Complex & Intelligent Systems (2024) 10:1483–1499
https://doi.org/10.1007/s40747-023-01227-9

ORIGINAL ARTICLE

Dynamic scheduling method for data relay satellite networks considering hybrid system disturbances

Zongling Li[1,2] · Xinjiang Chen[3] · Qizhang Luo[4] · Guohua Wu[4] · Ling Wang[5]

Received: 8 January 2023 / Accepted: 18 August 2023 / Published online: 14 September 2023

Abstract

System disturbances, such as the change of required service durations, the failure of resources, and temporary tasks during the scheduling process of data relay satellite network (DRSN), are difficult to be predicted, which may lead to unsuccessful scheduling of tasks. A high-efficiency and robust DRSN calls for smarter and more flexible disturbances elimination strategies. Here, we unify the above three system disturbances as temporary task arrival and extend the static scheduling model of DRSN. Specifically, we derive and define a scheduling model that unifies the static scheduling and dynamic scheduling processes. Meanwhile, we propose a k-step dynamic scheduling algorithm considering breakpoint transmission (k-steps-BT) to solve the above model. Based on the principle of backtracking algorithm and search tree, k-steps-BT can eliminate disturbances quickly by rescheduling tasks and can determine the rescheduling scheme when temporary tasks arrive. Finally, extensive experiments are carried out to verify the proposed model and algorithm. The results show that the proposed model and algorithm can significantly improve the task completion rate of dynamic scheduling without drastic adjustments to the static scheduling scheme.

Keywords Data relay satellite network · Dynamic scheduling · Breakpoint transmission · Required service duration changed · Resource failure · Temporary tasks

Introduction

With the rapid increase of the earth observation satellites, the demand for space transmission has been growing greatly [1, 2]. Satellites located in low and medium orbits mainly transmit the data of tasks through ground terminals (GTs) and the transmissions largely depend on the coverage of GTs. However, due to the relatively low coverage of the GTs, the visible ranges and visible opportunities of user satellites and GTs are limited. If the user satellites are directly connected to the ground network, a large number of GTs need to be deployed in different areas on the Earth, resulting in high operation and maintenance costs [3, 4]. Additionally, the deployment of GTs is restricted by geographical and political conditions, e.g., it is not easy to construct GTs either on the ocean or in oversea countries [5]. The above two reasons lead to a low data transmission capacity between user satellites and GTs. To satisfy the increasing demands of space data transmission, data relay satellites (TDRSs) are widely used for data relay. Theoretically, three TDRSs can achieve global coverage and provide data relay service for user satellites

✉ Xinjiang Chen
xinjiangchen@outlook.com; xinjiangchen@stu.pku.edu.cn

Zongling Li
leezl0519@163.com

Qizhang Luo
qz_luo@csu.edu.cn

Guohua Wu
guohuawu@csu.edu.cn

Ling Wang
wangling@tsinghua.edu.cn

1 Beijing Key Laboratory of Embedded Real-Time Information Processing Technique, Beijing Institute of Technology, Beijing 100081, China

2 Institute of Spacecraft System Engineering, China Academy of Space Technology, Beijing 100094, China

3 College of Engineering, Peking University, Beijing 100091, China

4 School of Traffic and Transportation Engineering, Central South University, Changsha 410073, China

5 Department of Automation, Tsinghua University, Beijing 100084, China

located in low–medium orbits, thereby significantly extending the visible duration between the user satellites and GTs, and improving the data transmission efficiency [6, 7].

More recently, researchers have proposed the concept of a space information transmission system named data relay satellite network (DRSN) [8, 9]. An overview of DRSN is shown in Fig. 1, in which DRSN consists of a backbone network layer, a user layer, and a terrestrial network layer. The backbone network consisting of TDRSs is responsible for providing data relay services to the user layer. The user layer includes various satellites, near-space vehicles, and deep-space explorers, which are collectively called user spacecrafts (USs). Moreover, the terrestrial network layer involves networked GTs as well as the management center (MC) of DRSN. In this sense, researchers generally transform the space data transmission problem into a task scheduling problem of DRSN (DRSNSP), which is crucial to be investigated for improving the data transmission efficiency of DRSN.

Previous studies on DRSNSP mainly focused on static scheduling problems. However, in practical applications, due to unexpected system disturbances such as service duration change, resource failure, and temporary tasks, the scheduling process may yield uncertainties that cause interference to the task scheduling of DRSN. Hence, studies on the dynamic scheduling of DRSN considering system disturbances are more in line with the requirements of actual scenarios [10]. Additionally, breakpoint transmission (i.e., task splitting) has been proved effective to improve the task completion rate and resource utilization of DRSN [9]. It would be promising to obtain better scheduling results by incorporating the breakpoint transmission mechanism into the dynamic scheduling of DRSN. To the best of our knowledge, there are no previous studies on the dynamic scheduling model and algorithm for DRSNSP simultaneously considering breakpoint transmission and hybrid system disturbances (including the required service duration change, resource failure, and temporary tasks).

To eliminate system disturbances and improve the task completion rate, we extend the static scheduling model of DRSN to make it suitable for dynamic scheduling. Meanwhile, we design a k-step dynamic scheduling algorithm considering breakpoint transmission (k-steps-BT) algorithm to solve the dynamic scheduling problem. The major contributions and innovations of this study are:

(1) A breakpoint transmission mechanism is considered in the dynamic scheduling problem of DRSN. The mechanism aims to transform original tasks into a set of subtasks, thereby fully utilizing the idle visible duration between user satellites and GTs to further improve the data transmission efficiency.
(2) A scheduling model suitable for both static and dynamic scheduling of DRSN is constructed, in which we transform the hybrid disturbances dynamic scheduling of DRSN into the temporary tasks' dynamic scheduling.
(3) A k-steps-BT algorithm is designed to solve the dynamic scheduling problem considering temporary tasks. Moreover, we verify that the proposed model and algorithm could significantly improve the task completion rate without drastic adjustment to the static scheduling scheme.

The remainder of this paper is organized as follows. The section "Related work" reviews the related work. In the section "Dynamic scheduling model", we extend the static scheduling model and make it suitable for both static and dynamic scheduling of DRSN. In the section "Dynamic scheduling algorithm", we introduce the k-steps-BT algorithm to solve the model. The section "Experimental results and discussion" is devoted to the experimental results and discussion. "Conclusion" contains some concluding remarks.

Related work

To solve DRSNSP, researchers have proposed different scheduling models and algorithms. The scheduling types involve static scheduling [9–11], dynamic scheduling [8, 12, 13], and hybrid static and dynamic scheduling [14]. Existing models include mixed-integer linear programming model [4, 9, 11], constraint satisfaction model [12, 14, 15], and graph model [7, 16]. The algorithms involve exact algorithms [4, 17], heuristics [10, 18–20] and metaheuristics [7, 14, 21, 22].

As for exact algorithms, Rojanasoonthon et al. [4] constructed a mixed-integer linear programming model for DRSNSP based on the modeling method of parallel machine scheduling problem, and proposed a branch-and-bound algorithm to obtain the optimal solution of the above model. Reddy et al. [17] designed a dynamic programming algorithm to solve the task scheduling of DRSN with single access link.

Regarding the rule-based heuristics, He et al. [8] constructed a dynamic scheduling model considering hybrid tasks, i.e., common tasks, emergency tasks, and temporary tasks, and proposed a stochastic optimization framework and two heuristics to solve the above model. The results show that the proposed algorithms can effectively increase the task completion rate. To solve the static scheduling problem of DRSN, Liu et al. [16] transformed the graph model into a mixed-integer linear programming model considering the antenna slewing time, and proposed a polynomial-time algorithm to solve DRSNSP. Wang et al. [5] designed a repeated game framework to solve the task conflict problem during the scheduling process of DRSN. Additionally, Wang [18] et al. proposed a two-stage heuristic algorithm

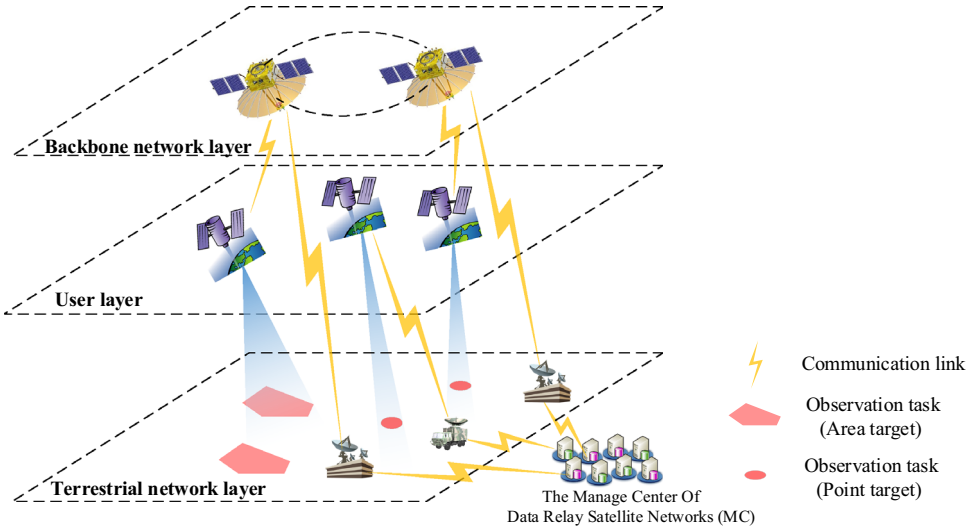


Fig. 1 An overview of a DRSN

based on a hierarchical scheduling strategy. The results show that compared with the greedy randomized adaptive search procedure (GRASP), the proposed algorithm has a greater gain in improving the task completion rate. Dai et al. [13] designed a real-time scheduling algorithm for the emergency task scheduling problem of DRSN. The algorithm divides the task scheduling process into initial scheduling, emergency task dynamic scheduling, and final scheduling. The results verify that the proposed algorithm can significantly improve the completion rate of emergency tasks compared with genetic algorithm (GA) and simulated annealing algorithm (SA). To solve the task scheduling problem of DRSN, Chen et al. [9] designed two heuristic algorithms considering breakpoints that can apply to the scenarios with different scheduling requirements, respectively. The experimental results show that, compared with the conventional algorithms, the proposed algorithms can effectively improve resource utilization. Wu et al. [10] designed a heuristic algorithm based on conflict resolution, which effectively solved the task scheduling of DRSN considering multiple service time windows. Zhang et al. [20] decomposed the long-term task scheduling and multi-dimensional resource management problem of DRSN into numerous mixed-integer programming problems and proposed an effective heuristic algorithm to deduce the data drop rate and energy consumption compared with the existed methods.

