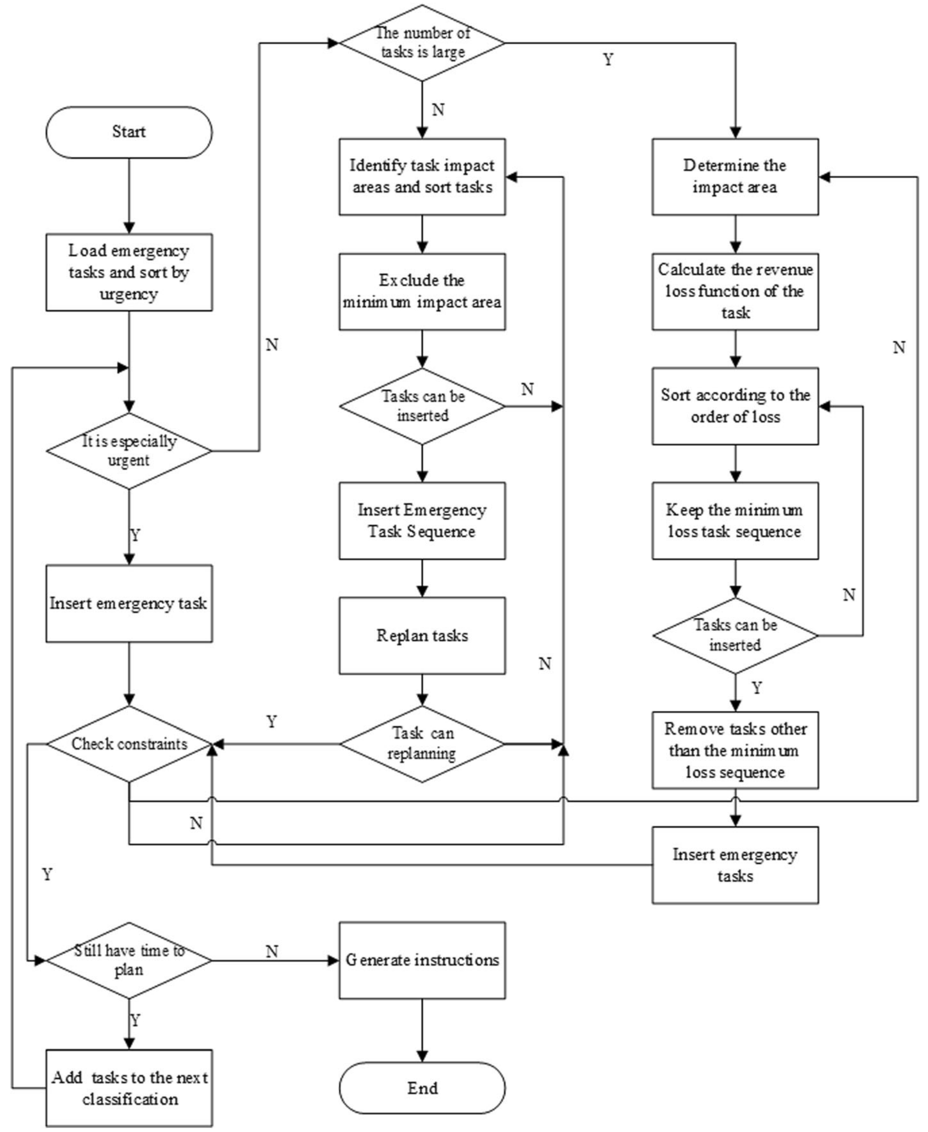
The number of tasks is large
Y
N
Start
Identify task impact areas and sort tasks
Determine the impact area
Load emergency tasks and sort by urgency
Exclude the minimum impact area
Calculate the revenue loss function of the task
N
N
Tasks can be inserted
N
Sort according to the order of loss
It is especially urgent
Y
Insert Emergency Task Sequence
Keep the minimum loss task sequence
Insert emergency task
Replan tasks
N
N
Tasks can be inserted
Y
Y
Check constraints
Task can replanning
Remove tasks other than the minimum loss sequence
N
Y
Insert emergency tasks
Still have time to plan
N
Generate instructions
Y
Add tasks to the next classification
End

Fig. 2 The overall flow of the algorithm

3 Methods

Due to the limited computing resources on the satellite, the design of the algorithm must be compatible with the resources [17], requiring the program to be as simple and small as possible. At the same time, the replanning of emergency task requires high efficiency. In this paper, we proposed a set of emergency task planning algorithms for combinatorial strategy selection. Through the algorithms, we classified different tasks in the emergency task sequence and ensured the high quality and efficiency of solution. The algorithms were designed into three kinds respectively: the emergency tasks with high priority, general emergency tasks with fewer tasks, and general emergency tasks with more tasks.