With regard to metaheuristics, as the most representative algorithms of metaheuristics, GA and artificial bee colony (ABC) algorithm have achieved remarkable results in solving DRSNSP. Song et al. [7] introduce the knowledge about satellite scheduling into DRSNSP, and proposed a knowledge-based GA to improve the operating efficiency of DRSN. To solve hybrid static and dynamic scheduling problems of DRSN, Deng et al. [14] proposed an improved GA as the static scheduling algorithm and a heuristic algorithm based on preemptive scheduling as the dynamic scheduling algorithm. Meanwhile, Fang et al. [21] proposed an improved GA to solve the static scheduling problem of DRSN. The experimental results show the effectiveness of the algorithms proposed by Deng et al. and Fang et al. Zhuang et al. [22] verified by simulation experiments that the artificial bee colony (ABC) algorithm is better than the simulated annealing (SA) in solving the static scheduling problem of DRSN.

More recently, the data-driven and knowledge-driven algorithms such as learning-based algorithms are widely used for solving task scheduling of DRSN. Li et al. [23] addressed the scheduling problem of DRSN by formulating it as a sequential decision-making problem. They proposed a Markov decision model and developed a deep reinforcement learning (DRL) algorithm to solve the scheduling problem. The experimental results demonstrated the superiority of the proposed algorithm over traditional heuristic and metaheuristic approaches, particularly for large-scale optimization problems. To enhance the emergency dispatch capability of DRSN, He et al. [24] proposed a multi-agent reinforcement learning-based algorithm for the satellite resource allocation problem considering heterogeneous resource competition and cooperation. The experimental results showed that the proposed algorithm can effectively improve the task completion rate. To handle the challenges posed by uncertain channel conditions, Wang et al. [25] developed a DRL algorithm for the cooperative resource scheduling of the ground terminal network and relay satellite network. To reduce system delays and enhance the quality of experience for users, Wang et al. [26] and Huang et al. [27] introduced a multi-layer neural network model and knowledge-driven network access approach to simulate task request and transmission, respectively. Fan et al. [28] developed a multi-layer network graph aggregation model for DRSN to group tasks with similar spatial–temporal characteristics, and demonstrated that the proposed

method can effectively reduce the solution time through a real-world case study. To solve the matching problem of satellite resources and data transmission requirements, Song et al. [29] and Liu et al. [30] proposed an improved k-means algorithm, respectively. Both studies successfully validated the effectiveness of their proposed algorithms across different scenarios.

According to the above literature review, it can be found that:

(1) Although exact algorithms can obtain the optimal solutions when solving small-scale scheduling problems, it is difficult to find the optimal solution within a reasonable time when coping with a large-scale problem. Meanwhile, heuristics and metaheuristics are more promising for solving DRSNSP.
(2) Breakpoint transmission mechanism has been proved effective to improve the task completion rate and resource utilization of DRSN [9]. However, there are few studies on dynamic scheduling of DRSN considering considers breakpoint transmission.
(3) Most studies on static scheduling of DRSN, and a few studies on dynamic scheduling, but there are no studies on mathematical models as well as algorithms for dynamic scheduling of DRSN simultaneously considering breakpoint transmission and hybrid system disturbances.

Dynamic scheduling model

In this section, we unify the hybrid system disturbances as the disturbance of temporary tasks and extend the static scheduling model of DRSN. The explicit symbols used throughout this section are listed in Table 1.

Table 1 Notations

Symbol	Description
T	A set of original tasks, $T = \{1, 2, \cdots, t\}$
T_α	A set of tasks in T that can be split
T_β	A set of tasks without splitting in T
$\overline{T}$	A subtask set transformed from T
R	A set of antennas,$R = \{1, 2, \cdots, r\}$
US	A set of user spacecrafts, $US = \{us_1, us_2 \cdots us_t\}$, where us_t is the user spacecraft that correspond to task t
VT_{tr}	A set of visible time windows, $vt_{tr}^j = [vts_{tr}^j, vte_{tr}^j]$, $vt_{tr}^j \in VT_{tr}$ is the j^{th} visible time window of task$t$ on antenna r, where vts_{tr}^j and vte_{tr}^j denote the start time and end time, respectively
$[sts_t, ste_t]$	The service time window of task t (i.e., the earliest start time and latest end time of task t)
$\left[avts_{tr}^j, avte_{tr}^j\right]$	The available time window of task t (i.e., the intersection of the visible time window and the service time window)
$\left[acts_{trj}, acte_{trj}\right]$	The actual start time and end time of task t
p_t	The priority of task t
R_t	The set of available antennas for task t
d_t	The required service duration of task t
acd_{trj}	The actual service duration of task$t$ in the time window vt_{tr}^j
$Adjust$	The alignment time of the antenna before executing a task
Rec	The reset time of the antenna after a task scheduled
x_{trj}	Binary decision variable, $x_{trj} = 1$ if task t is scheduled to be serviced by antenna r in the visible time window vt_{tr}^j, otherwise $x_{trj} = 0$

Static scheduling model

In this study, we incorporate a breakpoint transmission mechanism into the task scheduling of DRSN. The breakpoint transmission mechanism has been introduced in the static scheduling of DRSN [9], in which a single task can be reasonably split into multiple subtasks and thus scheduled in multiple time windows. Mathematically, the formulation of the static scheduling model of DRSN considering breakpoint transmission is shown in Eq. (1), where f_1 is the objective function, which means to maximize the task completion rate and $Z_C(t)$ is a function that splits task t into n subtasks. The first part of the numerator in f_1 is the number of completed tasks without using breakpoint transmission, and the second part is the number of completed tasks using breakpoint transmission. Note that, if an original task is split into multiple subtasks, it would be scheduled only if all subtasks are successfully scheduled. Constraints C1 ~ C6 represent task requirement constraints, and C7 ~ C9 represent resource usage constraints. Constraints C1–C2 guarantee that the actual service duration of task t equals its required service duration. Constraint C3 indicates the actual start time of the original task or subtask is not earlier than the earliest start time. Constraint C4 means that the actual end time of the original task or subtask is not later than the latest end time. Constraint C5 says that each original task or subtask only be served by one antenna. Constraint C6 guarantees that each original task or subtask is executed in one visible time window. Constraint C7 restricts that the actual start time of the original task or subtask should not be earlier than the start time of the visible time windows and Constraint C8 requires

that the actual end time of the original task or subtask is no later than the end time of the visible time window. Constraint C9 guarantees that the antenna serves one task at a time. Constraint C9 guarantees that the antenna serves one task at a time, where $acts_{mrj}$ and $acts_{nrj}$ represent the actual start timing of task n and m, $acte_{mrj}$ and $acte_{nrj}$ represent the actual end timing, and *adjust* and *rec* mean the alignment and reset time of the antenna r before/after a task scheduled, respectively

$$\max\ f_1 = \frac{\sum_{t\in T_\beta}\sum_{r\in R}\sum_{vt_{tr}^j\in VT_{tr}} x_{trj} + \sum_{\{t_1,t_2,\cdots,t_n\}=Z_C(t),t\in T_\alpha}\sum_{r\in R}\sum_{vt_{tr}^j\in VT_{tr}} \min\left(x_{t_1rj},\cdots,x_{t_nrj}\right)}{|T|}$$

$$s.t.\begin{cases} C1: acd_{trj} = d_t,\ \forall t\in T_\beta,\ r\in R,\ vt_{tr}^j\in VT_{tr} \\ C2: \sum_{n=1} x_{t_nrj}\cdot acd_{t_nrj} = d_t,\ \{t_1,t_2,\cdots,t_n\} = Z_C(t),\ \forall t\in T_\alpha,\ r\in R,\ vt_{tr}^j\in VT_{tr} \\ C3: acts_{trj}\cdot x_{trj}\geq sts_t,\ \forall t\in \overline{T},\ r\in R,\ vt_{tr}^j\in VT_{tr} \\ C4: \left(acts_{trj}+acd_{trj}\right)\cdot x_{trj}\leq ste_t,\ \forall t\in \overline{T},\ r\in R,\ vt_{tr}^j\in VT_{tr} \\ C5: \sum_{r\in R} x_{trj}\leq 1,\ \forall t\in \overline{T},\ vt_{tr}^j\in VT_{tr} \\ C6: \sum_{vt_{tr}^j\in VT_{tr}} x_{trj}\leq 1,\ \forall t\in \overline{T},\ r\in R \\ C7: acts_{trj}\cdot x_{trj}\geq vts_{tr}^j,\ \forall t\in \overline{T},\ r\in R,\ vt_{tr}^j\in VT_{tr} \\ C8: (acts_{trj}+acd_{trj})\cdot x_{trj}\leq vte_{tr}^j,\ \forall t\in \overline{T},\ r\in R,\ vt_{tr}^j\in VT_{tr} \\ C9: \left[acts_{mrj}-adjust,\ acte_{mrj}+rec\right]\cap\left[acts_{nrj}-adjust,\ acte_{nrj}+rec\right]=\emptyset,\ \forall m\in\overline{T},\ n\in\overline{T},\ r\in R \\ x_{trj}\in\{0,1\},\ \forall t\in\overline{T},\ r\in R,\ vt_{tr}^j\in VT_{tr} \end{cases} \quad (1)$$

Dynamic scheduling model

Here, we extend the static scheduling model and make it suitable for dynamic scheduling of DRSN. Specifically, we consider three disturbances: required service duration change, resource failure, and temporary tasks. Note that the above three disturbances are unified. Specifically, for a task with a changed required service duration, we can treat it as a temporary task. In doing so, we can cancel the scheduling scheme of the task, and release the resources occupied by it (if the task is scheduled in the static scheduling scheme). Resource failure mainly refers to TDRSs or Uss' malfunction. Tasks executed by the failed resource will fail to be scheduled, and thus, these tasks need to be adjusted. Similar to the processing method of tasks with changed required service duration, tasks that are scheduled unsuccessfully due to resource failure can also be regarded as temporary tasks. Hence, all tasks affected by the above three disturbances considered in this study can be converted to temporary tasks. Based on the above analysis, the static scheduling model can be appropriately extended to obtain the dynamic scheduling model.

(1) Evaluation metric of dynamic scheduling scheme

Without loss of generality, the greater the deviation between the dynamic scheduling scheme and the static scheduling scheme, the more adjustment cost [31–33]. Therefore, to evaluate the deviation between the dynamic scheduling scheme and the static scheduling scheme, here we introduce the dynamic disturbance measure (DDM) as an evaluation metric to measure the deviation, which reads

$$\psi\left(S_{sta},\ S_{dyn}\right) = \lambda_{del}n_{del} + \lambda_c n_c, \quad (2)$$

where ψ represents DDM, S_{sta} represents the static scheduling scheme, S_{dyn} represents the dynamic scheduling scheme, n_{del} represents the number of tasks deleted from the static scheduling scheme due to required service duration change, resource failure or temporary tasks, λ_{del} represents the weight of the deleted tasks, n_c represents the number of tasks adjusted from the static scheduling scheme, and λ_c represents the weight of the adjusted tasks. Meanwhile, λ_{del} and λ_c satisfy

$$\lambda_{del}+\lambda_c = 1. \quad (3)$$

(2) Extension of the task set

Based on the above analysis, we convert the system disturbances involving required service duration change, resource

failure and temporary tasks to the disturbance of temporary tasks. Let T_{add} be a set of temporary tasks, we have

$$T_{new} = T \cup T_{add}, \tag{4}$$

where T_{new} represents the task set for dynamic scheduling.

We use the breakpoint transmission mechanism in the dynamic scheduling model. That is, a single task can be reasonably split into multiple subtasks and thus scheduled in multiple time windows. Note that, the task splitting process is performed dynamically during the task scheduling process, which is detailed in the section "Dynamic scheduling algorithm". Suppose T^{α}_{add} to be the task set in T_{add} that can be split and $\overline{T}^{\alpha}_{add}$ to be the subtask set of the task in T^{α}_{add}. Namely, we have

$$\overline{T}^{\alpha}_{add} = \left\{t_n | \{t_1, t_2, \cdots, t_n\} = Z_C(t), t \in T^{\alpha}_{add}\right\}, \tag{5}$$

where $Z_C(t)$ is a function that splits task t into n subtasks.

Let T^{β}_{add} represent the set of tasks without splitting in T_{add}, that is

$$T_{add} = T^{\alpha}_{add} \cup T^{\beta}_{add}. \tag{6}$$

To unify the original task and the subtask, similarly, we convert the original task set T^{β}_{add} into a subtask set $\overline{T}^{\beta}_{add}$. Namely, we have

$$\overline{T}^{\beta}_{add} = T^{\beta}_{add}. \tag{7}$$

In doing so, we can convert the temporary task set T_{add} into a subtask set $\overline{T}_{add}$, that is

$$\overline{T}_{add} = \overline{T}^{\alpha}_{add} \cup \overline{T}^{\beta}_{add}. \tag{8}$$

To make the model formulation more concise, based on the above analysis, the task set of dynamic scheduling T_{new} can be converted into a form of subtask set. Namely

$$\begin{aligned} T_{new} &= T \cup T_{add} \\ &= \left(T_{\alpha} \cup T_{\beta}\right) \cup \left(T^{\alpha}_{add} \cup T^{\beta}_{add}\right) \\ &= \left(\overline{T}_{\alpha} \cup \overline{T}_{\beta}\right) \cup \left(\overline{T}^{\alpha}_{add} \cup \overline{T}^{\beta}_{add}\right) \\ &= \overline{T} \cup \overline{T}_{add}, \end{aligned} \tag{9}$$

where $\overline{T}_{\alpha}$ and $\overline{T}_{\beta}$ are the subtask sets corresponding to T_{α} and T_{β}, which can be obtained according to the derivation method of $\overline{T}^{\alpha}_{add}$ and $\overline{T}^{\beta}_{add}$. $\overline{T}$ is the subtask set form of T derived by breakpoint transmission [9].

(3) Extension of the set of visible time windows

The method for generalization of the visible time window set of the dynamic scheduling model is as follows. We add the visible time window of each task in the temporary task set to the visible time window set of the static scheduling model. We denote VT^{new}_{tr} as a visible time window set of dynamic scheduling model, and thus, for each $t \in T_{new}$, its visible time windows are defined as

$$vt^{new}_{trj} = [vts^{new}_{trj}, vte^{new}_{trj}],\ j = 1, 2\cdots, \forall r \in R,\ vt^{new}_{trj} \in VT^{new}_{tr}. \tag{10}$$

(4) Extension of the constraints set

The dynamic scheduling of temporary tasks is also constrained by task requirements (C1 ~ C6 in Eq. (1)) and resource usage constraints (C7 ~ C9 in Eq. (1)). Therefore, the static scheduling model with specific adjustments could be used for dynamic scheduling.

(5) Extension of the static scheduling model

To sum up, we extend the static scheduling model of DRSN considering breakpoint transmission into a dynamic scheduling model suitable for temporary tasks, which reads

$$\max f = \frac{\sum_{t\in T_{new}^{\beta}} \sum_{r\in R} \sum_{vt_{trj}^{new}\in VT_{tr}^{new}} x_{trj} + \sum_{\{t_1,t_2,\cdots,t_n\}=Z_C(t),t\in T_{new}^{\alpha}} \sum_{r\in R} \sum_{vt_{trj}^{new}\in VT_{tr}^{new}} \min\left(x_{t_1rj},\cdots,x_{t_nrj}\right)}{|T_{new}|}$$

$$\text{s.t.}\begin{cases}
\text{C1: } acd_{trj} = d_t,\ \forall t \in T_{new}^{\beta},\ r \in R,\ vt_{trj}^{new} \in VT_{tr}^{new} \\
\text{C2: } \sum_{n=1} x_{t_nrj} \cdot acd_{t_nrj} = d_t,\ \{t_1, t_2, \cdots, t_n\} = Z_C(t),\ \forall t \in T_{new}^{\alpha},\ r \in R,\ vt_{trj}^{new} \in VT_{tr}^{new} \\
\text{C3: } acts_{trj} \cdot x_{trj} \geq sts_t,\ \forall t \in T_{new},\ r \in R,\ vt_{trj}^{new} \in VT_{tr}^{new} \\
\text{C4: } \left(acts_{trj} + acd_{trj}\right) \cdot x_{trj} \leq ste_t,\ \forall t \in T_{new},\ r \in R,\ vt_{trj}^{new} \in VT_{tr}^{new} \\
\text{C5: } \sum_{r\in R} x_{trj} \leq 1,\ \forall t \in T_{new},\ vt_{trj}^{new} \in VT_{tr}^{new} \\
\text{C6: } \sum_{vt_{tr}^{j}\in VT_{tr}} x_{trj} \leq 1,\ \forall t \in T_{new},\ r \in R \\
\text{C7: } acts_{trj} \cdot x_{trj} \geq vts_{tr}^{j},\forall t \in T_{new},\ r \in R,\ vt_{trj}^{new} \in VT_{tr}^{new} \\
\text{C8: } (acts_{trj} + acd_{trj}) \cdot x_{trj} \leq vte_{tr}^{j},\ \forall t \in T_{new},\ r \in R,\ vt_{trj}^{new} \in VT_{tr}^{new} \\
\text{C9: } \left[acts_{mrj} - adjust,\ acte_{mrj} + rec\right] \cap \left[acts_{nrj} - adjust,\ acte_{nrj} + rec\right] = \emptyset, \forall m \in T_{new},\ n \in T_{new},\ r \in R \\
x_{trj} \in \{0,\ 1\},\ \forall t \in T_{new},\ r \in R,\ vt_{trj}^{new} \in VT_{tr}^{new}
\end{cases} \tag{11}$$

Hereby, it is important to note that, on the one hand, the above model could be slightly adjusted to be suitable for dynamic scheduling considering the required service duration change and resource failure. On the other hand, although this model is extended from the static scheduling model, it builds a framework that can uniformly characterize the static scheduling and dynamic scheduling of DRSN, that is, it can be applied to both static and dynamic scheduling of DRSN.