3.1 The overall process of the algorithm

Step 1: Tasks classification. Classify the tasks in accordance with the priority and urgency degrees, and each priority consists of a kind of ambiguous priority, giving priority to the first priority order and calculating the coverage of emergency task time windows. If time window overlapped with that of the task of the first priority exceeds the threshold, then analyze the combination of the two time windows;

Step 2: Strategy selection. If it is an emergency tasks with high priority, use (emergency task insertion algorithm) ETIA. If it is the general emergency tasks with fewer tasks, use general emergency task insertion algorithm (GETIA). If it is the general emergency tasks with more tasks, use general emergency task planning &insertion algorithm (GETPIA);

Step 3: Constraint checking. Check the results of the programming generated by the respective algorithms and check whether they satisfy the constraints. If they fail to pass, try the program with the second total profit. If they still fail to pass, try the third one, so on and so forth;

Step 4: Try more emergency tasks. Check if there is still time, storage, and remaining power to join more emergency tasks, and if so, return to the second step and add the next priority emergency task; if there is no time, storage or remaining power, then directly carry out the fifth step;

Step 5: Profit calculation. Calculate the profit of each part and finally determine the overall benefit using the objective function. The overall flow chart of the algorithm is shown in Fig. 2.

The whole algorithms contain three sub-algorithms. The three sub-algorithms are mainly designed according to the urgency of the emergency tasks and the number of emergency tasks in the same category. First of all, we designed the emergency task insertion algorithm for emergency tasks with high priority.

Table 1 Emergency task insertion algorithm

```
Algorithm I    Emergency task insertion algorithm
Start Procedure
Initialize position percentage $\rho$, original task number $i$
emergency task number $j$
for each emergency task do
    Determine the search starting point $\boldsymbol{i}$ according $\boldsymbol{\rho}$
    Search backward from the beginning task $\boldsymbol{i}$
    Delete task $\boldsymbol{i}$
    if $(E_i\text{-}S_i)>T_j+\sigma$ then
        Insert emergency task
        loop for
    end if
    $i=i+1$
end for
End Procedure
```

The emergency task insertion algorithm (ETIA) mainly solves the problem of the emergency tasks with high priority, and the pseudo-code of the algorithm is shown in Table 1. The basic idea is rapidly positioning correspondingly through the location of

Table 2 General emergency tasks insertion algorithm

```
Algorithm II    General emergency tasks insertion algorithm
Start Procedure
Initialize the sets $w$, $w_1$, $w_2$, $w_3=\Phi$, $mintime=0$
original task index $i$, emergency task indexr $j$,
the earliest start time $ES$, the latest end time $LE$,
Task time $T_j$
for each level of emergency tasks do
    $midtime=(ES+LE)/2$
    Search from midtime to both sides, form $w_1$, $w_2$
    Form an ascending sequence $w$ according to the
    income level of $w_1$, $w_2$
    for index $i$ in w do
        $mintime=mintime+w[i]$
        Delele $w[i]$
        if $mintime>T_j+\sigma$ then
            loop for
        end if
    $i=i+1$
    end for
    Insert emergency tasks
end for
Task Replanning
End Procedure
```

the percentage and then searching backward, until there is enough time to insert the emergency tasks. Insert the next task after a task is inserted. Since there are no many high-priority emergency tasks, we could insert the tasks one by one (Table 2).

In general emergency tasks insertion algorithm (GETIA), the processing of tasks is relatively more complicated than that of high-priority emergency tasks. The algorithm is applied with the idea of Greedy Algorithm [18–20]. For the original task plan within the time window of emergency tasks, the task sequence should be sorted according to the profit sequence. The task elimination should be performed in the direction of the maximum return of the new task sequence. Therefore, the task with the lowest return is supposed to be removed firstly (Table 2).

Figure 3 showed the process from considering the inserted position to eliminating the conflicting tasks in the original task plan. Firstly, we determined the scope of tasks that may need to be eliminated to resolve the task conflict according to the time window of the emergency task. After comparing the profit of the tasks, we selected one or more tasks with the lowest level of profit, then replaced it with the emergency task (Table 3).