Dynamic scheduling algorithm

k-Steps-BT

To improve the task completion rate of dynamic scheduling without drastic adjustments to the static scheduling scheme, here we propose an algorithm named *k*-steps-BT. Based on the principle of backtracking algorithm [34, 35] and search tree [36–38], *k*-steps-BT realizes conflict resolution and reschedules tasks quickly when temporary tasks occur. *k*-Steps-BT includes three modules: (1) temporary task-resource matching; (2) task splitting; (3) adjustment tree expansion and control.

(1) Temporary task-resource matching

The temporary task-resource matching aims to generate a set of available time windows for scheduling the temporary tasks according to the priority of the temporary tasks, similar to [9]. If the available time windows satisfy the service duration of a temporary task, the task will be scheduled by a immediate predecessor. That is, the temporary task is inserted immediately after the earliest start time of the available time windows. If there exist temporary tasks that cannot be scheduled after temporary task-resource matching, we record these tasks and move to module (2).

(2) Task splitting

According to the breakpoint transmission mechanism, a single task will be reasonably split into multiple subtasks and then scheduled in multiple time windows. The process of breakpoint transmission is shown in Algorithm 1.

Algorithm 1 The process of breakpoint transmission

Input: Temporary task set T_{add}, Visible time window set VT_{tr};

Output: Task splitting scheme S_{split};

1: **Initialize** $T_{add}^{\alpha} \leftarrow \varnothing, AVT_{t*} \leftarrow \varnothing$;

2: Sort the task set T_{add} in descending order by priority;

3: **For each** $t^* \in T_{add}$

4: Generate the available resources set for $t*$, $AVT_{t*} \leftarrow \left\{ \left[avts_{t*r}^{j}, avte_{t*r}^{j} \right] \middle| avte_{t*r}^{j} - avts_{t*r}^{j} \geq d_{t*} \right\}$;

5: Calculate each $dur_{t*r}^{j} \leftarrow \left(avte_{t*r}^{j} - avts_{t*r}^{j} \right)$;

6: **If** $\forall\ dur_{t*r}^{j} < d_{t*}$

7: $T_{add}^{\alpha} \leftarrow T_{add}^{\alpha} \cup t^*$;

8: **End if**

9: **End for**

10: **For each** $t^* \in T_{add}^{\alpha}$

11: Sort the AVT_{t*} in descending order by $dur_{t_n^* r}^{j}$;

12: Splitting $t*$ into subtasks $Z_C(t^*) = \left\{ t_1^*, t_2^*, \cdots, t_n^* \right\}$;

13: **For each** $dur_{t_n^* r}^{j}$

14: **If** $\exists \sum_{n=1} dur_{t_n^* r}^{j} \geq d_{t*}$

15: Insert task, and update the visible time window VT_{tr} and record the splitting scheme S_{split};

16: **Else if** $\exists \sum_{n=1} dur_{t_n^* r}^{j} < d_t$

17: Task scheduling failed and transfer to module 3);

18: **End if**

19: **End for**

20: **End for**

Specifically, if each available time window $\left[avts_{t*r}^{j}, avte_{t*r}^{j} \right]$ of the temporary task $t*$ in available resources set AVT_{t*} cannot meet scheduling requirements of $t*$, we add $t*$ to the splitting task set T_{add}^{α} (Lines 1–9). To minimize the number of splits (Since the antenna will have an idling time when it switches from one task to another [39] [40], which will cause additional scheduling overhead), we choose the split method based on the minimum number of splits to split tasks (Lines 11–12). If the original task $t*$ can be split into multiple subtasks and be executed in multiple available time windows, $t*$ will be scheduled. Afterward, we update the visible time window VT_{tr} and record the task splitting scheme S_{split}. Otherwise, we transfer to module 3 (Lines 13–19). Additionally, according to different split objects, task splitting can be divided into three types: split of the scheduled tasks, split of the temporary tasks, and split of both scheduled tasks and temporary tasks. The above three task splitting types are shown in Figs. 2, 3, and 4, respectively.

(3) Adjustment tree expansion and control

Figure 5 is a schematic of the adjustment tree expansion and control. As shown in Fig. 5, if temporary task $t*$ cannot be scheduled by the breakpoint transmission, we define the scheduled task whose execution duration intersects the available time windows of temporary task $t*$ as an adjustable task. If temporary task $t*$ is successfully scheduled by removing the adjustable task, we define the adjustable task as a new node of the adjustment tree, that is, we regard the adjustable task as a new temporary task. In doing so, we define the above process as adjustment tree expansion. Sequentially, we perform a new round of adjustment tree expansion of the

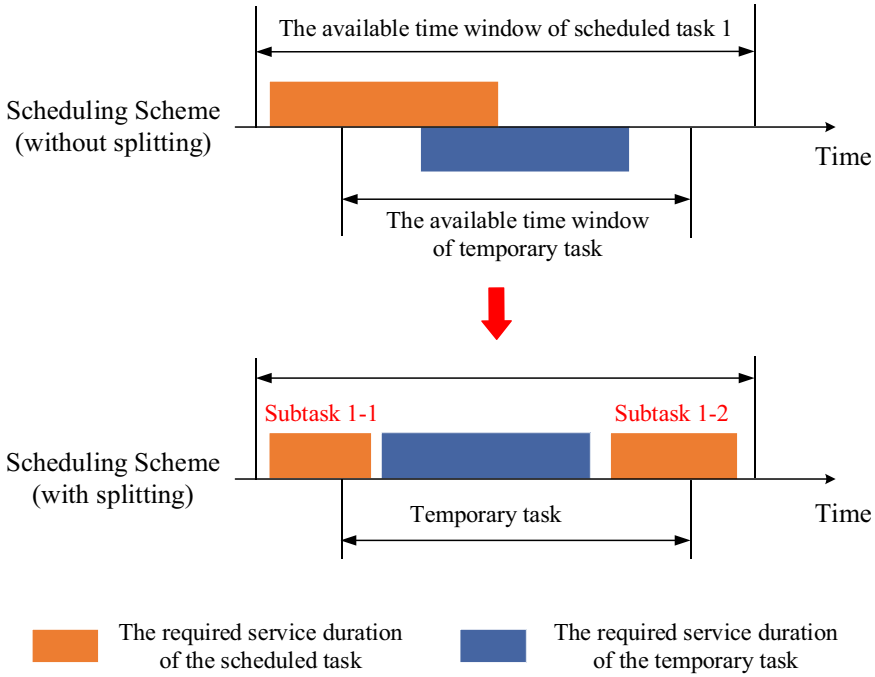


Fig. 2 Split of the scheduled tasks

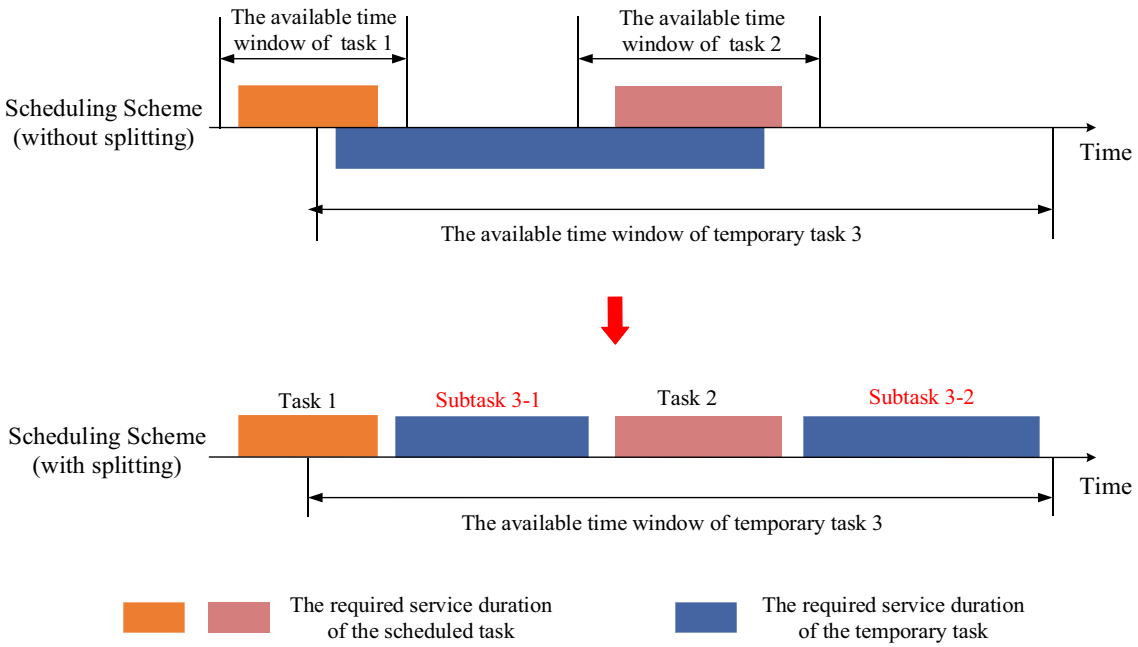


Fig. 3 Split of the temporary tasks

new temporary tasks, and this adjustment operation can be regarded as the 2nd step of the adjustment of $t*$. We repeat the above process until the k-steps is reached, that is, the value of k determines the adjustment depth of the k-steps-BT algorithm. In this sense, it is obvious that the dynamic scheduling of temporary tasks is a process of constantly expanding the adjustment tree.

Note that, as k increases, the size of the adjustment tree will grow exponentially. In addition, we consider the task splitting in the scheduling of each node task in the adjustment tree, which further increases the running time of the algorithm. Consequently, to meet the timeliness requirements of dynamic scheduling, we control the size of the expansion tree. In this study, we generally take $k = 2$.

Algorithm framework

In summary, we first present the pseudo-code of k-steps-BT in Algorithm 2. Then, we analyze the algorithm flow and computation complexity of k-steps-BT.

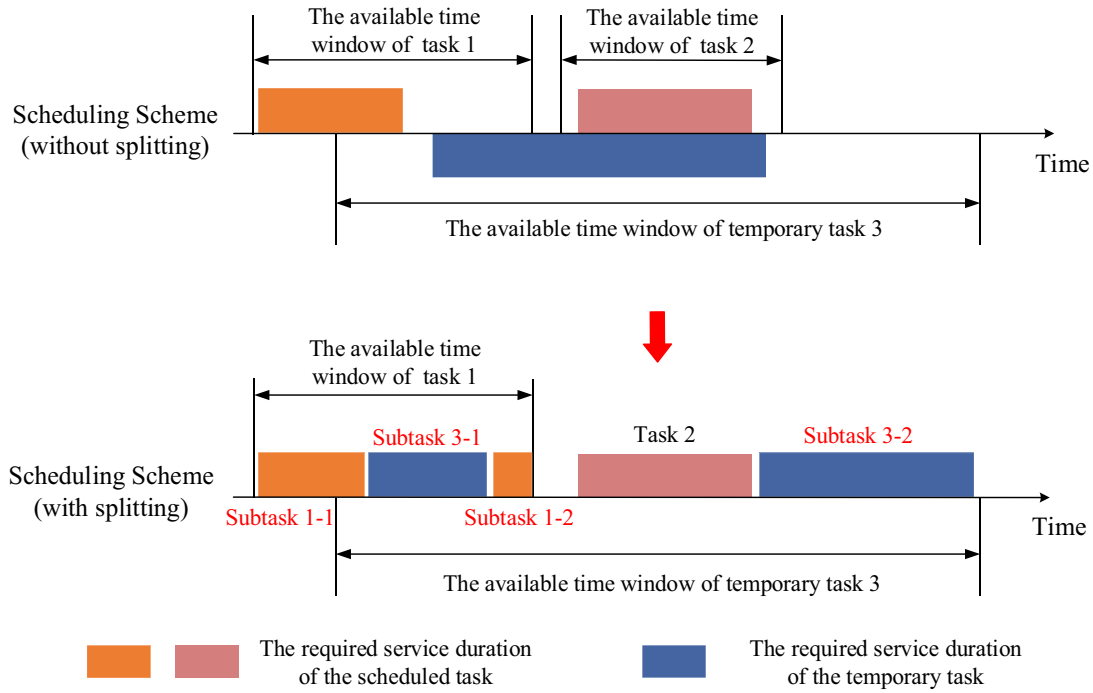


Fig. 4 Split of the scheduled tasks and temporary tasks

(1) Analysis of Algorithm Flow

As shown in Algorithm 2, during the initialization phase of k-steps-BT, we let the current search step $k = 1$, the maximum search step $k_{\max} = 2$ and the dynamic scheduling scheme $S_{dyn} = \emptyset$ (Line 1). We perform task-resource matching for each temporary task $t* \in T_{add}$ to generate the available time window set AVT_{t*} (Lines 3–4). If the available resources meet the scheduling requirements of the $t*$, $t*$ will be scheduled. Then, we update the visible time window VT_{tr} and add the scheduling information of $t*$ to the dynamic scheduling scheme S_{dyn} (Lines 4–7). If the available resources do not meet the scheduling requirements of $t*$, we schedule $t*$ by breakpoint transmission (i.e., Algorithm 1). In doing so, if $t*$ is successfully scheduled, we update the visible time window and add the scheduling information of $t*$ to the dynamic scheduling scheme S_{dyn} (Lines 8–13). If $t*$ is unsuccessfully scheduled by breakpoint transmission, the execution duration of the scheduled tasks are used as the resources of the $t*$. Sequentially, we perform the temporary task-resource matching to generate the adjustable task set for $t*$ in the current search step (Lines 14–17). If $t*$ is scheduled after releasing the resources occupied by the adjustable task t, we remove the scheduling scheme of t, regard t as a new temporary task, and then transfer to $k+1$ search step (Lines 18–21). Note that temporary tasks generally have higher priority than common tasks. Hence, it makes sense to prioritize temporary tasks. The algorithm repeats the above process until the maximum search step $k_{\max}$ is reached. If the temporary task is failed to be scheduled during $k_{\max}$ steps, it reveals that the adjustment cost of the temporary task $t*$ is too high. In this sense, the temporary task $t*$ is rejected to be scheduled (Lines 15–29). Finally, we integrate the static scheduling scheme S_{sta} and dynamic scheduling scheme S_{dyn} as a final scheduling scheme (Line 33).

(2) Complexity Analysis of k-steps-BT

We can observe from Algorithm 2 that the time complexity of temporary task sorting is $O\left(|T_{add}|^2\right)$ (Line 2) and the complexity of temporary task-resource matching is $O(|T_{add}| \times |VT|)$ (Lines 3–4), where $|T_{add}|$ represents the number of temporary tasks, $|VT|$ indicates the total number of visible time windows and is defined as $|VT| = \sum_{t \in T} \sum_{r \in R} |VT_{tr}|$, where $|VT_{tr}|$ represents the number of visible time windows of task t on antenna r. The complexity of scheduling temporary tasks using breakpoint transmission is $O\left(|T_{add}| \times |VT|^2\right)$ (Lines 5–13). The complexity of adjustable task set generation of the current step is $O(|T_{add}| \times |ST|)$ (Lines 16–17), and the complexity of adjustment tree expansion is $O\left(|T_{add}| \times \left(|ST| + |ST|^2 + \cdots + |ST|^{k_{\max}}\right)\right)$ (Lines 18–30), where $|ST|$ is the number of scheduled tasks. Since $|ST| \leq |T|$, the time complexity of k-steps-BT is $O\left(|T_{add}| \times \left(|T| + |T|^2 + \cdots + |T|^{k_{\max}}\right)\right)$, which reveals that the time complexity of k-steps-BT increases exponentially with the increase of k. Therefore, to meet the timeliness requirements of dynamic scheduling, we control the size of the expansion tree. Specifically, we take $k_{\max} = 2$.

Algorithm 2 k-steps dynamic scheduling algorithm considering breakpoint transmission (k-steps-BT)

Input: Static scheduling scheme S_{sta}, Temporary task set T_{add}, Visible time window set VT_{tr};

Output: Dynamic scheduling scheme S_{dyn};

Initialize $k \leftarrow 1, S_{dyn} \leftarrow \varnothing, k_{\max}$;

Sort the temporary task set in descending order by priority;

For each $t^* \in T_{add}$

  Generate a set of available time windows for task t^*,
  $AVT_{t^*} \leftarrow \left\{ \left[avts_{t^*r}^j, avte_{t^*r}^j \right] \middle| avte_{t^*r}^j - avts_{t^*r}^j \geq d_{t^*} \right\}$, $dur_{t^*r}^j \leftarrow \left(avte_{t^*r}^j - avts_{t^*r}^j \right)$;

  If $\exists\, dur_{t^*r}^j \geq d_{t^*}$

    Insert task, and update the visible time window VT_{tr};

    Add t^* to the scheduling scheme S_{dyn};

  Else if $\forall dur_{t^*r}^j \leq d_{t^*}$

    Using breakpoint transmission (**Algorithm 1**) to schedule task t^*

    If t^* scheduling successful

      Update the visible time window VT_{tr}, Add t^* to the scheduling scheme S_{dyn};

    End If

  End if

  If task t^* scheduling failed using breakpoint transmission

    Repeat

      Take the resource occupied by the scheduled task as the resource of t^*;

      Generate a set of available time windows AVT_{t^*} for t^*;

      For each scheduling task t in S_{sta}

        If $\exists \left[acts_{trj}, acte_{trj} \right] \cap \left[avts_{t^*r}^j, avte_{t^*r}^j \right] \neq \varnothing$ and $dur_{t^*r}^j \leq \min\left(acte_{trj}, avte_{t^*r}^j \right) - \max\left(acts_{trj}, avts_{t^*r}^j \right)$

          Delete t from S_{sta}, schedule task t^* and regard t as a new temporary task t^*;

          $k \leftarrow k+1$;

        End if

      End for

    Until $k > k_{\max}$

    If the temporary task t^* scheduling successful

      Update the visible time window VT_{tr}, and add t^* to the scheduling scheme S_{dyn};

    Else

      Task t^* scheduling failed;

    End if

    $k \leftarrow 1$;

  End if

End for

$S_{dyn} \leftarrow S_{sta} \cup S_{dyn}$

Experimental results and discussion

Simulation parameter setting

The proposed algorithm is implemented in Matlab and executed on an Intel(R) Core(TM) i5 2.80 GHz, with 8.0 GB RAM. The scheduling period is from 2019–10-08 00:00:00 to 2019–10-09 00:00:00. The application scenario involves 300 common tasks (the task ID are 1 to 300), 6 temporary tasks (the task ID are 301 to 306), 10 USs, 3 TDRSs, and each TDRS carries a single access antenna. The benchmark and simulation scenarios' settings are detailed in [9].