The general emergency task planning &insertion algorithm (GETPIA) uses the idea of Pareto optimization [21–24], which requires that any improvement needed should not damage the profit of other parts. For each task j added to the task sequence, we suppose that j dominates i, which is written as $j \prec i$, subject to $f(j) \leq f(i)$. Pareto solving method can not only effectively reduce the number of arithmetic operations, but also complete the emergency task insertion and task re-planning in one algorithm flow at the same time with the resource constraints of autonomous programming on the satellite (Table 3).

Figure 4 described the process of replanning the task plan after choosing the location of emergency task in Algorithm III. The data from $l_1$to l_5 represent the loss value of the five task execution paths. In the process of selecting the path of the algorithm, assume that l_3 is the maximum loss value, then traverse all the situations from emergency task with high priority to task two. Since l_3 is larger than l_5, it is impossible to execute task 1 and then task 2 after executing the emergency task with high priority according to the Pareto principle because l_3 has been removed from the operation. This will help to improve the operation efficiency of the algorithm.

4 Experimental analysis

The proposed algorithms are implemented by Matlab2016b on a laptop with Core I5-3337U 1.8 GHz CPU, 4 GB memory, and Windows 8.1 operating system.

This part of the experiment was designed to verify the feasibility of the algorithm. Since agile satellite task planning has no baseline, so we did the simulation

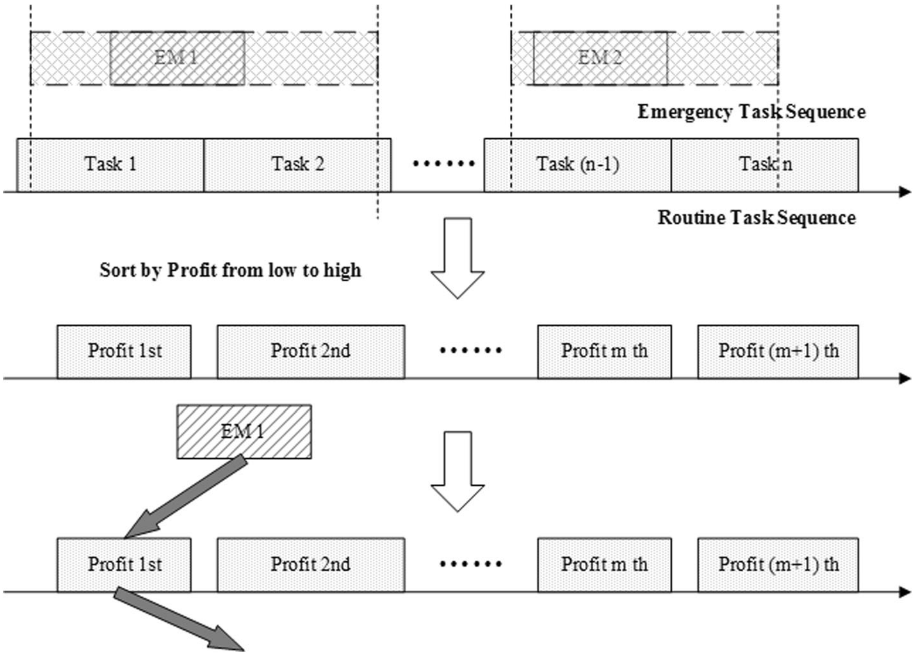

Fig. 3 The diagram of task insert selection

Table 3 General emergency task planning & insertion algorithm

```
Algorithm III   General emergency task planning & insertion algorithm
Start Procedure
Initialize the sets EM←Emergency Tasks
the earliest start time ES, the latest end time LE,
loss[i][j]←loss of task i to task j, losspath[]={},
mintime=0,
Generate a loss matrix D[]
if loss[i][l] or loss[l][k] > loss[i][j]
  Put the path number in tabu list
end if
if D^n< >D^n-1 then
  for i in D and not in tabu list do
    for j in D and not in tabu list do
      if loss[i][l]+loss[l][j]<loss[i][j] then
        D[i][j]=loss[i][l]+loss[l][j]
      end if
    end for
  end for
end if
Sort the loss values in ascending order→ L[i]
Get time spent on each run→ Time[i]
for index i in L do
  if mintime>LE-ES+σ then
    Select loss path i
    loop for
  end if
end for
Insert emergency tasks in the corresponding position
End Procedure
```

experiments with scenario simulation and data under agile satellite observation constraints generated randomly. We analyzed generalization issues and did not specify the type of satellites.

The main points of simulation experiments are as follows:

1. The satellite task planning scenarios have been planned;
2. A series of emergency tasks with different priorities were randomly generated according to the task requirements, and the time windows of these tasks may overlap with those of each other.

We designed experiments to verify the three sub-algorithms in the emergency task planning algorithms for combinatorial strategy selection and prove the overall feasibility of the algorithm respectively.

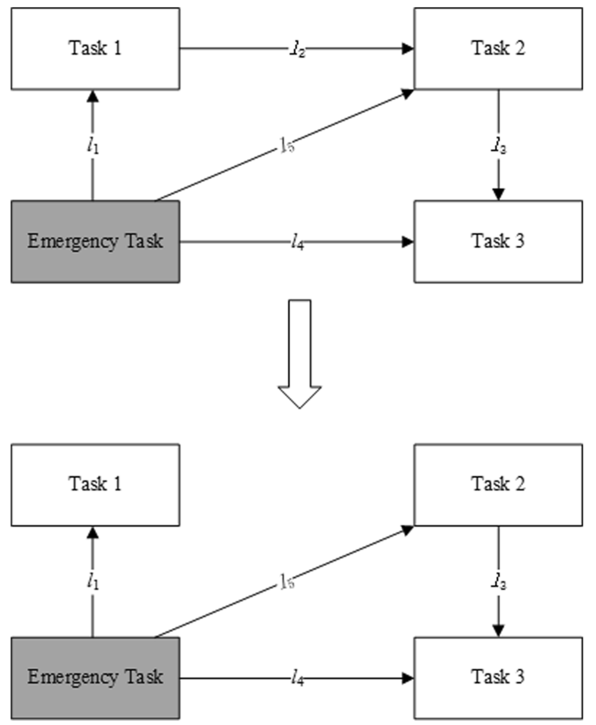

Fig. 4 The diagram of task replanning

5 Results and discussion

Experiment 1 was conducted to verify ETIA and the result is shown in Table 4.

In the experiment 1, we simulated different original tasks and considered different positions of emergency tasks in the original task planning sequence. Besides, we compared ETIA with the traditional emergency insertion algorithm. Taking the calculation efficiency of satellite into consideration, we assumed that the calculation time of satellite was 500 times of the simulation experiment. The experimental results showed that the average running time of ETIA was less than that of the traditional insertion algorithm. As the number of original task plans increased, the completion time of the algorithm tended to remain stable. However, ETIA is not ideal for inserting multiple emergency tasks, partly because of its greater volatility in profit, and on the other hand, inserting tasks one by one is inefficient for multiple tasks. When the task scale was 100, the emergency insertion algorithm could save 10.306 s than the conventional one. Therefore, we concluded that Algorithm I could