Dynamic scheduling scheme analysis

In this section, we first use a state-of-the-art metaheuristic solver named adaptive variable neighborhood descent combined with a tabu list (AVND-TL) which is our previous work [9] as the static scheduling algorithm to find a scheduling scheme of common tasks. Then, we use k-steps-BT to solve the dynamic scheduling of temporary tasks, where $k = 2$. The results are shown in Table 2, where mode "0" means the task scheduling fails, " + 1" means the task is scheduled without using breakpoint transmission, and " + 2" means the task is successfully scheduled using breakpoint transmission.

In Table 2, the scheduling duration of tasks 16 and 47 in the static scheduling scheme conflicts with the service time windows of temporary tasks 304 and 305, respectively. The scheduling scheme of the scheduled tasks 16 and 47 is adjusted by k-steps-BT algorithm and the conflict is further resolved during the dynamic scheduling process. That is, the scheduled tasks 16, 47 and the temporary tasks 304 and 305 are all successfully scheduled by using k-steps-BT algorithm. Additionally, the temporary tasks 303 can be scheduled by breakpoint transmission. Note that, when $k = 2$, the temporary task 301 fails to be scheduled, because the adjustment cost is too high. Therefore, based on the preliminary results in Table 2, it can be verified that the k-steps-BT algorithm is effective.

Comparisons and sensitivity analysis

In this section, we discuss the impacts of different numbers of temporary tasks and different values of k for k-steps-BT. For the sake of comprehensive and fair comparisons, we prepare a set of real-world scenarios each scenario involves 300 common tasks, 38 temporary tasks, 10 USs and 3 TDRSs. Additionally, we compare the proposed approach with a state-of-the-art method named preemptive dynamic scheduling algorithm (PDSA) [14] and a scalable framework for dynamic scheduling of DRSN called whole rescheduling algorithm (WRA) [12] on the same scheduling instances. Note that WRA is an algorithm framework and we adopt AVND-TL [9] in the framework of WRA.

As for dynamic disturbance measure (DDM), we calculate the DDM between the dynamic scheduling schemes generated from the WRA, PDSA, and k-steps-BT with the static scheduling scheme by Eq. (2), where we set $\lambda_{del} = 0.8$ and $\lambda_c = 0.2$. Specifically, the number of tasks deleted from the static scheduling scheme, i.e., n_{del}, can be determined by comparing the dynamic with static scheduling schemes. The number of tasks adjusted from the static scheduling scheme, i.e., n_c, can be calculated in the same way. Note that, although a single task can be split into multiple subtasks, and it (including its subtasks) can be adjusted multiple times during the dynamic scheduling, it is only recorded as one adjustment.

(1) Experimental results of different number of temporary tasks

We set $k = 2$ in k-steps-BT algorithm and present the experimental results with different number of temporary tasks Fig. 6. With regard to temporary task completion rate, we can observe from Fig. 6a that WRA is better than PDSA and k-steps-BT, and PDSA is better than k-steps-BT in the case with a small-scale temporary task, in which the number of temporary tasks is less than 30. That is, the number of temporary tasks completed by k-steps-BT is more than that of the PDSA when there are more than 30 temporary tasks. As for total task completion rate, WRA is better than k-steps-BT and k-steps-BT is significantly better than PDSA. Regarding the results of dynamic disturbance measure, we can observe from Fig. 6c that k-steps-BT is substantially better than PDSA and PDSA is better than WRA. Note that, the dynamic disturbance measure of the PDSA rises sharply when the number of temporary tasks is more than 20, while the dynamic disturbance measure of k-steps-BT is still relatively stable. Regarding the running time of algorithms, PDSA and k-steps-BT are substantially better than WRA.

It is noteworthy that the advantage of the k-steps-BT algorithm in enhancing task completion rate becomes less pronounced when the count of temporary tasks is limited. In such cases, the DRSN conserves available resources, enabling most temporary tasks to be scheduled without requiring sophisticated adjustments. In contrast, the WRA algorithm profoundly ensures optimal task completion rate by subjecting all tasks to rescheduling. Nevertheless, as the temporary task volume expands, both the computing time and dynamic disturbance measurement of WRA experience substantial escalation. Furthermore, in comparison to the k-steps-BT algorithm, the PDSA algorithm obviates the need for retrospective scheme adjustments. It directly arranges temporary tasks based on their priorities, showcasing a pronounced advantage in computing time and temporary task completion rate. However, for extensive temporary task

Table 2 Dynamic scheduling scheme (no. 301–306 is the scheduling scheme of temporary tasks)

Task ID	USs ID	Mode	Service duration/Executed TDRS ID	Duration (sec)
16	5	+ 1	[04:59:28, 05:11:59]/1	751
47	4	+ 1	[18:51:05, 18:59:41]/1	516
301	5	0	–	–
302	8	+ 1	[00:36:57, 00:41:39]/2	282
303	5	+ 2	[01:41:59, 01:44:12]/1、[02:23:50, 02:25:57]/3、[02:55:45, 02:57:19]/2	354
304	5	+ 1	[04:45:30, 04:52:29]/1	419
305	4	+ 1	[18:36:16, 18:41:37]/1	321
306	10	+ 1	[19:55:32, 19:59:50]/3	258

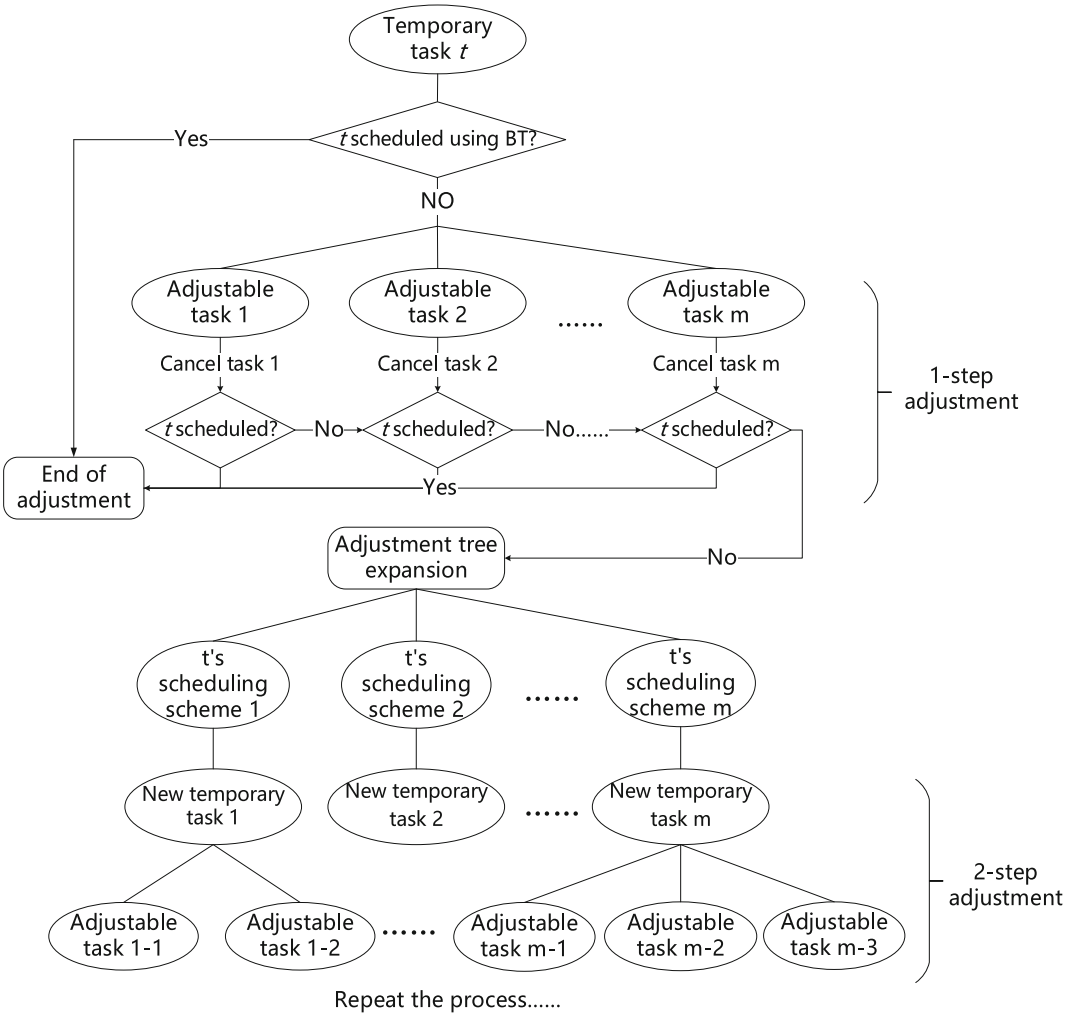
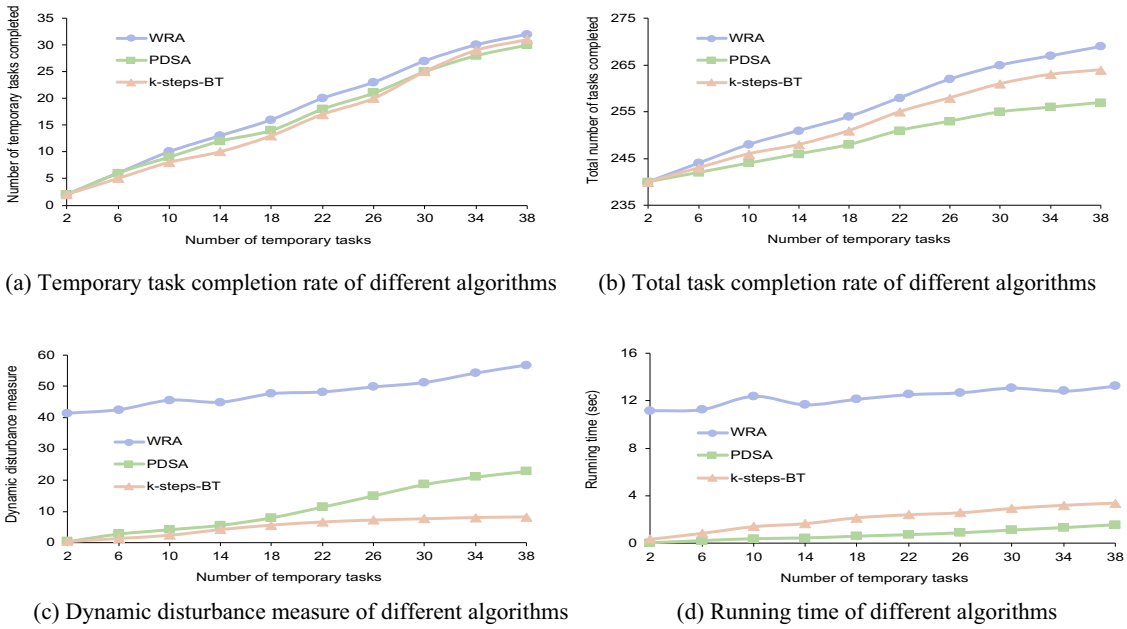