Table 4 Comparison of ETI and RP

Scale	ETIA(s)	RPA(s)
100	2.803	13.109
200	14.14	16.938
300	14.523	17.481

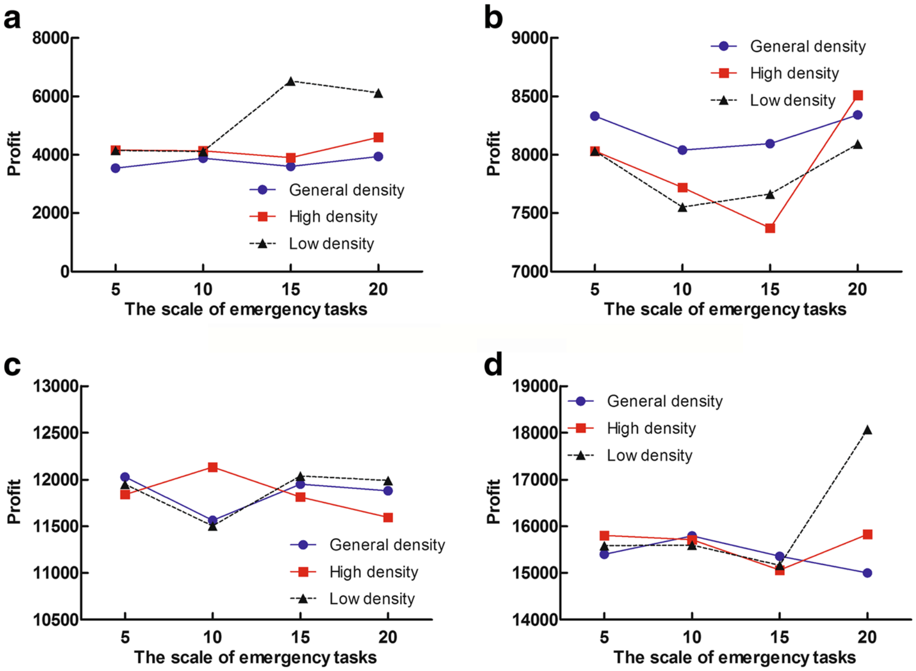

Fig. 5 The profit of GETIA (**a**, **b**, **c**, **d**)

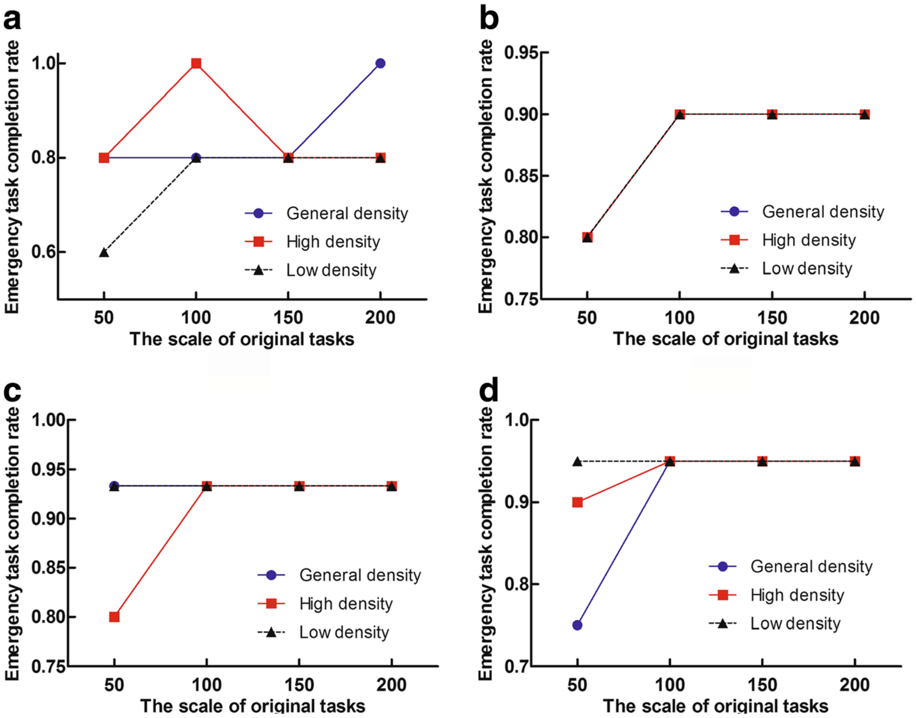

Fig. 6 The completion of emergency tasks (**a**, **b**, **c**, **d**)

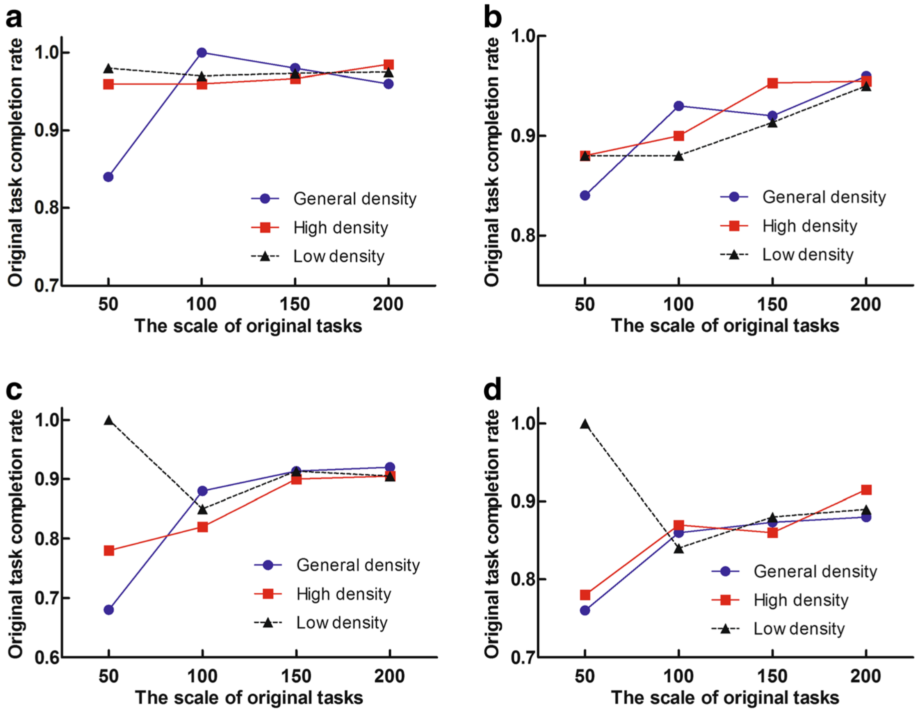

Fig. 7 The completion of original tasks (**a**, **b**, **c**, **d**)

effectively solve the problem of reprogramming a few emergency tasks with high priority.

In experiment 2, we discussed the GETIA. Firstly, we considered the scale of different tasks and the number of different emergency tasks; chose the task scales of 100, 150, and 200 for analysis; and selected 5, 10, 15, and 20 emergency tasks, respectively, and we considered the three types of high density, low density, and high density for each emergency task. Figure 5 represents the original task scale of 50, 100, 150, and 200 from Fig. 5a–d, respectively. In Fig. 5a, the GETIA has a good effect on low-density emergency tasks when the original task scale is 50. When the task scale rises to 100, the GETIA performs well at the general density, which is shown in Fig. 5b. In Fig. 5c, after the task scale reaches 150, the GETIA has higher profit when the scale of emergency task with high density is 5 and 10, then the profit shows a significant drop, which may be due to the excessive number of original tasks deleted. When the scale of original tasks is 200 and the number of emergency tasks with low density is 20, the profit of GETIA with high density is much higher than that of general density and low density.

Another experiment was taken. We respectively compared the completion of 5, 10, 15, and 20 emergency tasks as the original task scale increases, and the result is shown in Fig. 6. Figure 6a–d shows cases when the number of emergency tasks is 5, 10, 15, and 20,

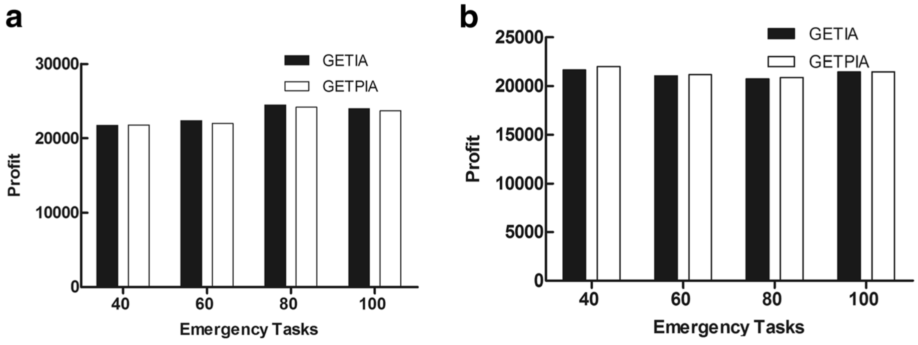

Fig. 8 Revenue comparison under 200 original tasks (**a**, **b**)

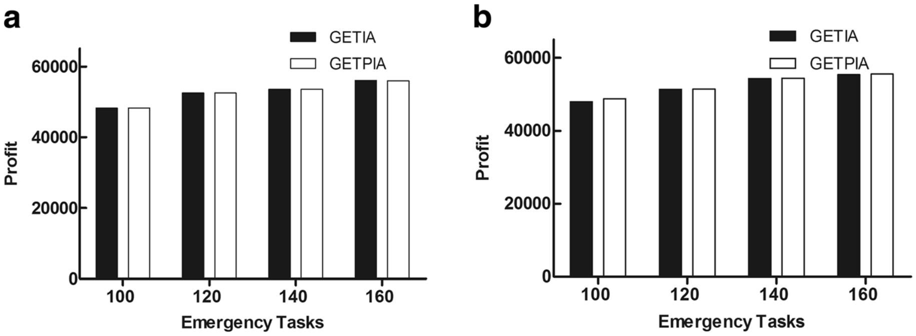

Fig. 9 Revenue comparison under 300 original tasks (**a**, **b**)

respectively. When the scale of emergency tasks is small, the completion rate of tasks is greatly affected by the distribution density of tasks. When the numbers of emergency tasks and the original task continue to increase, the completion rate of emergency tasks at different densities shows a high degree of consistency. The result also indicates that the GETIA can complete the emergency tasks well.