Fig. 5 Schematic of adjustment tree expansion



(a) Temporary task completion rate of different algorithms

(b) Total task completion rate of different algorithms

(c) Dynamic disturbance measure of different algorithms

(d) Running time of different algorithms

Fig. 6 Experimental results of different number of temporary tasks ($k = 2$)

scheduling, the PDSA algorithm's performance in terms of task completion rate and dynamic disturbance measurement still significantly trails that of the k-steps-BT algorithm.

Given the aforementioned observations, it is evident that the k-steps-BT algorithm offers significant advantages over the PDSA and WRA algorithms in effectively addressing the dynamic scheduling challenges within DRSN involving extensive temporary tasks. Conversely, the PDSA algorithm appears better suited for tackling scheduling problem entailing smaller-scale temporary tasks.

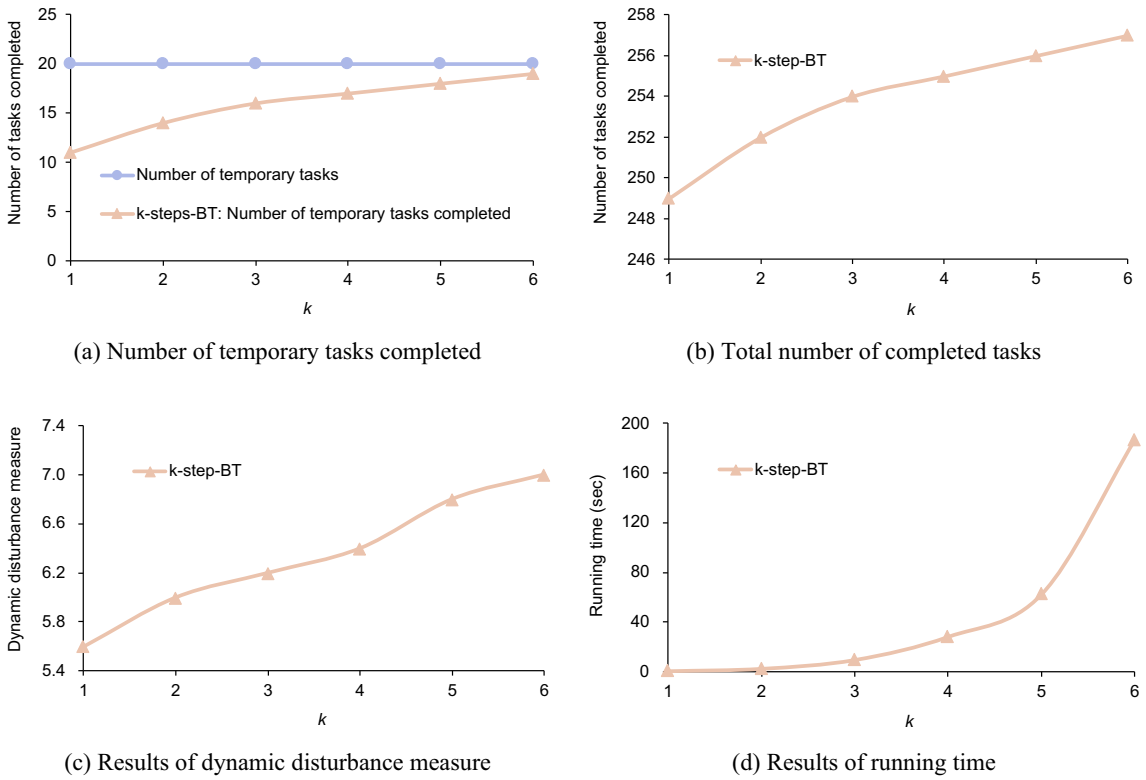
(2) Experimental results of different values of k

Here, we analyze how the value of k affects performance of k-steps-BT. The experimental results are reported in Fig. 7, in which the number of temporary tasks is 20. The results show that k-steps-BT with different values of k have different performances in the number of temporary tasks completed, the total number of tasks completed, the dynamic disturbance measure, and the running time of the algorithm. Specifically, the increase of the k can effectively increase the completion rate of temporary tasks and the total task completion rate. However, the dynamic disturbance measure and running time increases sharply with the increase of k.

Discussion

The dynamic scheduling of temporary tasks will inevitably interfere with the scheduling tasks. Therefore, to solve the dynamic scheduling problem caused by arriving of temporary tasks, we designed an algorithm named k-steps-BT with a faster response speed and lower adjustment cost. Meanwhile, the effectiveness of the k-steps-BT is proved through extensive experiments. In this section, we further discuss the applicability of k-steps-BT based on the experimental results.

The experimental results of Fig. 6 show that although WRA can effectively improve the task completion rate, the higher running time and dynamic disturbance measure of WRA do not meet the requirements of dynamic scheduling (timeliness and minor adjustment of scheduling scheme), which is a typical example of "no free lunch" [41, 42]. In addition, PDSA focuses overly on improving the completion rate of temporary tasks, while ignoring the total task completion rate, which leads to a higher dynamic disturbance measure. However, the number of temporary tasks completed of the k-steps-BT is more than that of the PDSA when there are more than 30 temporary tasks, and the total number of tasks completed of the former is substantially greater than that of the latter. Meanwhile, the dynamic disturbance measure of the PDSA rises sharply when the number of temporary tasks more than 20, while the dynamic disturbance measure of k-steps-BT is still relatively stable. The results of the above situation can be accounted for the following two reasons. On the one hand, k-steps-BT with a backtrack mechanism are more flexible for dynamic adjustment of tasks, which can handle conflicts between temporary tasks and scheduled tasks well. On the other hand, PDSA can only schedule temporary tasks by deleting the scheduled tasks because of the limited resources in the case with large-scale temporary tasks, which will lead to large adjustments



(a) Number of temporary tasks completed

(b) Total number of completed tasks

(c) Results of dynamic disturbance measure

(d) Results of running time

Fig. 7 Experimental results of different value of k

in the dynamic scheduling scheme (i.e., the dynamic disturbance measure). Note that the time overhead of k-steps-BT is slightly more than that of PDSA (the maximum gap is within 2 s), which is reasonable for the timeliness requirement of dynamic scheduling. In this sense, k-steps-BT substantially outperform PDSA.

It is noteworthy that k-steps-BT achieves a good balance in terms of the temporary task completion rate, total completion rate, dynamic disturbance measure, and running time. Specifically, the numerical value of k determines the search depth of k-steps-BT in the solution space according to the idea of "exploration and exploitation" [43]. The$k$-steps-BT with a larger numerical value of$k$ has a robust global searching ability but takes more time and vice versa. Therefore, we can select k-steps-BT with different k values for scenarios with different user requirements. If users pay more attention to the total task completion rate and temporary task completion rate, the k-steps-BT with a larger numerical value of k is more appropriate. Conversely, if users focus on the timeliness of scheduling, the k-steps-BT with a smaller numerical value of k is more appropriate.

Note that a great balance trade-off between timeliness and performance should made in practical applications, and we can observe from Fig. 7 that the running time and dynamic disturbance measure increase sharply with the increase of the k, which do not meet the dynamic scheduling requirements of DRSN. In this sense, to balance the performance and timeliness of k-steps-BT, k is recommended to be 2.