Subsequently, we analyzed the influence of GETIA on the original task. As shown in Fig. 7, we compared the completion rate of the original tasks under different task scales and different emergency task densities. Figure 7a–d represents the case of 5, 10, 15, and 20 emergency tasks, respectively. Figure 7 shows that the influence of GETIA on the original task has great uncertainty. In the best case, the original task sequence can continue implementing the original planning scheme. In the worst case, the original task can be only completed 68%, that is to say, the task is influenced greatly. The result also explains why great changes happen in the returns of the new task sequence above.

We also conducted a comparison experiment between GETIA and GETPIA. The experiment was based on 200 original tasks, and the emergency task scales were set as 40, 60, 80, and 100, respectively, taking both long-term and short-term emergency tasks into account. The result is shown in Fig. 8. Figure 8a indicates that when dealing with a short-term emergency task, the profit of GETPIA is lower than that of GETIA. However, in Fig. 8b, GETPIA performs better than GETIA when the emergency task takes a long time. The reason is that the GETPIA is more concerned with the loss profit of the task when the number of tasks changes, compared with GETIA. When the number of emergency tasks is large and the duration is longer, the addition of an emergency task may result in the deletion of tasks in multiple original task sequences, resulting in a decrease in profit.

We increased the scale of the original tasks to 300 and the number of emergency tasks to 100, 120, 140, and 160 respectively. The results are shown in Fig. 9. Figure 9a shows a short-term emergency task and Fig.9b indicates a long-term emergency task. Dealing with 200 original tasks, GETPIA has higher profit than GETIA for long-term emergency tasks. As the scale of the original task increases, the gap between two algorithms also increases.

6 Conclusions

In this research, a model was established to solve the problem of autonomous programming for the emergency missions of agile imaging satellites. We proposed a combination of three strategies for the algorithm based on the number and characteristics of emergency missions. After that, those three strategies in the algorithm were verified by three experiments. We find that the algorithm could meet the requirements of autonomous programming for a satellite as a whole. The three sub-algorithms had stringent requirements for the characteristics of emergency missions, and the quality of solution to the algorithm would be declined when it was beyond their scopes of application. ETIA has high computational efficiency. GETIA has a good performance in solving small-scale and short-term emergency missions. GETPIA is more suitable for large-scale missions and when the duration of emergency missions is longer. The conclusion demonstrates that these three algorithms can solve the problem of autonomous replanning for emergency missions on the satellite. As a result, a more advanced study of satellite autonomous mission programming is necessary, and some learning methods can be used to enhance the forecasting ability of satellites.

Abbreviations

AGATA: Autonomy Generic Architecture-Test and Application; CASPER: Continuous Activity Scheduling Planning Execution and Replanning; DLR: German Aerospace Agency; ETIA: Emergency task insertion algorithm; GETIA: General emergency tasks insertion algorithm; GETPIA: General emergency task planning & insertion algorithm; RPA: Routine Planning Algorithm; VAMOS: Autonomous Task Planning Onboard a Spacecraft

Funding
This research was supported by the National Natural Science Foundation of China (Grant numbers: 71690233, 61473301, 71701203).

Availability of data and materials
Due to legal constraints the data for the research is not publicly available upon publication. The data can be requested from the corresponding author on reasonable request.

Authors' contributions
JYS and JDH, YZZ, and WYC designed the research. JYS performed the research. JYS wrote the paper. JDH adjusted the format. YZZ processed data. WYC guided the paper's idea. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Author details
[1]College of Systems Engineering, National University of Defense Technology, Changsha 410073, Hunan, China. [2]School of Science and Engineering & School of Management and Economics, Chinese University of Hong Kong, Shenzhen 518172, Guangdong, China. [3]Department of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, 117 Transportation Buld., 104 S. Mathews Ave., MC-238, Urbana, IL 61801-3080, United States.