Conclusion

In this paper, first, we transform the dynamic scheduling considering hybrid system disturbances into the temporary tasks dynamic scheduling of DRSN. Second, we construct a dynamic scheduling model considering breakpoint transmission, which builds a framework that can uniformly characterize the static scheduling and dynamic scheduling of DRSN. Finally, we design an efficient algorithm to solve the above model and verify the effectiveness of the proposed model and algorithm by extensive experiments. Specifically, the proposed algorithm can improve the task completion rate with minimum adjustment cost and reasonable scheduling overhead when system disturbance occurs.

Future works will be focused on applying the breakpoint transmission to distributed scheduling considering hybrid system disturbances and designing more efficient algorithms to improve the performance of DRSN.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China (No. 62073341) and in part by the High-performance Computing Platform of Peking University.

References

1. Xiong M, Xiong W, Liu Z (2023) A co-evolutionary algorithm with elite archive strategy for generating diverse high-quality satellite range schedules. Compl Intell Syst. https://doi.org/10.1007/s40747-023-01008-4
2. Sun J, Chen X, Zhang J, Yao W (2021) A niching cross-entropy method for multimodal satellite layout optimization design. Compl Intell Syst 7:1971–1989
3. Deng C, Guo W, Hu W (2018) Dynamic scheduling algorithm for data relay services in next-generation tdrs systems. J Aerosp Infor Syst 15(11):665–669
4. Rojanasoonthon S, Bard JF, Reddy SD (2003) Algorithms for parallel machine scheduling: a case study of the tracking and data relay satellite system. J Operat Res Soci 54(8):806–821
5. Wang L, Jiang C, Kuang L, Wu S, Huang H, Qian Y (2018) High-efficient resource allocation in data relay satellite systems with users behavior coordination. IEEE Trans Veh Technol 67(12):12072–12085
6. Shu Z, Song A, Wu G, Pedrycz W (2023) Variable reduction strategy integrated variable neighborhood search and nsga-ii hybrid algorithm for emergency material scheduling. Comp Syst Model Simul 3(2):83–101
7. Song Y, Xing L, Wang M, Yi Y, Xiang W, Zhang Z (2020) A knowledge-based evolutionary algorithm for relay satellite system mission scheduling problem. Comput Ind Eng 150:106830
8. He L, Li J, Sheng M, Liu R, Guo K, Zhou D (2019) Dynamic scheduling of hybrid tasks with time windows in data relay satellite networks. IEEE Trans Veh Technol 68(5):4989–5004
9. Chen X, Li X, Wang X, Luo Q, Wu G (2021) Task scheduling method for data relay satellite network considering breakpoint transmission. IEEE Trans Veh Technol 70(1):844–857
10. Wu G, Luo Q, Zhu Y, Chen X, Feng Y, Pedrycz W (2021) Flexible task scheduling in data relay satellite networks. IEEE Trans Aerosp Electron Syst. https://doi.org/10.1109/TAES.2021.3115587
11. Rojanasoonthon S, Bard J (2005) A GRASP for parallel machine scheduling with time windows. INFORMS J Comput 17(1):32–51
12. Deng B, Jiang C, Kuang L, Song G, Lu J, (2017). Preemptive dynamic scheduling algorithm for data relay satellite systems. In *2017 IEEE Int Conf Communi.* pp. 21–25
13. Dai CQ, Li C, Fu S, Zhao J, Chen Q (2020) Dynamic scheduling for emergency tasks in space data relay network. IEEE Trans Veh Technol 70(1):795–807
14. Deng B, Jiang C, Kuang L, Guo S, Lu J, Zhao S (2017) Two-phase task scheduling in data relay satellite systems. IEEE Trans Veh Technol 67(2):1782–1793
15. Horan S (2003) Nontracking antenna performance for inertially controlled spacecraft using TDRSS. IEEE Trans Aerosp Electron Syst 39(4):1263–1269
16. Liu R, Sheng M, Xu C, Li J, Wang X, Zhou D (2016) Antenna slewing time aware mission scheduling in space networks. IEEE Commun Lett 21(3):516–519
17. Reddy SD, Brown WL (1986) Single processor scheduling with job priorities and arbitrary ready and due times. Beltsville Comp Sci Corp 70(1):1–11
18. Wang L, Jiang C, Kuang L, Wu S, Fei L, Huang H (2018) Mission scheduling in space network with antenna dynamic setup times. IEEE Trans Aerosp Electron Syst 55(1):31–45
19. Lin P, Kuang L, Chen X, Yan J, Lu J, Wang X (2014) Adaptive subsequence adjustment with evolutionary asymmetric path-relinking for TDRSS scheduling. J Syst Eng Elect 25(5):800–810
20. Zhang S, Cui G, Wang W (2021) Joint Data Downloading and Resource Management for Small Satellite Cluster Networks. IEEE Trans Veh Technol 71(1):887–901
21. Fang, Y. S., & Chen, Y. W. (2006). Constraint programming model of TDRSS single access link scheduling problem. In *2006 Int Conf Mach Learn Cybernet*. pp. 948–951
22. Zhuang, S., Yin, Z., Wu, Z., & Shi, Z. (2014). The relay satellite scheduling based on artificial bee colony algorithm. *In 2014 Int Symp Wireless Pers Multimed Communi (WPMC)*. 635–640.
23. Li J, Wu G, Liao T, Fan M, Mao X, Pedrycz W (2023) Task scheduling under a novel framework for data relay satellite network via deep reinforcement learning. IEEE Trans Veh Technol. https://doi.org/10.1109/TVT.2022.3233358
24. He H, Zhou D, Sheng M, Li J (2023) Hierarchical cross-domain satellite resource management: an intelligent collaboration perspective. IEEE Trans Commun 71(4):2201–2215
25. Wang, Y., Zhou, D., Sheng, M., & Li, J. (2022). Adaptive and Cooperative Resource Scheduling for Satellite-Terrestrial Networks. In GLOBECOM 2022–2022 IEEE Global Commun Con. pp. 3923–3928
26. Wang, L., Yang, J., & Zheng, X. (2022). Fast Assessment of Requests Schedulability in Data Relay Satellite Systems: A Multilayer Neural Network Method. In 2022 IEEE/CIC Int Conf on Commun China. pp. 389–394
27. Huang, L., Sun, R., Cheng, N., Hui, Y., & Liang, D. (2023). Delay-Oriented Knowledge-Driven Resource Allocation in SAGIN-Based Vehicular Networks. In 2023 IEEE Wireless Communi Network Conf. pp. 1–6
28. Fan H, Yang Z, Zhang X, Wu S, Long J, Liu L (2022) A novel multi-satellite and multi-task scheduling method based on task network graph aggregation. Expert Syst Appl 205:117565
29. Song Y, Ou J, Wu J, Wu Y, Xing L, Chen Y (2023) A cluster-based genetic optimization method for satellite range scheduling system. Swarm Evol Comput 79:101316
30. Liu H, Chu Y, Zhang Y, Hou W, Li Y, Yao Y, Cai Y (2021) Strategy of multi-beam spot allocation for GEO data relay satellite based on modified k-means algorithm. Mathematics 9(15):1718
31. Sari NN, Jahanshahi H, Fakoor M, Volos C, Nikpey P (2020) Optimal robust control approaches for a geostationary satellite attitude control. Int J Autom Control 14(3):333–354
32. Wu G, Luo Q, Du X, Chen Y, Suganthan PN, Wang X (2022) Ensemble of Metaheuristic and Exact Algorithm Based on the Divide-and-Conquer Framework for Multisatellite Observation Scheduling. IEEE Trans Aerosp Electron Syst 58(5):4396–4408
33. Yan J, Xing L, Li C, Zhang Z (2021) Multicommodity flow modeling for the data transmission scheduling problem in navigation satellite systems. Complex Syst Model Simulat 1(3):232–241
34. Sharma A, Goyal N, Guleria K (2021) Performance optimization in delay tolerant networks using backtracking algorithm for fully credits distribution to contrast selfish nodes. J Supercomput 77(6):6036–6055
35. Civicioglu P (2013) Backtracking search optimization algorithm for numerical optimization problems. Appl Math Comput 219(15):8121–8144

36. Tchendji VK, Zeutouo JL (2019) An efficient cgm-based parallel algorithm for solving the optimal binary search tree problem through one-to-all shortest paths in a dynamic graph. Data Sci Eng 4(2):141–156
37. Kizilkaya B, Caglar M, Al-Turjman F, Ever E (2019) Binary search tree based hierarchical placement algorithm for IoT based smart parking applications. Int Things 5:71–83
38. Afek Y, Kaplan H, Korenfeld B, Morrison A, Tarjan RE (2014) The CB tree: a practical concurrent self-adjusting search tree. Distrib Comput 27(6):393–417
39. Perea F, Vazquez R, Galan-Viogue J (2015) Swath-acquisition planning in multiple-satellite missions: An exact and heuristic approach. IEEE Trans Aerosp Electron Syst 51(3):1717–1725
40. Almeida F, Giménez D, López-Espín JJ, Pérez-Pérez M (2013) Parameterized schemes of metaheuristics: Basic ideas and applications with genetic algorithms, scatter search, and GRASP. IEEE Trans Syst, Man, Cyber Systems 43(3):570–586
41. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
42. Ho YC, Pepyne DL (2002) Simple explanation of the no-free-lunch theorem and its implications. J Optim Theory Appl 115(3):549–570
43. He Y, Xing L, Chen Y, Pedrycz W, Wang L, Wu G (2020) A generic markov decision process model and reinforcement learning method for scheduling agile earth observation satellites. IEEE Trans Syst Cybernet Systems 52(3):1463–1474