Received: 5 March 2018 Accepted: 13 April 2018
Published online: 29 April 2018

References

1. M LemaÎTre, G Verfaillie, F Jouhaud, JM Lachiver, N Bataille, Selecting and scheduling observations of agile satellites. Aerosp. Sci. Technol. **6**(5), 367–381 (2002).
2. Rabideau, G., Tran, D., Chien, S., Cichy, B., Shenvood, R., & Mandl, D., et al. (2006). Mission Operations of Earth Observing-1 with Onboard Autonomy. Proceedings of IEEE International Conference on Space Mission Challenges for Information Technology, 2006. Smc-It (pp.7 pp.–373). IEEE.
3. ZY Lian, YJ Tan, ZZ Yan, Temporal reasoning technology for aeos scheduling. Syst. Eng. Electron. 35(6), 1206–1211 (2013).
4. Wolfe, W. J., & Sorensen, S. E. Three scheduling algorithms applied to the earth observing systems domain. Management Science. **46**(1), 148-166 (2000).
5. Y Min, Z Min, Small satellite task autonomous scheduling design method based on fuzzy neural network. J. Astronaut. **28**(2), 385–291 (2007).
6. JC Agn, N Bataille, D Blumstein, E Bensana, G Verfaillie, Exact and approximate methods for the daily management of an earth observation satellite. RAIRO – Oper. Res. **41**(4), 381–398 (2007).
7. S Chien, R Knight, A Stechert, R Sherwood, G Rabideau, Using Iterative Repair to Improve the Responsiveness of Planning and Scheduling, International Conference on Artificial Intelligence Planning Systems (AAAI Press, 2000), pp. 300–307.
8. B Wille, MT Wörle, C Lenzen, Vamos—verification of autonomous mission planning on-board a spacecraft. IFAC Proc. Vol. **46**(19), 382–387 (2013).
9. MA Gleyzes, L Perret, P Kubik, Pleiades system architecture and main performances. ISPRS – Int. Arch. Photogramm. Remote Sens. Spati. Inf. Sci. XXXIX-B1, 537–542 (2012).
10. SW Baek, SM Han, KR Cho, DW Lee, JS Yang, PM Bainum, et al., Development of a scheduling algorithm and GUI for autonomous satellite missions. Acta Astronaut. **68**(7–8), 1396–1402 (2011).
11. X Chu, Y Chen, Y Tan, An anytime branch and bound algorithm for agile earth observation satellite onboard scheduling. Adv. Space Res. 2017(1), 1–15 (2017).
12. Augenstein, S., Mann, J. M., & Berkenstock, D. (2016). Satellite Scheduling System.
13. B Sun, L Mao, W Wang, X Xie, Q Qin, Satellite mission scheduling algorithm based on GA. Proc. SPIE 6795, 30 (2007).
14. Maillard, A., Verfaillie, G., Pralet, C., Jaubert, J., Fontanari, F., & Sebbag, I. (2015). Postponing decision-making to deal with resource uncertainty on earth-observation satellites. International Workshop on Planning and Scheduling for Space..
15. C. Pemberton A, J & G. Greenwald B, L. On the need for dynamic scheduling of imaging satellites. 34. (2012)
16. Q Dishan, H Chuan, L Jin, M Manhao, A dynamic scheduling method of earth-observing satellites by employing rolling horizon strategy. ScientificWorldJournal 2013(3), 304047 (2013).
17. B Song, Y Li, Y Chen, F Yao, Y Chen, A repair-based approach for stochastic quadratic multiple knapsack problem. Knowl.-Based, 145, 145–155 (2018).
18. P Wojtaszczyk, Greedy algorithm for general biorthogonal systems. J. Approx. Theory **107**(2), 293–314 (2000).
19. C Kahraman, O Engin, İ Kaya, RE Öztürk, Multiprocessor task scheduling in multistage hybrid flow-shops: a parallel greedy algorithm approach. Appl. Soft Comput. **10**(4), 1293–1300 (2010).
20. HE Romeijn, DR Morales, A class of greedy algorithms for the generalized assignment problem. Discret. Appl. Math. **103**(1), 209–235 (2000).
21. H Li, Q Zhang, Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. IEEE Trans. Evol. Comput. **13**(2), 284–302 (2009).
22. MP Andersson, T Bligaard, A Kustov, KE Larsen, J Greeley, T Johannessen, et al., Toward computational screening in heterogeneous catalysis: pareto-optimal methanation catalysts. J. Catal. **239**(2), 501–506 (2006).
23. D Lei, A pareto archive particle swarm optimization for multi-objective job shop scheduling. Comput. Ind. Eng. **54**(4), 960–971 (2008).
24. PK Shukla, K Deb, On finding multiple pareto-optimal solutions using classical and evolutionary generating methods. Eur. J. Oper. Res. **181**(3), 1630–1652 (2007).